WideFS: Network application interface for Microsoft Flight Simulator

Freeware by Pete Dowson, 25th July 2003

Support Forum: http://www.forums.simflight.com/dowson

Version 6.00

WIDEFS IS *NOT* FREE!

WideFS 6 will *not* run on Flight Simulator without FSUIPC also being installed, and at least Version 3 of FSUIPC is required. WideFS ***must*** be registered with FSUIPC, using an Access Key, before it will work. You can purchase an access key from http://secure.simmarket.com/product_info.php?products_id=539. Full details about purchasing access keys and registering both FSUIPC and WideFS are also to be found in the early pages of the FSUIPC User Guide. You can save money if you purchase keys for FSUIPC and WideFS at the same time.

This package contains just the following parts:

WideServer.dll	The server module itself, for the FS Modules folder
WideServer.ini	Initial configuration for the server
WideClient.exe	The client program, running on client PCs
WideClient.ini	Initial configuration for the clients
WideFS.doc	This document, in Word 97 format
WideFS.pdf	This document, in Acrobat format

Note: All my Windows based FS software is always available in the latest versions from http://www.schiratti.com/dowson

IMPORTANT NOTES

This is NOT a program for linking several copies of Flight Simulator, nor will it allow scenery or cockpit views on multiple PCs. It is purely for running EXTERNAL applications on a separate PC whilst still communicating with FS as if on the same PC. For such applications to work through WideFS they need to access Flight Simulator through the interface provided and defined by FSUIPC (or originally, for FS98 programs, FS6IPC).

Also be sure to match versions of WideClient and WideServer. This is often essential for correct connections, as the protocol being used is developed and may be subject to subtle changes in each new version.

Introduction: What is WideFS?

There are now many add-on programs written to interface to Microsoft's Flight Simulator. Nearly all of these exchange information with FS by using an interface devised many years ago (originally for FS95, the first Windows version of FS) by Adam Szofran. This interface provided methods for programs running outside the Simulator, but in the same PC, to read various values from FS and write different values back. This enabled a wealth of programs such as moving maps, weather control, external instrumentation, extended autopilots and even flight management systems to be developed.

The interface, using a technique known as "Inter-Process Communication" (IPC), was extended by Adam to operate on FS98 (using a module called FS6IPC—the Microsoft internal name for FS98 was FS6). But this was the last in that line as Adam was understandably recruited by Microsoft to work from within instead of without. <G>

It was during this time that I put together my first PC network. An excellent package called 'WidevieW' became available, written by Luciano Napolitano, which allowed several installations of FS on different PCs across a Network to show different outside views from the same user aircraft in flight. I used this as a basis for making Chris Brett's EFISv2 run on a separate PC from the Flight Simulator to which it was interfacing. This was done by fooling the remote copy of WidevieW into thinking it was running with FS, whereas it was actually a program of my devising (the 'client'), and similarly another program intercepted the data at the Simulator end (the 'server'). This package was released as freeware under the name WideEFIS, for obvious reasons.

Not much later I learned a little more about network programming—but not much. Mostly I just re-used the simpler code examples in the Microsoft development kits. I used the IPX/SPX protocol because it looked simpler to understand, was fast in operation, and also because WidevieW used it too. Applying all this resulted in WideFS, a package consisting of two parts:

1. **WideClient.exe**, a program which on the one hand provides what looks like an exact replica of the FS6IPC interface, fooling FS application programs into interfacing with it thinking that it was FS98, and on the other transmitted these exchanges and received data back across the Network, using the SPX (and later TCP) protocol.

2. **WideServer.dll**, a Flight Simulator module, installed into and actually running as part of FS98, which serviced the messages and requests from WideClient and interfaced directly into the same data inside FS as did FS6IPC.

When FS2000 came out I wrote a new IPC module for that simulator, but kept not only the same interface as that devised by Adam, but also the same variable access. This was deliberate, to allow programs written for FS98 to run with FS2000 unchanged. I also made it run on FS98 itself, and extended it to CFS1, CFS2 and FS2002, keeping as far as possible complete compatibility with that FS98 standard. Since this was now a 'universal' IPC interface for Flight Simulator I called it FSUIPC, breaking away from Adam's specific FS version implications.

Nowadays FSUIPC has taken over the interfacing role inside FS, and WideServer interfaces to it, so both parts are needed even if all the application programs are running on client PCs. This is because with FS2000 and then more so with FS2002, access to many of the internal parts of FS, which was so easy in FS98, has become very difficult. Internally it is no longer a matter of simple memory offset and size management and mapping, but often also involves calling procedures elsewhere in FS, and following pointer chains into data private to other modules. Rather than duplicate such complex code, it is now all concentrated into FSUIPC, freeing the WideFS parts to deal predominantly with the efficient running of the Server to Client (or Server to many Clients) exchanges.

So, does this answer the question, 'what is WideFS?' Maybe, if you've followed its history well enough. Otherwise, for now, just consider WideFS as a package that enables you to run Microsoft Flight Simulator applications on computers which aren't even running Flight Simulator.

And **please** don't confuse it with WidevieW. The similarity in the name is because of its history, as described. Both packages deal with spreading the simulation experience across several PCs in a network—hence the 'Wide' part. But you use WidevieW for multiple views, to see around you, and you use WideFS for additional instrumentation, moving maps, and weather control programs, running on separate PCs and lessening the burden of having too many processes on the all-important Simulation PC itself.

Good flying!

Installation

Follow these steps:

On the Flight Simulator PC

1. Place WideServer.dll and WideServer.Ini in the Flight Simulator Modules folder, which is inside your main FS folder.

That's it. FS will load it automatically for you.

Note that you must also have FSUIPC.dll (version 3.00 or later) in the Modules folder. WideFS is totally dependent upon that module, which is available as a separate download.

When you run FS, go to the FSUIPC options (Modules menu, FSUIPC entry), and **Register** your WideFS by entering the access key. (You *did* get one, didn't you? If not, use the link given above).

For full details of this process, please see the FSUIPC User Guide. Note also that you have to close down and restart FS in order for the registration to become effective. You don't have to repeat this each time, though. Just keep a safe copy of the FSUIPC.KEY file (from the Modules folder). You may have to re-register if you re-install Windows, or want to use FS on another PC. But you are not restricted to using WideFS in only one FS installation.

On each client PC

(i.e. all the other PCs in your Network on which you are NOT going to be running FS itself, but on which you want to run some application which interfaces to FS).

1. Place WideClient.Exe and WideClient.ini in any desired folder (I recommend either in the same folder as your FS utility, or a dedicated folder called WideClient).
2. Also install here any bitmap you want WideClient to display as background for your utilities (i.e. replacing the FS panel for which they may have been designed). If you do want to use a bitmap, you will need to edit the WideClient.ini file and add the bitmap filename using the "Background=filename" parameter. (See the section on WideClient parameters, below).
3. Drag a shortcut for WideClient onto your desktop.

Now, when you have your Network correctly configured, you can either run WideClient (from the shortcut), THEN your utility program, or you can, for more convenience, edit the WideClient.ini file and enter the full pathname for your utility in the Run1= parameter line. WideClient will then start your application for you. Up to three programs can be started in this way—see the section on WideClient parameters later.

Configure your Network

Decide first whether you are going to use the IPX/SPX protocol for WideFS, or the TCP/IP protocol. WideFS now uses TCP/IP by default. Here are the pros and cons as I see them at present:

IPX/SPX Good	IPX/SPX Bad	TCP/IP Good	TCP/IP Bad
Faster, simpler protocol	Not a default windows protocol. Complications in setting it up, mainly on Windows NT and its successors (2000 and XP).	Installs by default on new networks so less for user to do to get it working	Slower more complex protocol. The drop in performance is more likely to be noticed with 10Mbit Networks and on Windows 98. Faster Networks using Windows XP is not quite so tardy in comparison.
No known danger of virus spreading or cross-network invasion from an Internet connection	Even more complex problems can apparently arise when using a Network with mixed Windows 95/98/Me systems and NT/2000/XP.	Compatible across all versions of Windows without many problems.	Possibility of virus or other invasions across the LAN from an internet connection on any one component connection to the Internet. Some sort of Internet firewall protection advised.
No measurable overhead to affect performance of Flight Simulator	Seems that Microsoft are neglecting IPX/SPX protocol more and more in recent Operating Systems, and this isn't likely to reverse. It could get worse in the future.	Microsoft pay more attention to fixing, developing and streamlining TCP/IP with each new operating system, so it is likely to get better, not worse.	Danger of introducing regular stutters to Flight Simulator unless you explicitly assign IP addresses to each PC on the Network.

Previously, because WideFS was originally written for IPX/SPX, and because there is a noticeable difference in performance on Windows 98 even with a 100Mbit network, the default protocol, assumed on installation, was actually IPX/SPX. Because of the complications, especially with Windows XP, the default is now TCP/IP. If you wish to change to IPX/SPX, install it using the Control Panel—Network program. Then, before running WideFS, just check through the points listed in the section **Notes for trouble-shooting IPX/SPX**, below.

To use IPX/SPX you need to add this line to the [Config] section of the WideServer.ini file (add the [config] section too if your ini file hasn't even got one!), and also to each WideClient.ini file:

UseTCPIP=No

Apart from the possibility (mentioned below) of needing a "ServerNode" parameter adding to the WideClient.ini files, there are no further changes required to WideFS.

If you are using Windows 2000 or XP, or a mixture including either of these, then I would recommend staying with TCP/IP. WideFS is not so fast then, but it will be fast enough and smoother and easier than trying to get IPX/SPX working nicely. When you do this, try to make TCP/IP the only protocol enabled on your LAN. Don't install IPX/SPX as well.

To use TCP/IP you only need to add this line to the [Config] section of each WideClient.ini file (add the [Config] section too if your ini file hasn't even got one!):

ServerName=NameOfServer

The ServerName is the name you gave to the PC on which you will be running Flight Simulator with WideServer. If you prefer you can use the Server's IP address instead of the name, thus:

ServerIPAddr=192.168.0.3

If both Name and Address are provided only the latter is used. Note that if the IP Address for the server is not one of those treated by Windows as 'local', you may find WideClient attempting to connect via your Internet connection. I'm not really sure how all that works, but you should be safe if all your networked PCs have assigned IP addresses in the form "192.168.0.N" where N runs from 1 to 255. To assign IP addresses (at least in Windows 98—NT and so on may be different)—open up the Network Properties dialogue, find and highlight the line mentioning "TCP/IP" bound to your local Network adapter (e.g. "TCP/IP->3COM Ethernet Adapter"), then click the Properties button. In the IP Address tab, select "Specify an IP Address", then set the IP Address to 192.168.0.N (N numbering your PCs) and the subnet Mask to 255.255.255.0.

Incidentally, assigning specific IP addresses in this way also reduces periodic but regular stutters in Flight Simulator, caused by the Network drivers querying the Network to get an address assigned (or something like that).

Please be aware of possible problems, especially when using TCP/IP, which may be introduced by Routers and Firewalls, whether hardware or software. Some firewalls (ZoneAlarm is one) will need you to give WideClient permission to connect to WideServer, and maybe vice versa. This may be needed each time you update the WideFS program, as it will be seen as a change and therefore a new potential danger. Do not think that closing the front-end of such programs solves this—it may make it worse. Best to officially tell the firewall that you are happy with the WideFS components.

You may also find the web site www.helmig.com useful in solving Network problems, whether with IPX/SPX or TCP/IP.

Running WideFS

WideClient can be up and running even when there's no WideServer program elsewhere on your network to serve it. It will simply keep trying to connect until one is ready. Similarly it will re-connect if a server is closed and re-opened.

Likewise, WideServer is very tolerant of clients starting and stopping. Each server can serve many clients, but each client only talks to one server.

Simply run Flight Simulator on the server machine (it will load WideServer), and run WideClient (plus your utility program(s) on your other machines. That is all there is to it! Apart from some possibly displays in Flight Simulator's title bar there will be no other sign in Flight Simulator that WideFS is doing its job, and you should not notice any affect on frame rates at all.

With FS2002 and FS2004 you can set a limit on the Frame Rate which FS will keep to. I have found that WideFS operates much more smoothly and efficiently if you use this to prevent FS running away with the processor. With a reasonably fast machine, say 2 GHz and more, try frame rate limits of 30–40 (20–30 for FS2004), but reduce this on slower machines, down to about 15–20 on 1 GHz PCs.

Note that the number of current connections is shown in the title bar of Flight Simulator. This may fluctuate if responses to clients aren't currently being sent. This is because the client assumes that a lack of response for many tries means a disconnected link, so opens a new one. The server detects disused links after a while, and closes them, but there's an overlapping period whilst more connections may actually be open than there are clients.

Some programs (Chris Brett's EFIS and John Hnidec's Moving Map are examples) place their windows ***within*** the Simulator window, or now, in this case the WideClient window. But most utilities don't actually need WideClient visible at all—you can minimise it (or simply set an INI file parameter to do this).

However, taking EFIS as the example, when it is up and running, it will place its Windows inside the blank grey window provided by WideClient, or over an optional background bitmap which you can set up via the INI file. You will want to resize the windows to suit your screen and desired layout of EFIS. Click on the FMC button to get the FMC window and position that as you like. I

find the LARGE size of the main EFIS window best. The parameters in the files as provided suit an 800x600 resolution screen setting. The window sizes and positions will be remembered for next time.

Once you have everything running you can read the sections towards the end of this document, detailing the WideServer and WideClient INI file parameters, to see what adjustments you may wish to apply to get the best performance for your set up. In particular you can try different settings for the "Timeout" parameter in WideClient.ini. The default is 12 mSecs, which seems to suit most programs.

Questions and Answers on how WideClient handles data

Q: Is WideFS a simple passthrough conduit to FS, or does it cache data locally?

A: WideClient maintains a memory map of all of the locations ever requested since it started running. When the values are requested by the client applications, it gets data from there and gives it to the client in a direct response. If there are data items which have not been requested before, it also sends appropriate requests to WideServer, whilst supplying the default value in its memory to the applications (this would be zero). There is an option in the INI (**WaitForNewData**, see below) which actually stops this return being made until WideServer has actually sent the newly requested data—this is actually enabled by default with a 500 mSec timeout. See the DOC.

Except for Write requests from clients, the Network traffic is totally controlled by WideServer, which maintains details of all data items requested by each client, separately, and monitors these for changes. This latter is done at FS frame rates. Only changes are sent out. If a connection dies or closes, the list for that client is cleared (by both ends) and the process starts over.

Q: Is there any recommended polling frequency for applications using WideFS?

A: No. It doesn't really matter, but if you are operating something graphical to run at FS speeds then you probably should try to match average frame rates, for smoothness of your displays. WideServer works better if you limit the FS frame rates to a bit less than its average performance in any case (as mentioned elsewhere in this document), so you would, say, set the limiter to 35 or 30 or 25 fps (according to processor) and poll WideClient at that sort of frequency.

Of course, if your program does a lot of processing or heavy graphics, or is sharing the client PC with other such applications, you might not be able or want to achieve such a frequency.

Since WideClient is supplying all values from its memory, directly, there's no benefit from splitting your data requests into more or less frequently needed items. WideServer will be sending all the changes anyway. If you poll some less frequently you will just be skipping some changes. Of course it is *not* the same for writes to FS. They need more consideration as they will all result in LAN blocks to WideServer and require FS processor time when there.

Q: Is there a log setting I can use to see if I am thrashing WideFS?

A: You don't have any control over the Network operations, excepting how you write things. Certainly you should optimise writes. Don't keep writing the same values to the same places, only write what you need to write, and don't write that frequently that things bog down. But when you are reading you are not affecting anything on the Network. Provided you don't actually ask for any data you don't need (for once you have it will be monitored for changes and all changes sent), then it doesn't matter.

Of course, if your program is also supposed to run well on the FS PC you have to consider the affect you may have on FS's performance. But there's a lot of other factors there apart from calling the IPC interface.

Error reporting

Unless there is something really bad preventing WideClient from running, it will normally keep trying. Any errors it gets are described in text form in a Log file (“WideClient.log”) which you will find in the same folder as the EXE file you are running. Please check there if you see WideClient still waiting for a connection when you believe it should be connecting.

Except for an incorrect or missing Registration, a serious error which prevents WideServer from offering a service will result in a message box appearing whilst Flight Simulator is getting ready. Pressing OK should allow you to continue using FS, but the WideFS link will not be operating. If you configure a “hot key” to restart WideServer (see **RestartHotKey** below, in the section on WideServer.ini options) then you can make WideServer try the whole initialisation sequence again. But really if the error is that serious you will probably have to close FS and sort out the network error being reported in the WideServer.log file (find this in the FS Modules folder.

If you have not yet registered WideFS with FSUIPC, the server will not start but will not have any adverse affect on FS operations. You will see a message in the WideServer.Log file reporting the registration problem.

Just about all of the errors that you are likely to see there will be due to network problems. I cannot give a precise list of messages with their meanings here—WideFS is merely reproducing the error number it gets from Windows software ("WinSock"), and the text is also direct from Microsoft's own documentation. If you get stuck with any real errors, check through ALL the points listed in the trouble-shooting section below.

If you write to me with any problems to look at (and I sincerely hope that this won't be needed, as everything I know is written down here somewhere), please attach a Zip file containing both the WideServer and WideClient LOG and INI files.

Notes for trouble-shooting IPX/SPX
(but also check some of these things, especially 7, 8, and 9, if you are using TCP/IP)

If you want to use IPX/SPX then be warned: the Windows software is not so user-friendly in this area, and seems to be getting less so. One must assume that Microsoft is neglecting this protocol these days. In particular, if either or both Server and Client PCs are running under Windows NT or its successors Windows 2000 or XP, then some additional configuration is almost always needed.

These steps may also prove useful with Windows 95, 98, 98SE and ME. Please try ALL of them before seeking more help. I'm afraid I really know very little about Networks as such, and all the hints listed here are actually contributed by other users.

1. NETBIOS may need to be enabled

You may need to enable NetBIOS over the IPX/SPX connection. Why? Sorry, I don't know, but this seems to be especially important in Windows 95 or 98 if you are running a mixed Windows network.

To do this in Windows 95/98, go into the Network application in Control Panel, select the IPX/SPX protocol, and click Properties. There will be a tab entitled NetBIOS. Select that, and then make sure "I want to enable NetBIOS over IPX/SPX" is selected.

I am not sure if this is also needed in Windows NT, 2000 or XP, nor exactly how you'd do it in those systems. But it is something to check if you cannot get a connection. One successful user has reported that on his working set up NetBIOS is *not* enabled. This is under Windows 98, so this may be an important difference to the above recommendation.

2. Server Node may need specifying

Run Flight Simulator with the WideServer.dll module installed, and then look at the WideServer.Log file that you'll find in the Modules folder. Very near the start there should be a line reading something like

```
17248 ServerNode=0.0.49152.1264.9490
```

The actual numbers won't be the same, but you should find the line okay. Now insert the parameter 'ServerNode= ...', quoting exactly the same numbers after the =. into the [Config] section of each WideClient.ini file you are planning to run to link to this Server.

Note that you will have to re-do this action if you change LAN adapters in the Server PC, or make other similar configuration changes. Also, please see item 7 below, concerning a potential problem with some LAN drivers providing the wrong Node identifiers.

3. Network Address may need setting

I have heard from one chap using a mixed network (Win2000 and Win98) who needed to do a bit more before it would work. To quote him:

"On the Windows 98 PC, under Network Properties, IPX properties, there is an Advanced settings tab. In that list there's something called Network Address. That must be set to a number other than zero. I just typed 13579 (or something like that), rebooted, and now it works."

I'm afraid I can't add anything to that.

4. Ensure IPX/SPX is associated with only one device on each PC

Another user reported that "I had to turn off the IPX protocol on the cable modem network card making sure the one for the LAN was enabled", so it seems likely that IPX/SPX, when added to more than one device, can produce routing problems.

Generally, when you add a protocol in the Network part of the Control Panel, Windows adds it without exception to ***all*** installed network-capable devices. You will need to go through and remove it from everything but the Network adapter.

5. Connections and Sockets

Check that the parameters ‘Maximum Connections’ and ‘Maximum Sockets’, which you'll find under IPX/SPX Properties—Advanced, are both set to a larger number, like 128. The default limits in Windows ME (in particular), and possibly other versions of Windows, are definitely too low!

6. Frame Type

One user reported that the ‘Frame Type’, under IPX/SPX Properties—Advanced, should be set to Ethernet 802.3. This shouldn't really be needed—I’ve always had mine on Auto—but if all else fails it is certainly worth a try.

7. Network drivers don’t always work

One instance has been reported of a presumably buggy LAN adapter driver, a 3COM one for Windows XP, providing incorrect Server Node data to WideServer (see item 2 above). Using the ServerNode parameters logged gave no connection. Reverting to the slightly earlier 3COM drivers actually provided and installed by Windows XP fixed this.

Of the five numbers making it up, the first two identify the specific Network Address. If you’ve only got one then these numbers are normally both 0—but see also item 3 above. The other three constitute the physical address of the adapter card itself, and appears to be called either explicitly the ‘physical address’, or sometimes the ‘MAC address’.

There are ways of verifying this address. These are different for the various Windows versions. On my Windows 98SE installations I can either use the Shareware program SiSoft Sandra Pro, looking at its “Network” information, or use the Windows-supplied DOS program IPConfig (run this in a DOS window with the parameter /ALL). On Windows XP (and presumably also earlier NT versions) you can get the information via local Network Properties—Support—Details.

The address you get will actually consist of six values in hexadecimal notation. To take the example from section 2 above:

```
17248 ServerNode=0.0.49152.1264.9490
```

Each of the last three numbers can be expressed in hexadecimal, so:

```
C000   04F0   2512
```

When these are stored in 6 consecutive 8-bit ‘bytes’ the two halves of each 16-bit number get reversed, due to the way Intel processors store numbers:

```
00 C0 F0 04 12 25
```

This is how this particular physical address (or ‘MAC’ address) will be seen in those other applications and parts of Windows.

8. Network adapters don’t seem to like sharing Interrupts

A lot of the problems I had with my Network in the early days turned out to be all due to the way either the BIOS or Windows had configured my system. The Network card was sharing an interrupt (IRQ) with another card.

The symptoms of the problems were odd. Most things worked fine, but there were strange things going on with WideFS-connected applications. I eventually found that the data transferred across the Network was suffering from occasional corruption—mostly the odd *extra* character being inserted. I proved this by transferring some very large files across, using drag-and-drop in Windows Explorer. Comparing them afterwards showed errors. Not many: about one character in a few million. But enough to make the Network untrustworthy and to occasionally do odd things to applications.

Eventually, after quite a bit of hair loss (and I didn’t have much in the first place!) I found it was IRQ sharing. More hair was then lost trying to correct this, by things like shuffling the PCI cards around, trying to change allocations in the BIOS, and in Windows too. Sorry, I really cannot explain how to do this stuff here—it was all guesswork, trial and error. Mostly error.

9. Network adapters can go wrong!

One other point to realise: LAN adapters are sometimes faulty. Out of seven known brands I’ve had three failures over as many years. It is sometimes worth changing them, or even swapping them between machines, to either eliminate or confirm this as a possible reason for problems. Symptoms may vary from no connection at all, to unreliable or very slow operation, to actual freezing during otherwise normal operation.

…

That’s it. If I get any other suggestions I’ll add them here.

WidevieW compatibility

First, please note that WidevieW is by Luciano Napolitano, and is NOT part of WideFS

If you have enough PCs on your network, you can still run two or more copies of Flight Simulator linked by WidevieW, giving you side views or maps. Only the main (flying) PC will run WideServer. You cannot run WideClient on a PC which is also running Flight Simulator—after all, its job is really to replace it and interface to it across the network connection instead.

If you are running WideFS with the default IPX/SPX protocol, note that the IPX/SPX port number used by WideServer and WideClient defaults to 8002, so avoiding clashes with Wideview's default. For very large networks running more than one flying copy of Flight Simulator, WideServers with different port numbers can link to WideClients with matching port numbers, but there can only be one WideClient on any one PC, as this provides the interface expected by applications and there's no way to differentiate between two or more.

EpicLink compatibility

EpicLink links two or more PCs with EPIC cards, in order to extend the device connectivity to your simulation. EpicLink is similar to WideFS in that it uses a Server-Client relationship across a LAN via the IPX/SPX protocol (only, in this case). However, the IPX/SPX port number used by EpicLink defaults to 8102, so avoiding clashes with both WideFS and WidevieW.

WideServer INI file options

Here all the user features of the server's control parameter file are described. Normally you won't need or want to mess around with many of these parameters. There are all placed into the 'WideServer.ini' file, which goes into the Flight Simulator modules folder along with the WideServer.dll module itself. The filetype 'ini' merely refers to the file as an Initialisation file—reflecting the fact that it is *only* read by the program on first loading. Changes should only ever be made when the program is not running.

Note that WideServer does *not* generate an INI file with all these parameters included. If they are omitted, as most of them will be most of the time, then they assume their default meaning, as documented here.

The [Config] section

This section of the INI file is where the physical configuration and other rather technical parameters are placed.

Port=8002: This controls the IPX/SPX port number to be addressed, where this protocol is being used. Only clients using the same Port number are served by this server. You can have several WideServers running on a network, each using a different Port number. However, you can of course only have one copy of WideClient in each machine, as its job is to 'pretend' to be an FS6IPC.DLL-equipped copy of FS98 (or FS95 or FS2000) so that FS6IPC-aware applications will use it.

The default of 8002 is chosen only to avoid the two ports (8000 and 8001) used by WidevieW.

UseTCPIP=Yes: Set this to No if you would prefer to use IPX/SPX instead of TCP/IP. See the discussion on these earlier in the document. You need to have it set the same in the WideClient.ini files, and, if you use TCP/IP, either the server PC's **ServerName** or **ServerIPAddr** (see the WideClient INI file options section, below).

AutoUpdateTime=13: This sets the *minimum* time between AutoUpdates, in milliseconds. The default value of 13 will stop the individual data frame rate for any client exceeding 75. Lower values than 13 can be set, but there is a limit based on the way the DLL works, and also the capacity of the LAN. 100Mbit LANs may cope better with lower times. But watch out for FS frame rates: there needs to be some time left over for the simulation—and of course the application at the Client end needs some time too!

RestartTime=10: This feature allows you to make WideServer automatically close down its network serving action and restart it (just as if it had been freshly loaded) when there have been no connections for at least the specified number of seconds.

If you have problems on your network starting up the WideFS clients then this may help unclog Windows' sockets software (?). You may also find it useful to use the 'Restart Hot Key' facility, described below.

To disable this feature set the RestartTime to 0.

MaximumBlock=4096: This parameter controls the block sizes used by WideServer when sending altered data out to client applications. It shouldn't be set less than 512, nor larger than 16384. The default of 4096 seems to suit most applications and the higher speed (100 mbps) Networks, and the effect of the limitation is usually all over after the first few seconds of an application starting. After that only changed values are re-sent so the limit is seldom reached, except possibly with TCAS applications and high numbers of moving AI aircraft in FS2002. If you are using a slow Network connection (10 mpbs, USB, Serial or Parallel) then you may find 2048 better.

The [User] Section

RestartHotKey=: This option allows you to define a hot key which you can use in Flight Simulator to force WideServer automatically close down its network serving action and restart it (just as if it had been freshly loaded). If you have problems on your LAN with WideFS clients apparently stalling then this may help unclog Windows' sockets software (?).

The format for specifying the hot Key is 'keycode,shift states', for example:

RestartHotKey=78,11

specifies Shift+Ctrl+N (N for "Network"). The first value determines the main key required. This is a Windows 'virtual keycode'. A list of these is given in each of my FS "Controls" packages (there are separate ones for FS98, FS2000 and FS2002, but these codes are the same for each). These packages are available separately.

The second value determines additional shift states needed, as follows

omitted or 8	key on its own
9	Shift +
10	Control +
11	Shift + Control +
12	Alt +
13	Shift + Alt +
14	Control + Alt +
15	Shift + Control + Alt +

Note, however, that not all combinations will work with all keycodes. The values can also be 0–7 instead of 8–15. The addition of '8' here is merely to provide compatibility with the way key presses are specified in Fligfht Simulator's CFG files. And especially note that the use of the 'Alt' key in many combinations is problematic and will tend to invoke FS's menus. Avoid this key if at all possible.

AllowShutdown=No: Set this to Yes *only* if you want to allow client programs to shut down your Flight Simulator computer by writing a special value to the IPC interface. (Details for programmers are in the FSUIPC SDK -- look for offset 3320).

AllowShutdown=App can be used instead if you just want WideServer to close FS down, leaving the PC itself up and running. Of course any applications featured in WideServer's "Close" parameters (below), will also close.

ShutdownHotKey=: This option allows you to define a hot key which you can use in Flight Simulator to get WideServer to actually write the special "shut down" value to the IPC interface. This will be obeyed by all those WideClients with appropriate **AllowShutdown** parameter settings, and then also by WideServer itself if so configured. Using this you can close FS down and have the Client PCs also close at the same time (or a bit earlier, in fact, to ensure they see the WideFS message).

The format for specifying the hot Key is the same as above. I use Shift+Ctrl+E, so the setting is:

ShutdownHotKey=69,11

Log=Errors+: This is a debugging aid which I may ask you to change if I need more information to help sort out a problem with application interfacing. The default will create a WideServer.log file in the FS modules folder which contains details of basic errors but otherwise just some starting and stopping information and (most important for me!) the version number.

Set to 'Yes', this parameter logs all simulator variable read and write calls (and the results) made through WideServer. This is really only of any use to application developers, so they can see how their program is operating once its requests have been through WideClient.

Other settings are 'No' (not recommended as you lose error data), 'Errors' (only showing error reports, no other useful stuff) and 'Debug' (which gives details of WideServer's dealing with Windows' Sockets too).

If you do switch on any type full logging (e.g. 'Yes' or 'Debug'), be sure to keep the session short. The log file gets *very* large! The 'Debug' setting is only used on request for possible help in resolving complex problems.

Monitor=: This can be used by developers to find out what is happening to any one particular area of the FSUIPC variables memory area—i.e. the "offsets" being written or read by WideFS applications. It is better than using "Log=Yes" just to sort out a known variable. The format is:

Monitor=<offset>,<size>

where the offset is in hexadecimal and the size is a number of bytes in decimal. If you only want to watch one byte, you can omit the ",<size>" part. For example

Monitor=028C,1

makes WideServer log all network reads from and writes to this location, which happens to be the Landing Lights switch.

For a full picture you should use the same parameter at the client end too—i.e. in the WideClient.ini file(s).

IgnoreSumcheck=No: Setting this to 'Yes' is a *last* resort in trying to get some server-client interaction on a system where the initially larger client request blocks are consistently failing with sumcheck errors. The proper solution is to fix the protocol on the LAN. (This facility dates back to Windows 95 days. There is some evidence suggesting that the IPX/SPX protocol is not sufficiently reliable on original Windows 95 releases to provide good blocks. Users are really advised to try to upgrade to Windows 98SE, or at least Windows 95 OSR2.1).

ShowCounts=No: Set to 'Yes' to show the total network read and write counts (in bytes) in the Flight Simulator title bar.

TitleBarUpdate=Yes: Set this to 'No' if you don't want any sign that WideServer is running. Sometimes other programs and add-ons can be affected by the changes that WideServer makes to the Flight Simulator title bar. With this set to 'No' these add-ons may run correctly.

Run1=, **Run2**=, **Run3**=: These merely tell the program to run specific programs (specified with full pathnames) once the interface is ready. For example:

Run1=d:\RealWx\rwx5.exe

Might be used to run the 'Real Weather' program.

If the program needs command-line parameters, these can be included by enclosing the whole value in quotes so that the space(s) needed don't cause problems. For example:

Run2="c:\epic\loadepic fs98jet"

These three Run commands can also have an extra parameter to make the program being run do so at HIGH priority (higher than FS), and optionally Hidden (i.e. running invisibly). The latter does not work with all programs, however. ONLY use these facilities with programs which are run and completed before flying commences—loading control programs into an EPIC card is a good example. The extra parameter is HIGH or HIDE, respectively, as in:

Run2=HIGH,"c:\epic\loadepic fs98jet"

Note that HIDE implies HIGH as well. It works with Loadepic. Running it at high priority improves the chances of a successful load somewhat—by the time Flight Simulator is running there can be more clashes at the Epic direct interface.

Be careful using these facilities with FS2002, especially for programs which are running whilst FS2002 is also running. Some stability problems have been reported when trying to play with priorities.

Close1=Yes¸ **Close2=Yes**, **Close3=Yes**: Add these to ask WideClient to close the programs it loaded by **Run** when Flight Simulator itself closes. This is performed by sending the program's Windows a WM_CLOSE message, so if it ignores these, or has no Windows defined, this won't work.

RunIf1=, **RunIf2**=, **RunIf3**=: These are similar to the Run commands above, but need the Windows CLASS name associated with the program being run, and they only run the program if it is not already running. This is useful for those programs which do not, themselves, detect this, and avoids having to close them down when closing Flight Simulator, when you know you are going to start it again soon.

Example: RunIf1=ThunderRT5Form, "d:\FlightDirector98\fd98.exe W"

The CLASS name is the part before the comma (,), with the usual full path and command to be executed after the comma.

So far the only useful CLASS names I've found for use here are for:

Flight Director 98 (ThunderRT5Form), as above, and
Real Weather 5 (also ThunderRT5Form!).

Note that the order of executing the six Run commands is Run1, RunIf1, Run2, RunIf2 ... etc.

CloseIf1=Yes¸ **CloseIf2=Yes**, **CloseIf3=Yes**: Add these to ask WideServer to close the programs it loaded by **RunIf** when Flight Simulator itself closes. This is performed by sending the program's Windows a WM_CLOSE message, so if it ignores these, or has no Windows defined, this won't work.

Close=: This parameter can list up to three Windows CLASS names, representing programs which should be closed when WideServer itself closes (i.e. on Flight Simulator shutdown). Separate each of the Class names with commas.

Action1=, **Action2**=, etc: These tell the program to execute the given command line on receiving the corresponding 'Action' message from a WideClient. See the WideClient ini file notes for how to set this up.

These commands can have an extra parameter to make the program being run do so at HIGH priority (higher than Flight Simulator), and optionally Hidden (i.e. running invisibly). The latter does not work with all programs, however. ONLY use these facilities with programs that are run and completed quickly—loading control programs into an EPIC card is a good example. The extra parameter is "HIGH" or "HIDE", respectively, as in:

Action1=HIDE,"c:\epic\loadepic fs98prop"

Note that HIDE implies HIGH as well. It works with Loadepic. Running it at high priority improves the chances of a successful load somewhat—when Flight Simulator is running there can be more clashes at the Epic direct interface.

KeySend1–**KeySend255**: These facilities are primarily for cockpit builders, using multiple joystick button facilities like those with EPIC. They are available for use with ***any*** client utility which will accept keyboard commands. You must specify the keyboard actions required in the appropriate WideClient.ini file (see WideClient INI section below), but you specify the joystick button number to be used to ***trigger*** that keyboard action here, in the WideServer.ini file.

When you set **ActionKeys=Yes** in any WideClient.ini file, then the relevant WideClient can receive requests from WideServer. The requests are pre-defined in the WideServer.ini file by relating a specific "KeySend" number (1–255) to a Windows joystick ***button*** number (0-511). You then define what that KeySend number means in the Client by settings in the WideClient.ini file. Each client PC can use different or the same KeySend values, it is up to you.

Note that in a registered installation of FSUIPC you can define which Joystick Buttons send which KeySends on-line, in the FSUIPC options page called 'Buttons'. In fact you can also program key presses to send KeySends if you wish. The KeySend facility is listed in the drop-down lists of "FS controls" in both 'Buttons' and 'Keys' pages. Full information is provided in the FSUIPC documentation—just remember that the KeySend number (e.g. the "1" in KeySend1) is the ***parameter*** for the KeySend. FSUIPC does ***not*** list all 255 possible KeySend controls separately!

For those of you still wishing to set up your KeySend buttons the old way, I'll continue …

For this purpose joystick buttons are numbered from 0 continuously through the connected joysticks (counting the first joystick as 0), allowing for up to the maximum of 32 buttons on each, as follows:

Joystick 0: buttons 0–31
Joystick 1, buttons 32–63
Joystick 2, buttons 64–95
Joystick 3, buttons 96–127
Joystick 4, buttons 128–159
Joystick 5, buttons 160–191
Joystick 6, buttons 192–223
Joystick 7, buttons 224–255
Joystick 8, buttons 256–287
Joystick 9, buttons 288–319
Joystick 10, buttons 320–351
Joystick 11, buttons 352–383
Joystick 12, buttons 384–415
Joystick 13, buttons 416–447
Joystick 14, buttons 448–479
Joystick 15, buttons 480–511

Note that Game Controllers counts buttons as well as joysticks from 1, not 0, but Flight Simulator counts from 0 (in the CFG file). [*FS2002 'counts' joysticks using weird GUID numbers, so take care*].

Of the KeySend numbers 1–255, those numbered 61–126 are normally specifically allocated to functions within Chris Brett's EFIS98 package—see the section below for the allocations. However, if you are not using EFIS98 you can use these just like any of the others.

The format is simply this, for example:

```
KeySend1=510        ; Joystick button 510 off to on event
KeySend2=481,1      ; Joystick button 480 on to off event
...
KeySend255=475,2    ; Joystick button 475 on to off and off to on events
```

Note for EPIC users

If you are using EPIC with my EPIC95 package, then in the Epic EPL code, control the buttons in the usual way, with

Enque16(BtnOn,<button number>) and
Enque16(BtnOff,<button number)

Do NOT use "BtnPulse" or "BtnRepeat" as the changes may then not be seen. If a Pulse is required, use BtnOn and BtnOff with a Delay(20) or similar between. The exact delay you need will have to be determined by experiment. Better, if you are using EPIC.VXD version 5.11 or later, you can use "BtnToggle" instead (a new facility in version 5.11).

With this facility you can set the KeySend parameters with the ",2" option and merely toggle the but on or off when the KeySend action is to be signalled. You can, of course, simulate the BtnToggle action by using an EPL flag to remember whether the button is on or off, and using BtnOff or BtnOn accordingly. The BtnToggle saves this hassle however.

KeySend61–KeySend126: Special allocations for use with the Papa Tango EFIS98 package running on the client

The format of the KeySend61–KeySend126 parameters when used to drive EFIS98 is identical to the others, so refer to the above for details. The difference only lies in their FIXED functional allocations in EFIS98, as follows:

KeySend61	Keyboard mode off
KeySend62	Keyboard mode on
KeySend63	ND Range 10
KeySend64	ND Range 20
KeySend65	ND Range 40
KeySend66	ND Range 80
KeySend67	ND Range 160
KeySend68	ND Range 320
KeySend69	ND Rose ILS
KeySend70	ND Rose VOR
KeySend71	ND Rose NAV
KeySend72	ND Arc NAV
KeySend73	ND Plan NAV
KeySend74	ND toggle VOR overlay ON/OFF
KeySend75	ND VOR overlay OFF
KeySend76	ND VOR overlay ON
KeySend77	ND toggle NDB overlay ON/OFF
KeySend78	ND NDB overlay OFF
KeySend79	ND NDB overlay ON
KeySend80	ND toggle WPT overlay ON/OFF
KeySend81	ND WPT overlay OFF
KeySend82	ND WPT overlay ON
KeySend83	ND toggle APT overlay ON/OFF
KeySend84	ND APT overlay OFF
KeySend85	ND APT overlay ON
KeySend86	ND toggle CNSTR overlay ON/OFF
KeySend87	ND CNSTR overlay OFF
KeySend88	ND CNSTR overlay ON
KeySend89	pointer1 ADF1
KeySend90	pointer1 OFF
KeySend91	pointer1 VOR1
KeySend92	pointer2 ADF2

KeySend93	pointer2 OFF
KeySend94	pointer2 VOR2
KeySend95	FMGC display off
KeySend96	FMGC display on
KeySend97	ND display off
KeySend98	ND display on
KeySend99	FCU display off
KeySend100	FCU display on
KeySend101	SPD under EFIS FCU control
KeySend102	SPD under FS control
KeySend103	HDG under EFIS FCU control
KeySend104	HDG under FS control
KeySend105	ALT under EFIS FCU control
KeySend106	ALT under FS control
KeySend107	FD off
KeySend108	FD on
KeySend109	SPD large decrement
KeySend110	SPD small decrement
KeySend111	SPD small increment
KeySend112	SPD large increment
KeySend113	HDG large decrement
KeySend114	HDG small decrement
KeySend115	HDG small increment
KeySend116	HDG large increment
KeySend117	ALT large decrement
KeySend118	ALT small decrement
KeySend119	ALT small increment
KeySend120	ALT large increment
KeySend121	FCU SPD selected
KeySend122	FCU SPD managed
KeySend123	FCU HDG selected
KeySend124	FCU HDG managed
KeySend125	FCU ALT selected
KeySend126	FCU ALT managed

Note that if ANY of the KeySend60–126 values are specified with keyboard codes in the Client PC, then this latter takes precedence over the EFIS98 action in that PC.

WideClient INI file options

Here all the user features of the client's control parameter file are described. Normally you won't need or want to mess around with many of these parameters. There are all placed into the 'WideClient.ini' file, which goes into the same folder as the Wideclient.exe program copy it is to be used with. The filetype 'ini' merely refers to the file as an Initialisation file—reflecting the fact that it is *only* read by the program on first loading. Changes should only ever be made when the program is not running.

The [Config] section

This section of the INI file is where the physical configuration and some rather technical parameters are placed.

UseTCPIP=Yes: Set this to No if you would prefer to use IPX/SPX instead of TCP/IP. Make sure the WideServer part is also set to use the same protocol. See the discussion on these protocols earlier in the document. If you omit this, or set it to 'Yes', you must also set the server PC's **ServerName** or **ServerIPAddr**, described below.

ServerName=: Either this or the **ServerIPAddr** must be provided if the TCP/IP protocol is chosen and set by the **UseTCPIP** parameter. The server name is the one you assigned in the Network properties: on Windows 98 and probably others it's the "computer name" in the Identification tab.

ServerIPAddr=: If you are using TCP/IP, you can elect to specify the Server by its IP Address instead of its name. If you give both, only the IP address will be used. To use this you need to have assigned a fixed IP Address for your Server PC, as described earlier (in the "Configure Your Network" section). The format is four decimal numbers separated by points, for example:

ServerIPAddr=192.168.0.3

ServerNode=: This is only used with the IPX/SPX protocol, and can be omitted for Windows 95, 98 and (probably) ME installations, but it is needed when Windows NT, Windows 2000 or Windows XP are being used, whether at the Server or Client end, or both. It may make initial connection more efficient on Windows 95/98/ME too.

The ServerNode to be set here is determined by the specific network adapter being used by the Server. You can get this most easily by running Flight Simulator with WideServer.dll installed, and then looking at the WideServer.log file (which you will find in the Modules folder, with the module itself). Near the beginning there should be a line containing

ServerNode=n.n.n.n.n

where the n's are decimal. Copy this part of the line into the [Config] sections in all the Wideclient.ini files you are using for programs which will connect to this specific server.

Be aware that there have been problems with some network drivers reporting the incorrect 'ServerNode' values to WideServer. See the trouble shooting section earlier.

Port=8002: This controls the IPX/SPX port number to be addressed. Only the WideServer running with the same Port number will be addressed by this WideClient. You can have several WideServers running on a network, each using a different Port number.

Note that you can only have one copy of WideClient in each machine, as its job is to "pretend" to be an FS6IPC.DLL-equipped copy of FS98 so that FS6IPC-aware applications will use it.

The default of 8002 is chosen only to avoid the two ports (8000 and 8001) used by WidevieW, in my earlier package "WideEFIS".

Window=43,44,886,589: Don't alter this unless you "lose" the Window! It stores the size and position you last used for the main window.

Visible=Yes: Normally leave this to default. Other values are:

No	if you don't want any sign of WideClient (in this case, to terminate use CTRL-ALT-DEL)
Min	to start up minimized
Max	to start up maximized

Note that there are not many client programs that actually need a Window. EFIS by Chris Brett is one. You should leave this parameter to default to 'Yes' for EFIS, then size the WideClient window to fit the EFIS plus FMC windows, arranged nicely side-by-side, with the control buttons beneath. WideClient will remember the window size and position. (Similar considerations apply to Moving Map. You can arrange all EFIS and Moving Map windows within WideClient's window, and they will all be remembered by their respective programs).

WaitForNewData=500: Network response with new data timeout, in milliseconds. This applies only to requests made for data which is not yet "registered" with the server, as being needed by this client. The client deliberately waits, examining messages coming back from the server, for up to this number of milliseconds, or until it sees the new data it requested arriving.

A large timeout here helps guarantee that the application will see good data right from the outset (assuming FS is already running and the Server is able to respond in this time), rather than be supplied with zeroes from WideClient's own memory. Once the client knows that the data requested is registered with the server it no longer waits but services the application from its memory and relies on updates for changes from the server.

This means that, with a larger timeout, the application may be slightly jerkier initially, but obtains good data. If the slower start is not acceptable, you can omit the timeout altogether by setting this parameter to 0.

Note that if the connection to the Server is restarted at any time, for any reason, then all the data it was previously receiving is again considered un-registered, so the new data timeout applies again.

If the **Timeout** value (see below) is larger than the **WaitForNewData** value, then it applies instead.

Timeout=12: Network response timeout, in millisecs. **Timeout** applies to every request made by your client applications. Since the Server is sending updates for requested data in any case, this timeout is only useful for limiting the processor time used by the application, to stop it hogging the client PC when there are other programs to run.

Many FS6IPC/FSUIPC client applications are reasonably well behaved, however, and do timeshare well enough, so this timeout can be set quite low, even down to 0. But be careful. With some applications, if there is not a delay before returning to the application from each of its data requests, then the loop can be too tight and there may be some real difficulties in accessing other facilities on that PC, moving and sizing windows, and so on.

NetworkTiming=5,1: This gives control over two quite critical periods. Both numbers are times in milliseconds. The first (defaulting to 5) specifies the minimum time to be given to the Application to allow it to read changes after WideClient has received each new frame. The second (defaulting to 1) specifies the minimum time to be given to the Network thread to allow new frames to be received and processed before acting upon each read or write request from the Application.

This is quite a balancing act and needs to be just right to achieve perfectly smooth operation. Ideally the Network thread should receive one frame which the Application can immediately process, and so on, but in practice frames are like buses, they run in bunches and sometimes not at all. These numbers allow experimentation with the way this is turned into apparent smoothness.

The [User] Section

Background=: WideClient will display a user-supplied bitmap in its window instead of the featureless grey. This allows suitable backgrounds for utilities such as EFIS, EFIS98 and Moving Map to be designed. Specify the BMP file to be used by adding this line into the [User] section of WideClient.ini:

```
Background=filename.bmp
```

The filename can contain the complete path to the file: useful if it isn't stored in the load path.

Run1=, **Run2**=, **Run3**=: These tell WideClient to run the specified programs (specified by their full pathnames), as WideClient is initialising. For example:

```
Run1=f:\efis\Efisv2.exe
```

This would load EFIS Version 2. Other suitable clients are RWX5.EXE (Real Weather 5 by Jeff Wheeler and Steve Halpern), and Aeroview.Exe (the moving map free on the CDROM with Nick Dargahi's book). These are the ones I use and have tested, but any FS6IPC.DLL using program should run fine. If the program needs command-line parameters, these can be included by enclosing the whole value in double quotation marks (") so that the spaces needed don't cause problems.

Delay1=¸ Delay2=, **Delay3=**: These optionally define delays, in seconds, to be executed after the corresponding **Run** parameter, above. Whilst the delay is operating WideClient is sleeping, so it will *not* attempt to make a connection during this time. The maximum delay which is set is 60 seconds.

Close1=Yes¸ Close2=Yes, **Close3=Yes**: Add these to ask WideClient to close the programs it loaded by **Run** when WideClient itself closes. This is performed by sending the program's Windows a WM_CLOSE message, so if it ignores these, or has no Windows defined, this won't work.

RunReady1=, **RunReady2**=, **RunReady3**=: These are identical to the **RUN** options above, except they are not actioned until WideClient is actually connected to WideServer.

DelayReady1=¸ DelayReady2=, **DelayReady3=**: These optionally define delays, in seconds, to be executed after the corresponding **RunReady** parameter, above. Whilst the delay is operating WideClient is actually still running. The maximum delay which is set is 60 seconds.

CloseReady1=Yes¸ CloseReady2=Yes, **CloseReady3=Yes**: Add these to ask WideClient to close the programs it loaded by **RunReady** when WideClient itself closes. This is performed by sending the program's Windows a WM_CLOSE message, so if it ignores these, or has no Windows defined, this won't work.

Close=: This parameter can list up to three Windows CLASS names, representing programs that should be closed when WideClient itself closes. Separate the Class names with commas.

For example, EFIS98's CLASS name is TToolWindow97, so you can set WideClient to close EFIS98 automatically when you close WideClient by the parameter setting:

```
Close=TToolWindow97
```

To obtain Class names is not easy without programmer's tools, but the program's author can tell you. Class names for some of the possible Client applications are provided below, in the section on the **KeySend** feature. Project Magenta class names can be configured by parameters in the individual module ini files.

The **Close** parameter provides no facility to distinguish between two or more programs which use the same Windows Class name.

Actions=: This tells the program to create a Menu with the listed commands. Each command, when selected, sends a special message to WideServer. The first command executes the **Action1** line in WideServer's ini file, and so on. Use commas to separate commands. For example:

```
Actions=Jet,Prop,Heli
```

These three menu entries might be used in conjunction with these WideServer.ini parameters:

```
Action1="C:\EPIC\LOADEPIC FS98JET"
Action2="C:\EPIC\LOADEPIC FS98PROP"
Action3="C:\EPIC\LOADEPIC FS98HELI"
```

thus getting the server to re-load different Epic control programs on request from the client.

Log=Errors+: This is a debugging aid which I may ask you to change if I need more information to help sort out a problem with application interfacing. The default will create a WideClient.log file in the FS modules folder which contains details of basic errors but otherwise just some starting and stopping information and (most important for me!) the version number.

Set to 'Yes', this parameter provides a log of all simulator variable read and write calls (and the results) made through WideClient. This is really only of any use to application developers, so they can see how their program is operating once its requests have been through WideClient.

Other settings are:

No	not recommended as you lose error data
Errors	only showing error reports, no other useful stuff

and a range of really heavy logging options (Debug, Debug+, DebugAll, All, AllRx, PartRx) which will tend to only ever be used under instruction when trying to nail really obstinate problems.

If you do switch on any type full logging (e.g. 'Yes' or any of the "heavy" settings), be sure to keep the session short. The log file gets *very* large!

You can also have the full data logging switched by Hot Key: e.g.

```
Log=K1190
```

The number after the K here represents Shift >. The value is the Virtual Key number (see one of my FS Controls packages for a list) plus 1000 for Shift and/or 2000 for Ctrl. The logging state is then shown in the WideClient title bar.

Monitor=: Developers can use this to find out what is happening to any one particular area of the FSUIPC memory area—i.e. the offsets being written or read by the WideFS applications on this client. The format is:

```
Monitor=<offset>,<size>
```

where the offset is in hexadecimal and the size is a number of bytes in decimal. If you only want to watch one byte, you can omit the ",<size>" part.

For example

```
Monitor=028C,1
```

This makes WideClient log all Network reads from and writes to this location, the Landing Lights switch.

For a full picture you should use the same parameter at the server end too—i.e. in the WideServer.ini file.

AllowShutdown=No: Set this to Yes only if you want to allow client programs (or the server's **ShutdownHotKey**) to shut down this client PC by writing a special value to the IPC interface. (Details for programmers are in the FSUIPC SDK—look for offset 3320).

AllowShutdown=App can be used instead if you just want WideClient itself to close down, leaving the PC itself up and running. Of course any applications featured in WideClient's "Close" parameters (above), will also close.

ShowRxFrameRate=No: Set this to Yes to show the frame rate—i.e. the number of frames per second received from the Server on this client. In general this will be very low (0 is less than 1, not absolute zero! <G>) when FS is not doing much, and, hopefully, something near to FS's frame rate when things are changing a lot. But this will depend upon the Applications. If there's only one Application and it is only reading the Time of Day (to the nearest second) then this will only change at most once a second, so the frame rate will hover around 0 or 1 no matter what is going on in FS. Performance summaries are shown at the end of the Log.

ShowCounts=No: Set to Yes to show the total network Read and Write counts (in bytes) in the WideClient title bar. In general it is more useful to use the reception frame count instead.

EFISkbfocus=No: [*Only suitable for use with the EFIS98 package on the Client*]

Set this to Yes if you are using Chris Brett's EFIS98 package on this Client, and wish to make the keyboard input mode in EFIS98 operate automatically, whenever the Client window has focus. Make sure EFIS98 is docked: do this after sizing the windows to suit. You can position them afterwards.

This option saves having to remember to press Shift+Space before selecting options by keyboard short cuts, or entering details into the FMGC or the Options dialogue. WideClient will do this for you, and will disable keyboard input whenever the Client window loses focus (e.g. to access some other program).

ActionKeys=No: Set this to Yes to allow this Client to receive KeySend requests from the Server. The server maintains a queue of up to 16 KeySend values, and any client can read them and process them. The order of the original joystick button presses is maintained, and each Client can do what it likes with what it obtains. You can program one KeySend event number (1–255) to do different things on different clients, or set each client to only respond to different KeySend events (probably more useful).

KeySend: If you set **ActionKeys**=Yes (see above), then this Client can receive requests from the Server. The requests are defined in the WideServer's INI file by relating a specific KeySend number (1–255) to a Windows joystick button number (0–511). Alternatively, when using any FSUIPC version *later* than 2.932 you can use its **Buttons** and/or **Keys** option pages to program KeySends as if they were special "FS controls", with the KeySend reference (1–255) as parameter. This allows use of both buttons (including PFC.DLL connections) and server key presses to be relayed as KeySend messages.

Of the KeySend numbers 1–255, those numbered 61–126 are normally specifically allocated to functions within Chris Brett's EFIS98 package. For details of these please see the earlier section on the WideServer INI parameters. When these are used, the action in WideClient can be automatic (provided **ActionKeys**=Yes is set), so there is then no need to enter KeySend61–KeySend126 parameters into the WideClient.ini file. But you can override the EFIS98 defaults by using these parameters if you like, whether EFIS98 is in use or not.

Subject to this specific use with EFIS98, all 255 KeySend parameters are available for use with *any* client utility that will accept keyboard commands. Specify the keyboard actions required as shown in the following examples:

```
KeySend1=65,9
        ; Shift+A
KeySend2=8,11
        ; Shift+Ctrl+Backspace
KeySend255=112,12
        ; Alt+F1
```

Here the first value determines the main key required. This is a Windows "virtual keycode". A list of these is given in my FS Controls documents, available separately. The second value determines additional shift states needed, as follows

8	key on its own
9	Shift +
10	Control +
11	Shift + Control +
12	Alt +
13	Shift + Alt +
14	Control + Alt +
15	Shift + Control + Alt +

Note that not all combinations will work with all keycodes. The values can also be 0–7 instead of 8–15. The addition of 8 here is merely to provide compatibility with the way key presses are specified in Flight Simulator's CFG files.

Use of the Alt key in many combinations is problematic. Try to avoid this key if possible.

Note that all these values operate the keystroke momentarily: i.e. they do a "key down" followed by a "key up". If you want one KeySend event to press a key and a separate KeySend event to release it, then you will need to alter the shift state above as follows:

Shift state + 8 to Press the key
Shift state + 16 to Release the key

For example

```
KeySend1=65,17   ; Shift+A press
KeySend2=65,25   ; Shift+A release
```

These parameters make KeySend 1 and 2 press the Shift+A combination for as long as the action in the server need it to. You would probably program the KeySend entries in WideServer.ini to operate KeySend1 when a button is pressed and KeySend2 when it is released.

Take care when using these more advanced features not to get your client PC in a bit of a mess, with assorted key states stuck on. To release stuck keys, press them on the client's keyboard—the Key UP codes should sort things out.

Directing Key Strokes more precisely

If the application that is to receive the keystroke is not a child window of Flight Simulator (i.e. of WideClient, which is substituting for FS in this case), the Windows keyboard focus may not allow the assigned keystroke to reach it. In this case you need to add another parameter, or maybe two, to each KeySend line, identifying the program to receive it.

If the program is one you are having WideClient load, using the **Run** or **RunReady** parameters, then the additional parameter can simply be the name of the parameter you used. For example:

KeySend1=65,9,Run1

and KeySend1=66,9,RunReady2

However, if it is a program being run separately then you will need more information about that program, in particular the program's main Window class name, and where there's a chance of confusion, the title for the application window concerned. The next section goes into this in some detail.

CLASS Names

Windows class names either have to be supplied by the author of the program, or obtained by other programs such as the Spy programs that come with development packages like Microsoft's Visual C++. Here are the Class names for some (now rather old) FS utility applications. When there are more than two programs running with the same Class name, the title (from the title bar) is needed as well, as an extra parameter.

FlightDirector98	ThunderRT5Form
Project Magenta:	ThunderRT5Form,"PFD GLASS COCKPIT"
	(but PM modules now have programmable class names—see PM INI files)
Real Weather 5	ThunderRT5Form
Aeroview	TestClass

For example:

```
KeySend1=65,9,ThunderRT5Form,"PFD GLASS COCKPIT"
                        ; Shift+A
KeySend2=8,11,ThunderRT5Form,"PFD GLASS COCKPIT"
                        ; Shift+Ctrl+Backspace
KeySend255=112,12,ThunderRT5Form,"PFD GLASS COCKPIT"
                         ; Alt+F1
```

Note that quotes " " are needed around the Window title when given. They are also needed around the Class name if it contains any spaces.

Roger Wilco

There are two special forms of the KeySend parameter which are specifically designed to operate Roger Wilco's Push To Talk (PTT) action:

KeySend<n>=RWon
KeySend<n>=RWoff

Just select appropriate KeySend numbers instead of <n>, and define matching KeySends in the WideServer.ini file to operate these on your chosen PTT joystick button. So, if you have defined:

KeySend1=Rwon

KeySend2=Rwoff

you define the buttons in the Server INI file using the same parameter names, KeySend1 and KeySend2 in this case.

Note that, probably easier, and as already mentioned earlier, with recent versions of FSUIPC you can actually define KeySends in the FSUIPC Buttons or page to do the same thing—or assign keypresses in FSUIPC's Keys page. If you do it this way, look in the FS controls drop-down list for "KeySend" and then set the parameter for the KeySend control to the KeySend number. In this

case, for Roger Wilco, you would assign the button or key *press* the KeySend control and parameter = 1 (to do Rwon), and the key button or key *release* the KeySend control and parameter = 2 (to do Rwoff). The numbers tie up the KeySend controls to the matching parameters in the WideClient.ini file.

Don't forget you need the **ActionKeys=Yes** parameter too, otherwise WideClient won't be looking out for any KeySends.

This works well with Roger Wilco Mark 1 and Mark 1c. It should be okay with other versions, but it hasn't been tested with them.

PostKeys=No: Normally all **KeySend** operations are performed using keyboard playback facilities in Windows. These are generally more reliable and have the advantage of reproducing exactly the same sequence of keyboard-related messages that would occur if the real keys were being pressed. However, they do seem to have a problem with keyboard focus, and, whilst WideClient does attempt to change focus to the target window if it doesn't already have it, this can sometimes cause problems, resulting in missed or ignored keypresses.

If you have the target window class name, as described above, or you know that the application docks itself as a child of FS (and so WideClient), you can tell WideClient to **Post** the key presses instead of playing them back like a recording. This needs no focus changing, and works, *provided* that you get the Window class name right, that it is unique, and that the target program is happily processing WM_KEYDOWN or KEYUP messages and doesn't care about WM_CHAR messages or the timing relationships between all these.

To post key presses just set **PostKeys**=Yes. This works well with Project Magenta's PFD displays, and probably also the MCP.

SendKeyPresses=No: [Only for use when running FS2000, FS2002 or CFS2 under Windows 98/ME/2000/XP on the Server PC]

If you set this parameter to Yes, then any non-system key presses (i.e. anything not including the ALT key) received by WideClient will be relayed to FSUIPC and thence to FS/CFS2. This has limited uses these days.

History of Recent Changes

Version 6.00 works on FS2004 as well as earlier versions of FS, and needs at least version 3 of FSUIPC. It also will not run unless it is Registered with FSUIPC, using a purchasable access key.

In addition to the changes explicitly for FS2004, WideFS is improved so that you don't get continuous re-connections from Clients when FS is in menu dialogues for long periods, and it also doesn't reset any data to zero when a long break occurs. However, data is not updated during such times. It does now maintain contact *and* data updates when FS is minimised, albeit at a much lower frame rate (something like 2–4 fps).

Version 5.50 is the first version that defaults to using TCP/IP instead of IPX/SPX. Apart from some other cosmetic changes, the only other important change is the fixing of the Shut Down facility on Windows 98/Me, so that it correctly powers off the PC if possible.

Version 5.41 consolidates the performance improvements incorporated into 5.40 with more minor adjustments to suit more systems "out of the box", and also introduces better error recovery and guaranteed ordering of frames—previous versions, during error retries, could get frames out of strict sequence. The numbers of retries are also reduced by better scheduling and not even retrying for Server transmissions of data changes, just leaving them to be seen again next time round. In addition, the excessive logging of missed sequence numbers is avoided in cases where the server knows that the frame couldn't be sent, by backtracking on the frame numbering system.

Version 5.40 is mostly about better performance, just as 5.30 was, but now the emphasis has been moved to ***smoothness*** with demanding applications like Project Magenta. The main changes apart from this are:

- A **ShutdownHotKey** is provided for WideServer, to allow FS and all clients to be closed with one keypress combination.
- A bug preventing smaller values for the MaximumBlock parameter is fixed.
- Areas in WideServer which are accessing parts of FS which may not be present on some occasions are now protected from crashing FS, and instead merely result in an error report in the log.
- WideServer's operation is now tied to the FS frame rate, not a regular "tick" (18/sec). This means it will sometimes be faster, sometimes slower than it used to be, but generally it should keep pace with FS's own changes. This should result in greater smoothness.
- TCP/IP operation is made *much* smoother by switching off Windows' default frame coalescing action—a method Windows uses to reduce traffic by not sending frames for a while in case another comes along to send. This "clumping" of frames was a major factor in making TCP/IP look worse than IPX/SPX.
- The **AllowShutdown=Yes** option now does allow a complete shutdown, powering off if possible instead of merely leading to the Windows message telling you that you can do this manually.

Version 5.30 is all about ***performance***. There are really no new facilities as such, but better speed and smoothness. It has been improved so much that using TCP/IP is now pretty good. Still not as fast as IPX/SPX, but fast enough.

Version 5.20 fixes a problem with the RunReady facility, which previously could load multiple copies of the same program, adds options to Close WideFS-loaded programs without having to know their Window class names, and provides a more precise way to direct KeySend key strokes for programs loaded by WideClient.

Version 5.10 includes a few minor improvements, and adds the RunReady, Delay and DelayReady facilities to WideClient.

Version 5.01 fixes a silly problem in WideServer where the title bar message "not able to serve (see log)" is not replaced by the "waiting for clients" message upon completion of the Server Network initialisation. This bug did not affect the way WideFS worked, but was very misleading.

Version 5.00: Main release, first big change for years:

- TCP/IP protocol is supported as well as IPX/SPX, though the latter is still defaulted for performance reasons.
- Documentation has been completely overhauled in both content and presentation. Provided in DOC and PDF format.
- Failure of WideServer to start the service does not now stop FS from running after 'Ok' is pressed to the error message box. The lack of service is shown in the title bar, and the Restart Hot Key (if configured) retries the service start up sequence from scratch, reporting the problem again if it recurs.

Version 4.701: Full Beta test version with working TCP/IP option, exposed for several weeks on a number of sites with great success. Led to general release with only minor changes (as Version 5.00).

Version 4.70: First version with TCP/IP support, but buggy. Released for Beta testing but replaced within hours by 4.701.

Version 4.65: Some minor improvements concerned with safety in case of LAN data corruption and small performances boosts.

Version 4.61: Dependence on Windows' WM_TIMER messages removed, to try to clear up performance problems on Win2000 and WinXP.

Version 4.60: Full support for FS2002 included, using FSUIPC 2.73 or later.

Version 4.58: Made more efficient by the addition of simple RLE compression of the transmitted FS data (server to client direction only). This mainly helps programs like FSInterrogate, which tend to read large areas much of which may be all zero.

Recovery is also improved on some other LAN conditions.

WideServer now uses a set of fixed internal buffers for the reception and transmission of LAN packets, rather than place a load on the FS heap space allocation system.

WideClient doesn't keep re-requesting the same data just because it hasn't got a response for it yet. That action tended to clog up the system in the early stages and make things take longer to start up. It now holds off re-requesting data for a few seconds each time.

Version 4.561: Fix to an error in the 4.56 changes that would leave clients un-refreshed for changes made by applications in the same client. This applied to locations below the 0x4000 offset level.

Version 4.56: Made packet transmission and selection more secure: trapped additional causes of lost packets and implemented corrective retries, and added additional data scans to catch cases where more than one PC in the Network have clients toggling the same flags. These changes are especially important when using recent Project Magenta builds.

Added logging facility to allow simple monitoring of any one specific area of the FS/FSUIPC variables/communications memory (see the Monitor parameters in the WideServer/WideClient text files included).

Version 4.50: Improved some of the response times, especially on faster Networks, by removing net traffic limiting code altogether.

Supports special facility to operate Roger Wilco's PTT from the FS PC. This is currently only usable with PFC buttons (via assignments in the PFC driver), but will be extended to FSUIPC soon.

Provides a Shut Down facility for Client programs (actually ANY program interfacing to FSUIPC version 2.50 or later). This is enabled in WideServer.ini or WideClient.ini by the "AllowShutdown=Yes" parameter.

Improves SendKeyPresses operation in cases where the key pressing is being done by another program (therefore faster) rather than by the user pressing real keys.

Version 4.40: WideServer's "AutoRestart" is now defaulted to restarting the server actions every 10 seconds whilst there are no connections. These restarts are no longer logged. This change makes it less critical which of the Server and Client are started first.

WideClient now tries to provide correct data on its first request, by automatically invoking a longer timeout for a response from the Server when new data is requested. This timeout defaults to 500 mSecs and is set by the "WaitForNewData" Configuration parameter. Previous operation is obtained by setting this to 0 if required.

Extended the KeySend facilities in WideServer to work with buttons from any joystick configuration, not only EPIC card connected buttons.

Extended the KeySend facilities in WideClient so that keyboard button events for Key Down and Key Up can be triggered separately.

Provided a facility in WideClient for posting keyboard events directly to known program Window class-names rather than rely on playback facilities that seem to still need focus changes.

Added a special facility to WideClient to trigger the transmit mode in Roger Wilco on specific KeySend events.

Published by Peter L. Dowson, 25th July 2003