WideFS History

Version 6.94 WideClient.EXE and WideServer.DLL (February 2012)

- Code signature updated, now expiring January 2015.
- [FSUIPC4 Server only] Added "BroadcastMode" support—see the Technical guide.

Version 6.86 WideClient only (April 2011)

- Added facilities to change assorted sound devices defaults before loading each program via the Run, RunReady and RunKey parameters, thus allowing programs to use different sound devices even though they may not provide sound device selection facilities themselves.
- Added a KeySend facility allowing programs started by WideClient to be brought to the foreground.
- Made WideClient pre-process FSUIPC 1070 (keypress and release controls) being sent to offset 3110, and update its local Hoy Key tables if necessary, anticipating the keys being actioned in that way in FSUIPC. This makes for more efficient co-operation between programs running on the same Client PC, one issuing keystrokes to control or respond to the other.
- When used with FSUIPC4 version 4.585 or later, the FSUIPC4 offset monitoring facility on the right-hand side of the Logging tab will now not only log changed to the offset, but also, for changes instigated by WideFS clients, log the PC name and the client program ID. The program responsible for writing to that offset can be identified from its ID in the client PC's WideClient.Log file.
- A "Deny" parameter can now be added to the [User] section of the WideClient.INI file to prevent applications on that client PC (only) from changing specific offsets or ranges of offsets. The facility is described in the updated Technical guide included.
- Made WideClient's operations more efficient and more reliable when run with several client applications. Previous versions may have occasionally given "bad address", frame or sumcheck errors when loaded up with several heavy user applications because of interlocks between threads This version has those interlocks better positioned.
- The Program Close facilities have been extended to cope with a variety of different ways applications can be designed to apparently defeat the simple sending of Windows "Close" messages or even the operation of the process termination ("KILL") method.

  The first and easiest option is to use the keyword "Last" instead of "Yes" or "Kill" as the value for the relevant Close parameter. Instead of seeking to close all the top level windows of the application in order, it merely closes the LAST visible one. Since Windows appears to enumerate an applications windows in reverse order of creation, this should usually find the first one.

  The second, which may be needed with applications which start off as one process (the named EXE in the "Run" line), but then create another process (i.e. run another EXE) of which, of course, WideClient is then unaware. If it is that new process which needs closing, WideClient needs to be told its name. You do this by adding the EXE name (or the whole pathname if there's likely to be any ambiguity), to the Close parameter, after the "Yes", "Kill" or "Last" part. For example:

  ```
  Close1=Kill,"Flightdeck_Companion.exe", or possibly
  Close1=Yes,"Flightdeck_Companion.exe"
  ```

  for FDC, which seems to be one such application. (Note: it has actually been verified that both of these methods do actually work on FDC). The outside quotation marks (") are optional, but certainly needed if you want to add any comments to the line.
- The new Lua sound facilities now work correctly on WinXP as well as Win7 and Vista. Before this the termination of the Lua program terminated the sound as well -- a quirk of the older DirectSound facilities.
- Additional Lua facilities are supported, including:

com.openhid	to use the com library facilities on USB HID devices
event.com	to allow for event driven device drivers instead of using loops doing com.reads.
event.sim	special sim events such as closing, flight saves and loads and aircraft changes
event.timer	simple wake up calls at specified intervals

  For full details please see the updated Lua documentation installed with FSUIPC versions 3.99 or 4.70, or later.

- WideClient automatically supports the FSUIPC4 facilities to obtain weather from Active Sky (ASE build 638 or later) in response to weather requests from applications. This applies when you run ASE on a separate PC from FSX. You will need to run WideClient.exe on the same PC as ASE. This is the case even if you otherwise have no real use for WideFS on that PC.

- Fixed an error in the Lua and VRI facilities whereby some input bytes on COM or USB devices could be lost each time the internal read buffer wrapped around (this occurs every 1024 bytes read).
- Lua program offset writing is prevented until a proper connection has been established for about 20 seconds. This is to avoid clashes with the intense start-up activity at this time. Offset reads by Lua plugins will get the local memory values during this time.
- Fixed an error which could potentially cause WideClient to hang in a "deadly embrace" when a Lua plug-in is using the event.offset function.
- Made the Lua timer event properly cancellable with "event.cancel", and also allowed a Lua program with only the timer event enabled to stay resident whilst that timer event is enabled. Previously you would have needed another type of event as well as the timer to get Wideclient to keep the Lua program loaded.
- Re-arranged the internal order of some of the major modules in order to try to stop some anti-virus programs mistakenly identifying part of the compressed and encrypted code as malware. Whether I've succeeded or not I do not know.
- The Lua event.offset() function now automatically executes the named function with the initial value as well as on every subsequent change. This saves having to explicitly call the function with an offset reading to get things initialised with their current start-up values.
- Added new Lua plug-in facilities for changing bits in offsets rather than complete bytes etc. These are via new Lua ipc library functions, ipc.togglebitsXX, ipc.setbitsXX, and ipc.clearbitsXX, where XX is UB, UW or UD for Byte, Word and Double Word offset values respectively. Details are available in the revised Lua plugins documentation.
- A bug in the Lua event.timer facilities made every Lua plug-in share the same timer. They now all correctly have their own timer.
- The function name provided as a string in the Lua event function calls can now be functions in tables. This enables functions in Modules, brought in by the "Require" function, to be used for event processing, because Modules so enabled provide tables of functions (and other values) for access in the current program. The format of the function reference string must be <table>.<function>, so if the Module is named (or equated to) "M", say, then function "fn" inside it would be referred to as "M.fn" in the event function. (The alternative form "M[fn]" is not allowed).

  The facility is actually extended to handle tables within tables, to no set limit other than the entire string name must be less than 64 characters (between the "").
- The facilities for running programs are enhanced by additional options which control how their Windows are displayed. This applies to programs run by Run, RunIf, RunReady and RunKey. There's a set of additional parameter types: Show, ShowIf, ShowReady and ShowKey, each relating to the similarly characterised Run program. For instance, Show3 refers to the Run3 program, whilst ShowKey4 refers to the RunKey4 program. The options are:

  Show<id>=<mode> or Show<id>=<delay>,<mode>

  where <mode> is one of MAX (for maximised), MIN (minimized), HIDE (hidden). Anything else does a 'restore' (to the window's default size and state).

  The <delay> part is a number of seconds to wait after the program is loaded before trying to change the Window. This is needed for many programs to make sure the correct top-level program window is affected, as many programs display a banner or initialisation window before the true operational one.
- A Lua error which can occur after an event.cancel function call for a Timer is fixed in this release.
- The offset 333C is now cleared to zero by WideClient when a disconnection is detected.
- Lua plug-ins, other than "initial.lua", are now not started by WideClient until all of its initialisations are complete and it is capable of supplying offset data on demand.
- Fixed an error which could cause WideClient to hang or crash if a local Lua plug-in is using the event.offset facilities when FS is restarted for any reason.

- The "Run ..." parameters can be given the full pathname of a Text file (.txt) instead of a program. WideClient will read the first line of that text file and run that to identify the program path and its parameters (if any). This can be used in conjunction with facilities added to Its Your Plane (IYP) in build 222 to allow WideClient to start and stop IYP by KeySend parameters, for example..

Version 6.81 WideClient only (May 2010)

- Added Lua plug-in facilities, including most of the libraries implemented in FSUIPC.
- Added a FontSize option for the ButtonScreen facilities.

Version 6.791 WideClient only (November 2009)

- Added facilities to change assorted sound devices defaults before loading each program via the Run, RunReady and RunKey parameters, thus allowing programs to use different sound devices even though they may not provide sound device selection facilities themselves.
- Added a KeySend facility allowing programs started by WideClient to be brought to the foreground.
- Made WideClient pre-process FSUIPC 1070 (keypress and release controls) being sent to offset 3110, and update its local Hoy Key tables if necessary, anticipating the keys being actioned in that way in FSUIPC. This makes for more efficient co-operation between programs running on the same Client PC, one issuing keystrokes to control or respond to the other.

Version 6.78 (November 2008)

This included many assorted improvements and new facilities in WideClient only, as listed below. There were no changes in WideServer, but the version number has been kept in line.

- Fixed a long-standing bug in the "AppOnly" shutdown facilities in WideClient, where the applications shut down (leaving WideClient running) okay but the "RunReady" applications then do not restart when FS is up and running again.
- The ButtonScreen facility has been extensively revised to provide more flexibility. The background colours can be set, and different methods of moving from page to page are provided. It has been extended further to allow a page title to be drawn too. The User Guide contains a revised ButtonScreen section with an advanced example.
- The WideClient window can be made to stay on top of other windows by using one of two new values for the "visible" parameter:

  Visible=OnTop for a normal sized Window, on top.

  Visible=OnTopMax for a maximised window, on top.
- The GPSout relay facility now works for USB devices (like PDAs) provided they respond to the "\\.\WCEUSBSH001" type of port name. Just set that name as the Port in place of the usual "COM1" etc.
- If a GPSout Port is specified, but not available at the time WideClient is started (e.g. because the PDA was not switched on), then WideClient retries every few seconds and GPSout transmission will commence when possible. Additionally, if a serious error occurs on the port, indicating a possible disconnection (e.g. the PDA is turned off), WideClient goes back to initialisation mode on the port and will keep retrying to open it again. It will be re-reading the INI file parameters when doing this, so the port can actually be changed whilst WideClient is still running and connected.
- The hold-off mechanism, to stop one client program hogging the connection, has been modified to be less severe on such a program, so that if it is the only, or main, client program it will not appear to be so sluggish to respond to mouse or keyboard operations. One particular program which should benefit is Radar Contact.

  This change also makes WideClient no longer appear to use 100% of the processor during such times. This indication was only happening because WideClient was soaking up Idle time, not hogging the processor, but it was worrying users unnecessarily.
- The facilities to Run and Close programs are extended with an ability to forcibly terminate programs obstinate enough to ignore the WM_CLOSE messages sent to them by the ordinary "Close" options. To terminate programs with brute force, instead of setting "CloseN=Yes" (or "CloseReadyN=Yes", etc), substitute "Kill" for the "Yes".

  Note that this method is the equivalent of using the Windows Task Manager (Ctrl_Alt_Del) to delete processes. Use it with care. It provides no opportunities at all for the murdered process to do any sort of tidy up or options saving.

- A facility is added to allow Thomas Richter's Electronic Checklist to be run on a hardware CDU like those from PFC, with some of the CDU keys themselves operating the options and hiding/showing the lists. Full details will be found on Thomas's website: http://www.technical-service-richter.de/
- The computer shutdown facility now operates slightly differently if there are any client application programs which have been started by WideClient, through the Run, RunReady or RunKey facilities. WideClient now tries to close these down first, then delays its own and the PC's shutdown by a few seconds to allow those programs to disconnect and close. This should stop the deadlock that sometimes occurs when WideClient closes whilst a program is waiting for a response and ignoring the Close messages from Windows.
- If the automatic server PC detection isn't being used, and both Server and UDP protocol are explicitly specified in the INI file, it could sometimes happen that the Client Computer ID sent to the Server would get lost. This had the unfortunate side effect of preventing any Buttons programmed from that Client being recognised by the Server (it indexes them according to Client ID). This has been fixed by sending the ID block again after receiving the first connection data from the Server.

Version 6.75 (June 2007)

Minor tidy up and a small correction to the code added in 6.72 for file placement on Vista.

Version 6.72 (March 2007)

- Some changes have been made in WideServer for Windows Vista. If WideServer detects that it is running on Vista, it places the LOG and INI files into the Documents folder, in the Flight Simulator Files folder (alongside saved Flights and so forth).

  However, although this is done to avoid the protection problems Vista seems to impose on "Program Files" folders, for FS2004 (and maybe before), Vista actually reports to FS, and therefore WideFS, that it is actually WinXP SP2, so at present this change has no effect. Vista seems to place files written to Program Files into a "compatibility" store, presumably aliassed from the FS Modules folder.

  When more is understood about what is going on here, more specific documentation will be provided.
- WideClient now provides correct support for the GoFlight EFIS module (up to two of them). Although this was supported before, it was completely untested, and the Mode and Range selectors were difficult to deal with. Now each of the positions of these selectors give a different FSUIPC button number. The other difference is that the Minimums and Baro dials only have two button numbers, one for each direction. There is no distinction between "fast" and "slow" operation as there is for other GoFlight rotary encoders.

Version 6.71 (November 2006)

- Minor changes to bring FSX into the action. Note that the WideServer for FSX is known as Version 7.xxx and is built into FSUIPC4.
- The RestartTime parameter now defaults to 0 to avoid regular stutters (until Clients are connected) experienced on a few folks systems.

Version 6.70 (July 2006)

- Fixed a bug in WideClient which prevented it connecting if the INI file specified both **Protocol** and **ServerIPAddr** values.
- Assorted internal improvements in preparation for future versions of FS.

Version 6.65 (June 2006)

- Automatic protocol setting added, with "ProtocolPreferred" facilities in the Server.
- WideClient supports new GoFlight device buttons and switches, provided the latest GFDev.dll is accessible (the latter is available from my Support Forum). This includes support for the Go Flight MCP Pro.
- WideFS now offers a facility to copy files. Any FSUIPC-interfacing application can ask for a file to be copied (renamed too if required) from any path to any other path anywhere on a currently connected WideFS network, whether Client to Server, Server to Client or Client to Client.

  Normally the file to be copied would be resident on the application's own PC, but as UNC paths can be used, it doesn't have to be. However it does rather negate the point of the facility if a UNC path is used for the source file and refers to another PC—unless of course that PC is a Windows Server and has no connection limits.

The destination will be a file created on a connected WideClient or WideServer PC. A UNC path must always be used for the destination -- the computer name from the UNC path is used to create the TCP/IP connection.

Files are created or replaced. Any errors such as read-only status will be reflected in a "send" failure at the source, because the connection will be severed by the receiver.

Details of this facility will be published in the next Release documentation for WideFS. Meanwhile interested developers can obtain full programming details on application to me at petedowson@btconnect.com.

Version 6.60 (April 2006)

- This version includes many small improvements—minor adjustments for more efficient operation in some areas.
- It includes a new UDP protocol option. UDP is simpler and faster than TCP, and is installed on your PC as part of the TCP/IP package. It differs from previous protocols supported by WideFS in that it is "connectionless"—data is sent to specific addresses (by IP address), but there are no checks on whether they get there and no error recovery to ensure they do. This is why it is faster, and also why I've not supported it till recently. With the block sequence and checking now in WideFS you should at least be able to detect if it is going wrong.
- The method of selecting the protocol to be used is changed. The WideClient "UseTCPIP" parameter is gone. If that was previously set to "Yes" (the default) then nothing replaces it, and the protocol is chosen automatically by the Client at run time, trying TCP, SPX (the part of IPX/SPX actually used) and UDP in turn.

  With a WinXP server and a WinXP client, you can add "ProtocolPreferred=XXX" (XXX being TCP, SPX or UDP) to the [Config] section of the WideServer.INI file, and those clients not otherwise committed will use that protocol if possible. This gives you a quick way of changing the protocol on all computers in the Network, for performance comparisons.

  If you wish to select a specific protocol on any client, just add "Protocol=XXX" (XXX being TCP, SPX or UDP) to the [Config] section of the WideClient.INI file

Version 6.51 (December 2005)

- WideClient now correctly sets the program's local path when one is loaded with one of the "Run" parameters.
- WideServer now recovers better from any accesses attempted by Clients to illegal FSUIPC offsets (which actually vary with aircraft type rather faster than FSUIPC can tell WideServer). Instead of logging an unrecovered error then effectively starting again (thus disconnecting and reconnecting all clients), only that one transaction is voided. The logging looks the same and will still be useful to track down illegal accesses.
- Facilities to divert the Log files to other folders, on other PCs in the Network, have been added. However, it has been determined that these are almost useless in a standard Windows network setup because of the severe limitations Windows imposes on the number of simultaneous connections to, say, an administrative computer. It seems to only be truly viable on a setup built around Windows Server, instead of a peer-to-peer network and I have not been able to test it on such a setup. If any WideFS user wants to pursue this he should contact me via the Support Forum.

Version 6.50 (August 2005—there were no versions 6.48 or 6.49) includes these changes:

- Program closing by "CloseReady" parameters (and similar) is more consistently successful. Timing problems at the Server end sometimes either resulted in programs not closing when they should, or more usually closing but then being reloaded again before FS had finished terminating.
- When both Server and Clients are running on Windows 2000 or XP the Clients can now locate the Server without being told the ServerName, IP address or, for IPX/SPX even the Server Node (except in some multiple network cases, where the Node might be ambiguous).

  This means that WideFS can simply be installed and, with no parameter changes, it should work.

  This is accomplished by having WideServer broadcast its name and node using the "Mailslot" facility (a variation on NetPipes). The broadcast is a very short block and, by default, it is only transmitted once per second. But it will do this continuously, and, of course, being a broadcast it affects all connected PCs. In the unlikely event that this presents too much of a load on your network you can adjust this behaviour by changing the following parameter in the [Config] section of the WideServer.INI file:

  AdvertiseService=Yes

  Set this to "No" or "0" for no broadcasting, or set a desired time between broadcasts (1–10 seconds. Default is 1 in any case).

Note that if there are more than one PCs running FS with WideServer's broadcasts enabled, it is pot luck which ones the Clients will attach to. In such a case you would still need to specify the Server details in the WideClient INI files.

If a client has attached to a broadcasting server (by using its broadcast) and that Server goes down (eg FS is terminated), then the Client keeps trying it for 10 seconds. After that it will listen for broadcasts again, so could attach to an FS on another PC.

- The parameter "ActionKeys" is now abolished. All WideClients listen for KeySends automatically.
- The Roger Wilco/Squawkbox/AVC PTT controls (RWon/Rwoff) are complemented by new ones for Squawkbox 3's private voice PTT: PVTon and PVToff. These will work with SB3 when the SB3 update supporting the controls is available. [See next item also].
- FSUIPC's controls for PTT and PVT now operate locally on the FS PC and on all WideClient PCs automatically. There is no need now to use KeySend controls nor to add the RWon/off or PVTon/off controls to the list in the WideClient INI file.
- The program load pathnames given in the Run, RunReady, RunKey parameters in WideClient.INI can now include command line parameters.

  Additionally, for users of Project Magenta's CDU program, the command line parameters can include one or more %P inserts. These are replaced by the name of the last "Company Route" loaded into the PM CDU. This allows ATC or route following programs to be loaded on a button press (by RunKey) with the name of the plan they need also to load. (A good application for this will emerge with the forthcoming Radar Contact Version 4).

  WideClient will now receive GPS data from GPSout and relay it to a local COM port. To do this you need to use GPSout 2.58 or later in the FS PC and set its port to "WideFS". Details are then added to the Wideclient.INI file where reception is needed. See the last section in the main part of the document for details.
- WideClient now features a Virtual Button-Screen option—see the section on this above for details.

Version 6.47 includes:

- Error trapping has been re-enabled in WideClient, just in case there are any more occurrences. Please check WideClient.log files if anything odd happens during a session. Don't forget that you can restart WideClient and still access its previous log as "WideClient0.log".
- The **ApplicationDelay** parameter now merely delays by "sleeping" if the value specified is less that 6, but loops processing messages and allowing other application calls when a larger delay is requested. In general, however, the default of 0 is still recommended.
- Neither Server nor Client now relies on Windows timers for timeouts, regulation or other time-dependent actions. Instead they use their own timer arrangements based on separate timing threads.

  This seems to give better response times and altogether a tighter operation, especially on multi-client systems.
- The program "Run" options have been extended to allow programs to be started and stopped from the Server via the KeySend mechanism. An extra category of Run program is added (**RunKey**N, where N runs from 1 to 9) which is never started automatically by Wideclient, but only via the specified KeySend arriving. The extra parameters are

  RunKey1 to **RunKey9**, and **CloseKey1** to **CloseKey9**

  The KeySend facility can accept any of these keywords after the **KeySendN=**

  RunKeyN, CloseKeyN, RunReadyN, CloseReadyN, RunN, CloseN

  Thus enabling you to program KeySends to allow *any* of the programs known to WideClient to be started or stopped by key or button from anywhere in the system.

Version 6.45 changes are:

- A facility has been added to allow Hot Keys to be defined in Wideclient.INI that can be seen in FSUIPC on the Server as "virtual buttons" (one of 288 buttons provided on "virtual joystick" numbers 64–72). The buttons are defined in a new section **[ButtonKeys]**, and, being implemented via the Windows' Hot Keys facility, do not need WideClient to have the keyboard focus.
- The client send queue parameters and checks are revised in order to try to eliminate the occurrence of multiple reconnections due to excessive send queue build-ups on some folks systems. The reason for the build-ups has not been determined, because I cannot make them happen at all on any of my set-ups. The changes are:

1. The check is now controlled by the parameter **OnMaxSendQ**, which automatically replaces the previous one, and this defaults to "**Log**", which simply logs the excessive queue and does nothing about it. Other values possible here are "**Flush**" which flushes the queue, and "**Recon**" which reconnects as well as flushing the queue. The last is the action that was occurring in 6.44. The danger is reconnecting is that the same problem immediately recurs, as on a reconnection all the request read data has to be re-requested, as if the client is a new one.

2. The **SendScanTime** parameter, which defaulted to 10, is replaced by **NewSendScanTime**, which defaults to 50. This limits the number of send queue scans to 20 per second, but it now does *not* limit the number of frames sent. Each single scan will send as many queued frames as it can before the Windows network indicates that one will block.

3. The **ApplicationDelay** implementation is changed to make the timing more accurate. Previously the value of 6, for instance, could have actually given rise to a delay of anything from 6 to 55, because of the granularity of the timer being used. The greater accuracy should make things run smoother still, but, now, to guarantee an average which approximates to the delays in WideClient 6.41 and before, the value would have to be set to something like 25.

   I recommend strongly, however, that this parameter is left to its default of 0. If you have increased this in an attempt to get over problems, please restore it to 0 and see if the problems are better dealt with by the **NewSendScanTime** change above.

- WideClient now logs interim performance data as well as final performance data when closed. The interim values are logged whenever the Server sends out any type of Close message that doesn't actually result in the Client closing. For instance, if the **AutoShutdown** parameter in the Server is set to "Apps" then, even if application closure is not allowed in the client, it will still log the performance so far when FS is closed.

- WideClient performance data now includes the maximum and average send data (frames and bytes). You will find that the averages tend to be surprisingly low, peaking really soon after connection (and reconnection) or when new applications are started. Upon closure, WideServer also logs an overall average "received" data line for every separate client which has been connected, but this is averaged over the whole session so the figures will tend to be less than those shown at the Client.

- Data for individual client applications is also now logged, referenced by its Process Id number (which is logged against its name when it connects). This data shows to number of FSUIPC_Process calls it is making and the actual total data size exchanged between it and WideClient. Hopefully this will be a useful aid to optimising applications.

- WideClient now sends its version number to the Server along with the Client PC name. This is logged by WideServer in the connection message and is a useful way to check that you do have all client PCs up to date. So that programs like the PM file checker can check this easily too, the *lowest* WideClient version number is provided in the 16-bit word at offset 3520, as the version number x 1000 in binary-coded decimal (BCD), e.g. hex 6450. If WideServer is out of date (before 6.442), or any one WideClient is earlier than 6.442, then offset 3520 will be zero.

- WideServer now provides the IPXROUTE data (with decoded ServerNodes) if the main details returned to it by Windows has a non-zero Network Number and a suspicious-looking MAC address for the adapter.

- Both WideClient and WideServer now rename the existing LOG file as "....0.Log" before creating a new one when started. This allows both clients and FS to be restarted without losing valuable diagnostic data, which can now be retrieved afterwards.

- More errors that previously would result in a message box report on screen are now merely logged. This is especially useful on Win98 and WinMe which seem to vary somewhat in how they report errors when one of the supported protocols is not installed.

- Additional data, items which do not change too much during a session, are now requested by WideClient on loading, so that they are available to applications immediately they connect. These include, for example, the FS install path, the aircraft name, and the Project Magenta NetDir field.

- A problem discovered in the way the "WaitForNewData" option operated is fixed in this release. When there are a number of different client applications all clamouring for different data on initial start-up, the "WaitForNewData" timeout was not always applied correctly—data requested by one application could, on arrival, sometimes effectively cancel the timeout pending for different data for another application.

  This is a very old bug, but it has only really had any noticeable affect on things recently with the other performance improvements that have been made, and the ever increasing load placed on WideFS these days with so many demanding applications.

Version 6.44 includes:

- The WideClient "TimeOut" parameter is now removed, and replaced by "ApplicationDelay", which more clearly signifies its true action. The default has been reduced to 0, which actually suits most current programs better than the Timeout default. To get the original value, use 6 for this parameter, not 12—the original 12 milliseconds has been halved internally for some time. A non-zero value is more likely to be needed on Win98 or WinMe clients, because the timeslicing isn't so good with those operating systems.
- The "Priority" parameter gas been added to the WideClient INI "Config" section. This allows the relative priority of the main program/application, and the now separate receive and send threads, to be changed should performance warrant this.
- Parameters are also added to controls the maximum length of the Send queue, and what action is to be taken in the event of this being exceeded. However, with the other changes made this should never need any changes from default values.
- The AutoShutDown action from the Server is followed by a one second delay before FS is allowed to terminate. This may improve the shutdown actions on Client PCs.
- The WideClient logging for the "monitor" option now provided separate timestamps for each separate application read/write call ("FSUIPC_Process") rather than lumping similar calls together.
- WideServer now records sumcheck errors in blocks correctly, even when no additional logging is requested, and if there are any it provides a total count in its performance summary at the end of the Log.
- WideServer now handles the reception of partial blocks correctly, re-forming them after arrival. The occurrence of such split blocks wasn't actually thought possible until recently—the Winsock protocols and buffering are supposed to deal with this.
- Wideclient now provides an additional method for sending Keystrokes as a result of KeySend parameters programmed in the FSUIPC Keys or Buttons pages (or indeed in the WideServer.INI file, where that original version of the feature is still used).

  This method is by "SendInput" and operates in the same way as FSUIPC's keystroke programming for Buttons. It is enabled in Wideclient.ini by the line **UseSendInput=Yes**. The SendInput method is not directed—whatever program has the focus will receive the keystrokes for any undirected KeySends listed. All directed KeySend actions will still use the previous method—either record-playback, or, if **PostKeys=Yes** is set, message posting.

  Note that this method should work well with TeamSpeak, for PTT actions from an FSUIPC programmed button.
- WideServer now detects the bad **ServerNode** which is obtained sometimes on Windows XP. It seems this is due to some "loopback adapter" XP installs. To find the correct ServerNode, WideServer now runs the WinXP utility IPXROUTE and extracts the Nodes it lists, and logs them in the correct form.

Version 6.41 includes these changes:

- Minor adjustments to improve performance a little. Now, providing the main FS PC is powerful enough to run FS with relative ease, *and* you have no particularly slow or overloaded Client PCs, you might be okay relaxing the FS frame rate limiter, or even setting it to "unlimited". Experiment first.
- WideClient can now be run before its Server PC is even switched on. It will simply keep retrying the connection. It will make a log entry every minute or so to tell you what's going on, but the advantage of this is that you can have WideClient starting up automatically (and loading your applications), and you don't need to worry about the order in which your PCs come on.
- WideServer can now automatically tell clients to either shutdown completely (**AutoShutdown**=Yes) or close their "CloseReady" applications (**AutoShutdown**=Apps) when FS itself is closed normally.
- There are no extra parameters for the WideFS INI files, but you might like to note that you can now set Monitoring for up to 8 separate regions—previously only one. This is by listing the offsets and sizes in the **Monitor** parameter.
- In the client PCs, monitored offset areas are now also logged when read/written by applications, not just when changed via the LAN. The has helped clear up some timing problems .
- WideClient now ensures that data associated with a timestamp is on WideServer's list when the data request is made, not leaving it till the data itself is first requested. This fixes some cases of incorrect data being read because the timestamp change is seen first.
- Finally, quite a big change: WideServer now automatically provides a service for both TCP/IP and IPX/SPX protocols if they are both installed. The "UseTCPIP" parameter is no longer used in the WideServer INI file (it will be deleted).

  You must still specify, for each separate Client, whether TCP/IP or IPX/SPX is to be used for that Client. If IPX/SPX is installed on the Server, the WideServer log will always provide the "ServerNode" parameter needed for the INI files of those WideClients using IPX/SPX.

The main benefit of all this is that you can now mix IPX/SPX and TCP/IP clients. This solves the problems of IPX/SPX on mixed WinXP + Win98 setups (like mine, in fact). You can run IPX/SPX on those clients best suited to that, and TCP/IP where IPX/SPX is problematic (as with Win98 connecting to a WinXP server).

Version 6.401 fixes problems with the **WriteLocalDirect**=No option of 6.40, which was defaulted. The idea didn't work for several important programs. The parameter is now scrapped, and the original problem (the reason for the change) has been solved a different way—by having WideClient save up the results of writes to application offsets and send the results to the server when eventually connected.

Version 6.40 includes a number of changes:

- A new "shutdown" facility has been added, allowing application programs to be shut down (with WideClient still running) via a different WideServer hot key or application program action. The latter is by the pattern 0xDCBA being written to offset 0x3320 instead of the more severe 0xABCD. For more details refer to the **CloseAppsHotKey** details above.
- The number of programs that can be started and closed by WideClient has been extended. The **RunN**, **RunReadyN**, **DelayN**, **DelayReadyN**, **CloseN** and **CloseReadyN** parameters now operate with **N** running from 1 to 9, giving 9 programs startable immediately and 9 when FS is ready.
- Retries on frames which are blocked by Windows (Winsock, actually) are now made no more frequently than every 250 mSecs. Previously the normal FS frame controlled the retry timing. It is hoped that this reduction will reduce the occurrence of multiple retries, in the hundreds, sometimes reported, those which it is said seem to cause Project Magenta (for example) to act many seconds late. Whether it will help or not remains to be seen—the types of problem reported have not been reproduced here.
- Writes to application-owned offsets in FSUIPC are no longer simultaneously written to the local WideClient memory. If this causes any sort of problem, the former behaviour can be re-instated by setting **WriteLocalDirect**=Yes (see above) in the WideClient.ini file, but in general it is considered safer not to.
- WideServer now performs at intervals of 250 mSecs (¼ sec) when FS is engaged in menus and such, rather than 5 seconds as it was previously. The very slow rate was intended merely to stop Clients timing out and attempting reconnection, but it seems there are applications for some add-in modules to retain client interactions whilst in their own menus. The 4fps rate is still much slower than a normal running rate but should be sufficient for this.
- Originally included in version **6.233 (**a limited release) are improvements in the handling of timestamp reads (see next item). When timestamp reads are combined in one process with other IPC operations, especially writes, the timestamp read is separated out and dealt with first. This ensures that they are seen to change later, should any of the writes prove to be one of the commands instigating a timestamp change.
- Similarly, an earlier limited release (**6.232**) included changes to assist with those programs which use facilities in FSUIPC that rely on timestamps changing to know that specific requests have been fulfilled. To be specific, these are the IPC based requests for path information, airport runway-in-use, and weather setting and reading, especially for specific weather stations on FS2004. (These changes were originally in a version **6.231** but did not work as desired in that version).

Version 6.23 contains a fix to the RestartTime facility for a bug that could, in certain circumstances, cause WideServer to continually restart so causing every Client to reconnect – at 10–15 second intervals! Before this fix the problem could be averted by setting the **RestartTime** parameter to zero or something larger than 20.

Additionally, in this version, the settings of all of the [Config] parameters are added to the INI file so that, in future, it will be more obvious when they are different from previous versions. This may have allowed the RestartTime problem to be diagnosed earlier.

A fix to the way Wideclient updates the local memory copy of FSUIPC offset data in included. Before this version, when an application wrote to an FSUIPC offset with no connection to the Server yet established, the local memory, in WideClient, was still updated. Now this doesn't occur until the connection is made.

Version 6.222 contains a change to WideClient only, this being the addition of the "AllowShutdown=AppOnly" variation described in the text. WideServer remains at version 6.221.

Version 6.221 is a minor release that only affects users with GoFlight GF-TQ6 throttle quadrants when used on the client. The changes merely fix the problem of fast/slow dial turning on GoFlight units like the GF-RP48 when a TQ6 is also present.

Version 6.22 includes new facilities to recognise buttons on Joysticks, EPIC and GoFlight equipment connected to client PCs, and transmitting button events to the Server for processing in FSUIPC's Buttons programming page. This needs FSUIPC version 3.20 or later.

The only other change has been in WideServer, to speed up the connection on initial load using the FS2004 "ready to fly" indicator provided by FSUIPC. Timing on FS2002 and earlier is not affected.

Version 6.101 was originally only a small change in WideClient.exe only (the Server DLL was still version 6.10). WideServer was updated to 6.101 later. The change in the Server is minor: merely to prevent an error message complaining about the failure to send a block for more than 5 seconds being repeated for every further retry of the same block thereafter. This is a cosmetic, not a functional change.

The earlier change in WideClient only affected Project Magenta users. It helps eliminate some apparent pausing when the MCP or FMS modules are run on the same PC as one of the PFD glass cockpits. The change is to treat the lower memory allocated to PM (04E0–0537 inclusive) in the same way as the upper offsets (4000 and higher). These are updated locally as well as transmitted to the Server, whereas the normal FS variable areas 0000–3FFF are only transmitted to the Server, as the read-back is not always the same. I cannot understand why this potential problem has never been identified before, as it would have been so for years. It was only reported recently!

Version 6.10 includes two changes in the Server part only:

- It provides a short period of grace (3 seconds) during which WideServer will tolerate lack of access to FS variables without assuming an extended "stop" situation, and
- It avoids treating any longer stoppage as a necessary reason for restarting all the services to Clients. This latter change is optional (via the **NoStoppedRestarts** parameter) but is enabled by default.

Version 6.02 fixes a timing problem which can occur if WideFS gets loaded by FS before FSUIPC, or even very soon after. It seems that whilst FSUIPC is verifying the Registration of WideFS, the WideServer DLL may already have checked access and been rebuffed. This has been fixed by making WideServer retry for the access permission 10 times over a period of 20 seconds, which should cover all eventualities at start-up.

Also, the start-up has also been made a little faster, and the message "stopped" (which worried some folks) is replaced by "getting ready" or similar. This is actually more accurate at start-up, even though internally it is the same—it is the access to FS innards which is stopped until FS's aircraft and panels and so on are fully initialised.

Version 6.00 works on FS2004 as well as earlier versions of FS, and needs at least version 3 of FSUIPC. 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).

Published by Peter L. Dowson, 11th February 2012