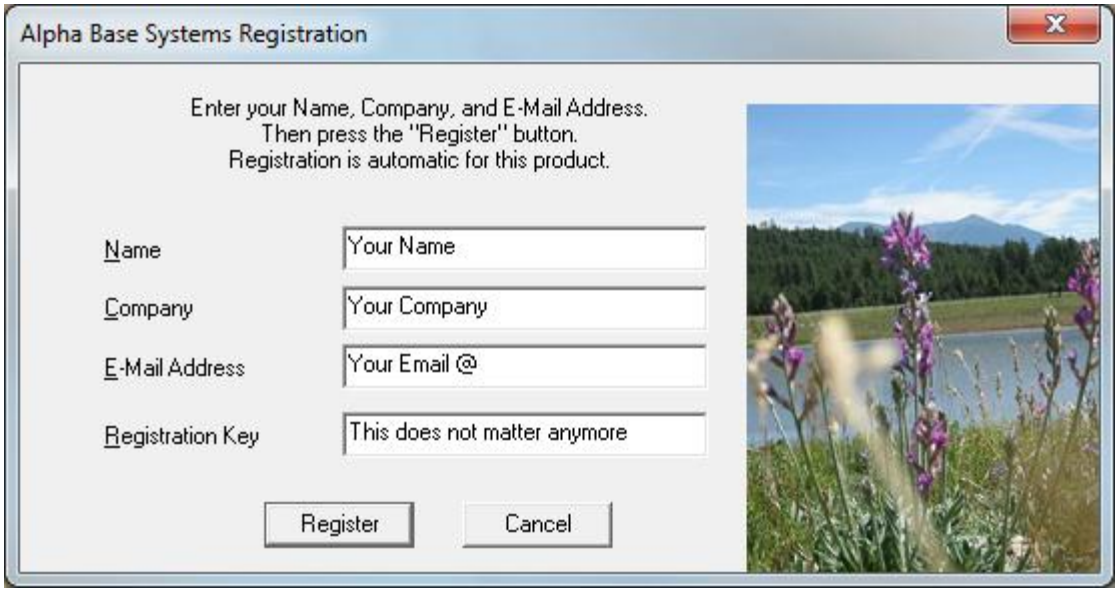
Release 9.9(10) of ABW

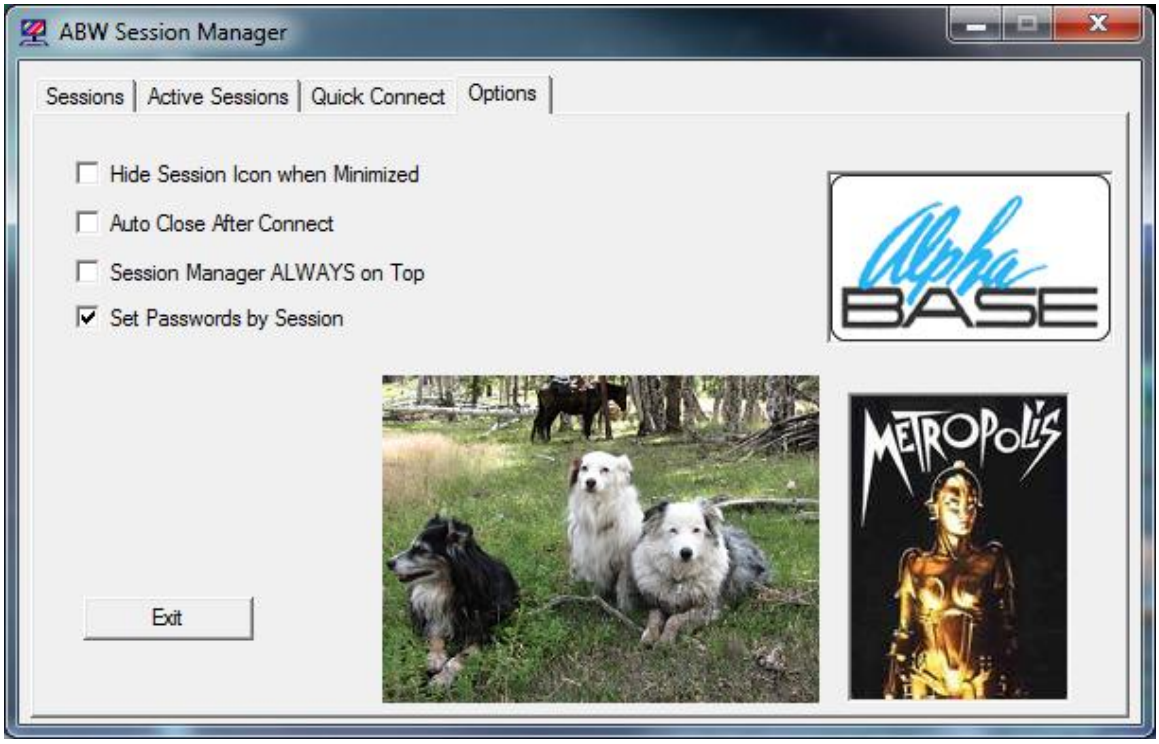
NOTE: Some features require 9.9(10) of Met or greater, some do not. This first section does not require a Met update:

1) Cosmetic changes. Mostly the layout was left the same and the logic the same. Cleanups and logos and pretty things.
2) The problem of session names with a space on them blowing up has been solved. It replaces spaces with underlines for you.
3) Registration has been automated – no need to call for an encode. Go to Help -> Registration and re-enter your information – you can change your user, company, and email address – ignore the Registration Key field. Click OK. You will then be registered for 9.9(10).



4) In the options sections of the session manager is a new check box "**Set Passwords by Session**." Currently ABW keeps passwords by the host name. If you want different User IDs and Passwords for different sessions – no matter where the destination – check this.

   Many people have one public machine they connect through using the magic of port forwarding their firewall. So the one single host may be used to connect to many actual hosts. With the new SSH feature (below) you will be able to connect to a single SSH server (in the public domain) and pass through to other machines in the private network.

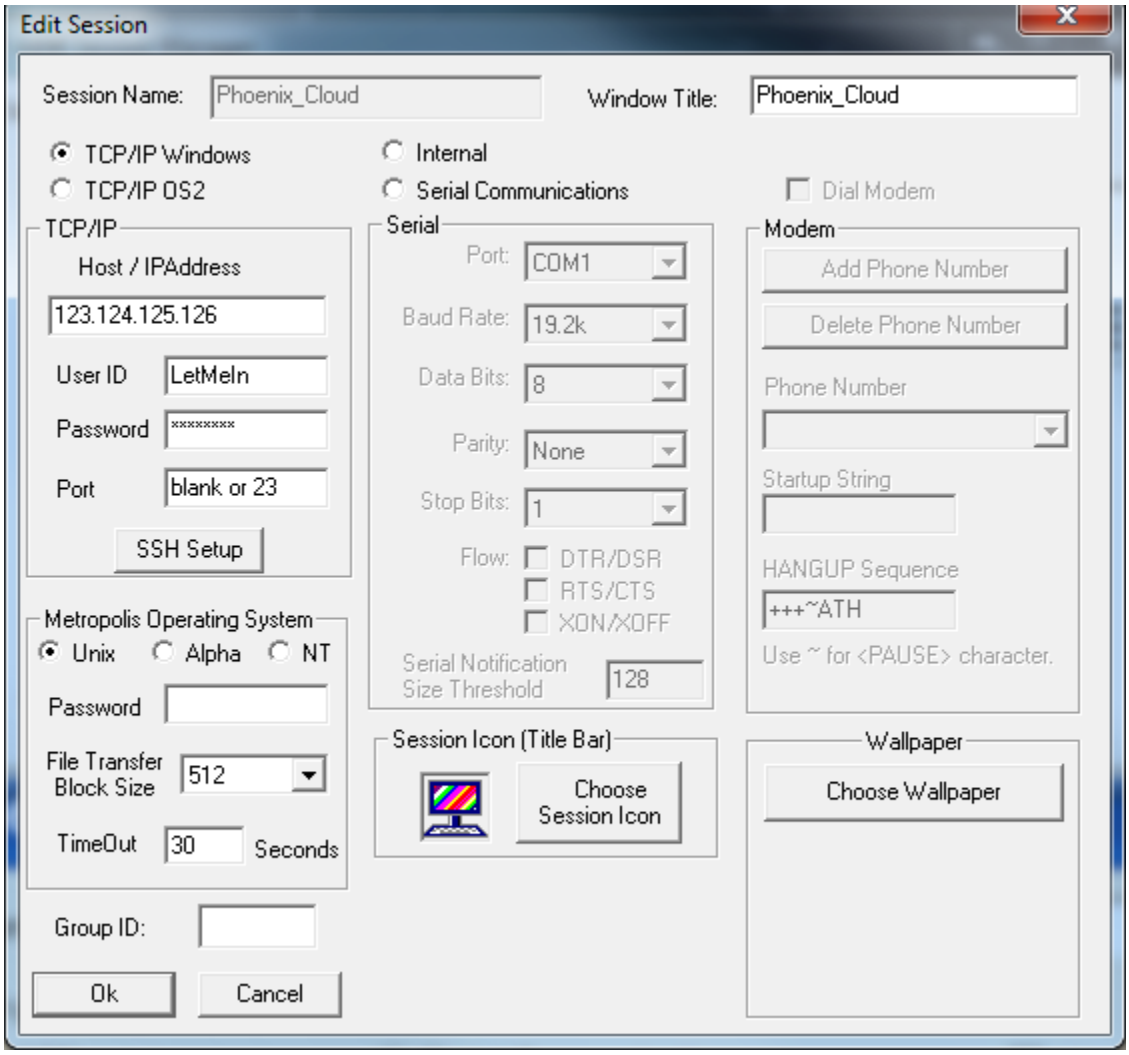


Note: When you check this option, the first time you connect or edit a session the password file will be MOVED to the private directory for that session. This means if you have several sessions with the same host name, all others will no longer have a user id and password stored. If you don't remember the password, you can copy it to all appropriate session directories. The file is located in the SESSIONS directory under ABW, and then the session name (so for a session named banjo the directory is SESSIONS\banjo). The file name will be the hostname.uid. It's current location (without the box checked) is in the ABW root directory.

5) **PCI compliance and medical records change** – all passwords stored in files are encrypted and then changed to base64. Conversion of old data will only occur when you edit the session. If you want to do this badly enough, go through each session and press edit and OK. This is not an excuse to use crummy passwords! All rules about good passwords should still be observed. It will not do this for any password less than 4 characters – if you are using 1,2,3 digit passwords you are not serious about security anyway. There is no reasonable encryption method for less than 4 bytes (in fact all the really good stuff does it in 16 byte chunks).

6) **PCI compliance and medical records change – SSH!!!!!!** You can now use SSH integrated into ABW.

To access the SSH setup you do so from the normal session edit screen:

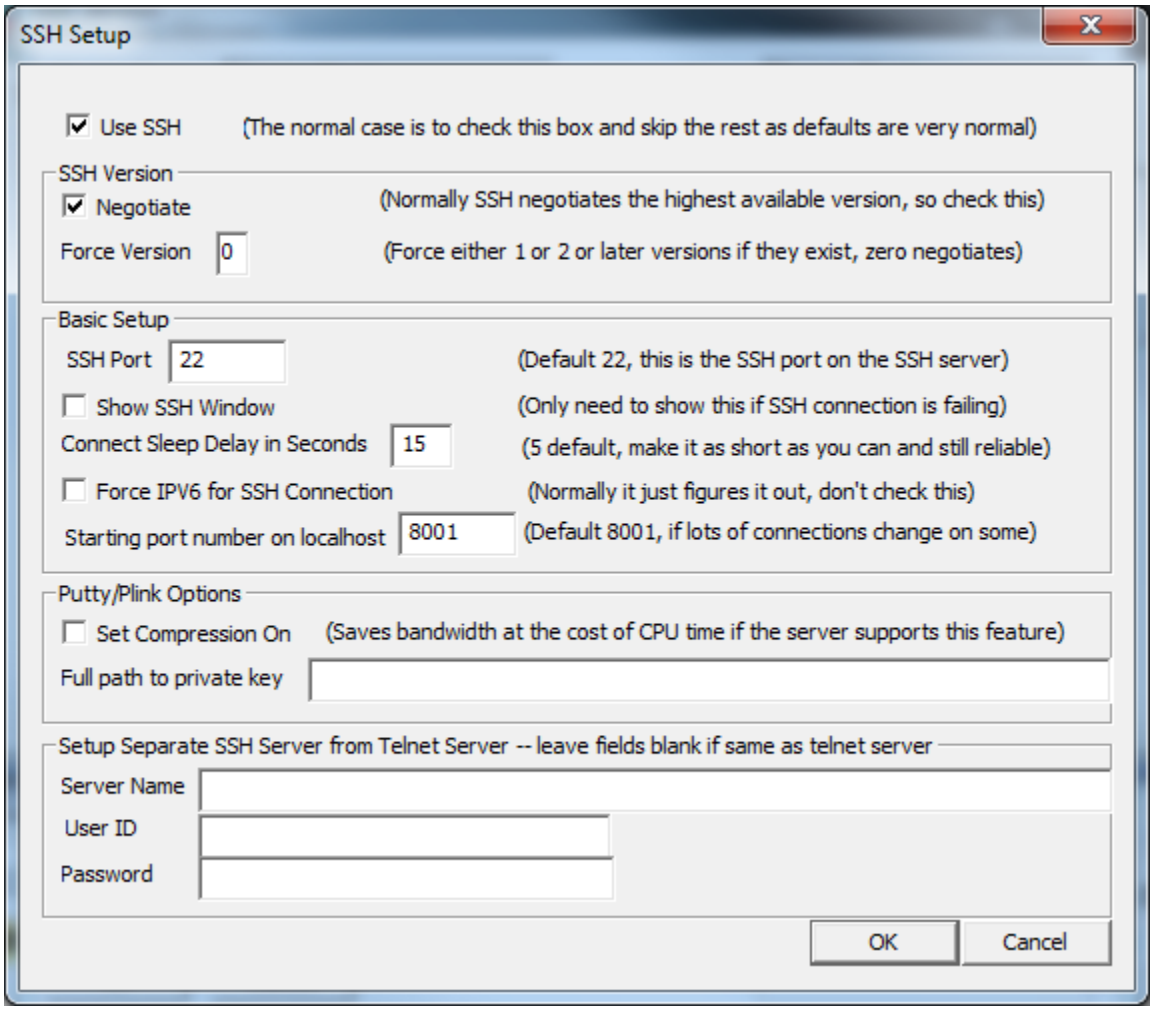


Notice under the Port the button for SSH Setup. The information on this screen will generally NOT change. The user id and password and port will remain the same as the final destination. It does not matter what is the SSH server – this is where the telnet will be connected to after the SSH tunnel exists – meaning you generally don't change these fields. It is important to note, though, that it is the SSH server on the remotes server that must interpret the hostname. So if you put a "pet name" into your own hosts file and the server does not have the same, it will fail name service on the remote side. This may also come into play with NAT networks. Sometimes within the network the public IP cannot be accessed. So you may need to change this to the private NAT IP address. Generally if your hosts are "real" URLs that are found in name service, or a hard coded IP address, then there are no changes required.

NOTE: you may have to make changes to your SSH (sshconf) to allow forwarding. You may have to make changes to your firewall to allow port 22 (SSH) or whatever port you use for SSH. You are advised to convert all

connections – internal and external – to SSH, and then change your firewall to disallow telnet except from these SSH tunnels. Otherwise, you will NOT be PCI compliant nor compliant with medical records privacy requirements.

Now for the SSH Setup screen:

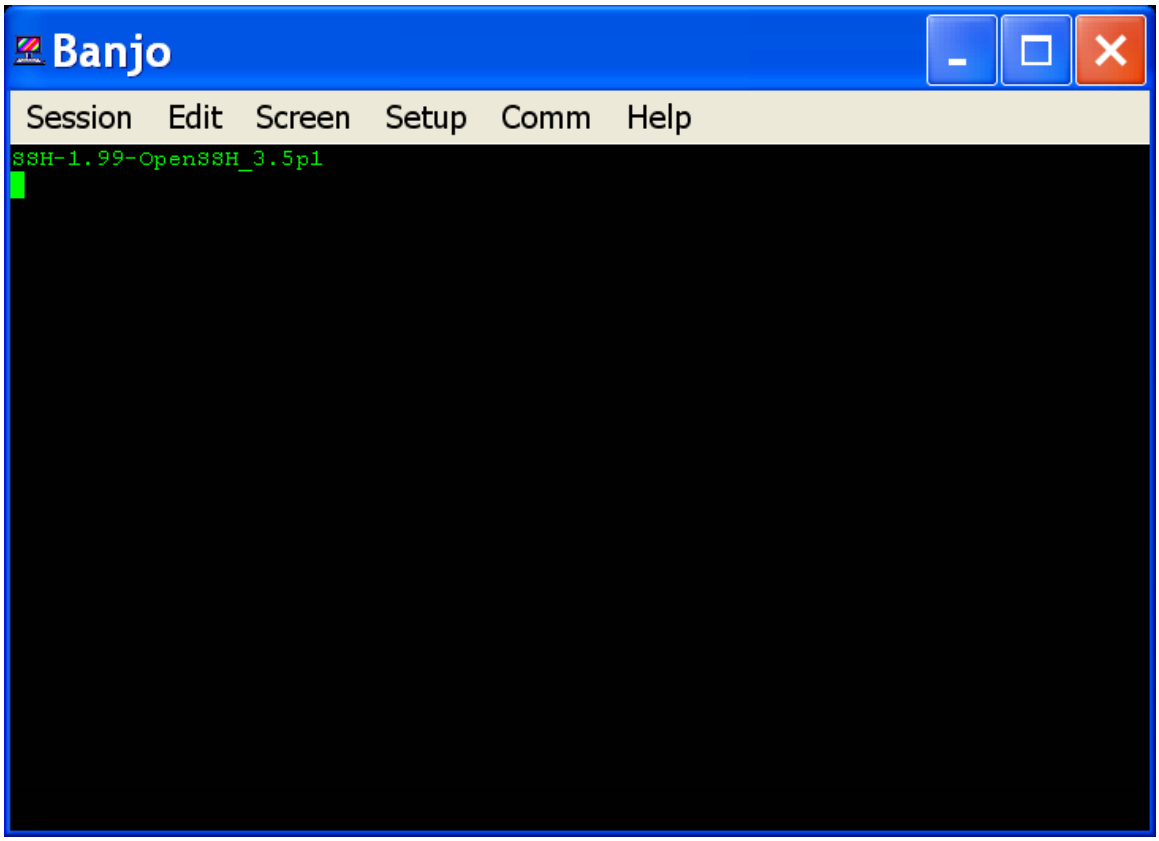


For a simple SSH connection the only thing you need to do on the client is check the "Use SSH" box and then OK. All the defaults are normal. You may want, the first time, to check "Show SSH Window" and past the certificate caching explained below. When you connect you normally will not see the SSH window (nor will it show in alt-tab). You will see it in Windows task manager as "plink.exe."

NOTE: we included plink.exe with the install. It MUST exist in the ABW directory. It will NOT look in the path for other versions. This is deliberate so that if you have installed putty/plink for other reasons we will not be interfering. But, we are not tied to this version of plink (right now it is absolutely current). You may update it at any time yourself, should a new version come out and you

want to resolve a security issue (the main reason a new plink would be created is to resolve security issues or add new security features).

Now you press OK on the Edit Session screen and you are done. You are ready to make your first SSH connection. When you go to connect you will get a sleep period waiting for the tunnel to connect. A window will pop up letting you know it is waiting. There is no communication between plink and ABW so there is no way for ABW to tell if the tunnel is ready. You will have to experiment to minimize this time. This is changed in the "Connect Sleep Delay" box in SSH setup. If ABW connects too soon you may see a screen like this:



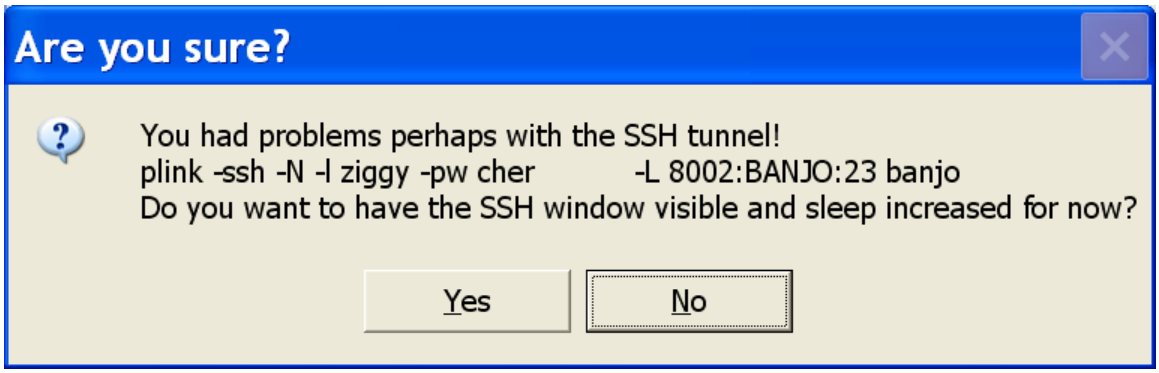
This will generally be the SSH version of the server and indicates that the connection to the tunnel was done before the tunnel was ready. Increase the Connection Sleep Time.

Important – READ THIS in Full!

If you have any problems connecting, then check the Show SSH Window in SSH setup. You may also want to make the Connection sleep time larger (30-60 depending on how old and slow you are) so you have time to see the window and/or respond to prompts within the window.

The main reason an SSH session will fail is due to firewall problems. With the plink window invisible, you will not see if plink is able to connect or not. Unfortunately, Putty/plink will exit if the connection fails completely. Meaning no matter the sleep time, it will look like it mysteriously died. So the window will flash up and then vanish before you see anything useful. You will see later in this document how you can see the plink command, which you can then test from a DOS prompt and adjust until you get it correct. One of the HUGE advantages of using Putty is that the network administrator has no excuse if things don't work – they can download Putty themselves, and when Putty works, ABW will work. It is more or less a standard for SSH in Windows.

The second reason that will occur on the first connection to a site without a "trusted" certificate (e.g. from Verisign or some such) plink will ask you if you would like to connect anyway (and cache the certificate). If you want it to work you must enter 'Y' and return on the plink window. If you did not click the option to show the SSH window (which is how I like it) then you would not see this prompt which makes it very hard to type in the answer! Fortunately, if it fails the first time and you ask to reconnect it will ask you if wish to make the window visible and it will also show you the plink command line (so the SSH tunnel experts can Google plink or putty and verify the command is correct – you may also go into a DOS window and type the command in manually to test it and perhaps to change parameters). See below, you should enter Yes! This will not change your setup and future connections will still not see the plink window. Note that this cache expires occasionally and it may crop up every so often as an annoyance (security is an annoyance). This can be influenced on the SSH server side.



This adds 15 seconds to the sleep time (temporarily) and puts the plink window up open and on top. You can then see error messages and respond to user prompts (be snappy, you only have 15 seconds plus your original sleep time). If you wish you can always make the first connection with the window showing and a large sleep time … but generally this is not needed because of this feature, and because the cache will expire and you will have to deal with this again. You will need to train users (and if you are a user reading this, remember this ☺). Note that if the SSH Window is already checked as visible this dialog box does not

come up and you will never see the plink command. If you want to see this dialog make sure that in the setup you do NOT check the box that makes plink visible and that you do a reconnect upon the first failure. However the easy way to avoid the problem is to have a trusted certificate on your server and client.

NOTE: if you kill plink.exe in the Task Manager of Windows, the ABWs that use the tunnel will lose their connection. Plink may die for other reasons, most notable SSH policy on the server (which may disconnect due to idle time). If that happens you will lose the connection.

Since this is a "tunnel" and not a pass-through SSH session, this opens up a world of possibilities. When you open up a second ABW window to the same session, it will NOT create another plink – it will simply use the existing tunnel and there will be no sleep.

Additional SSH Options. SSH version is best left with the negotiate box checked (or zero in the version). This will allow plink to negotiate a version with the server, which is generally the highest both can talk in. Which these days is likely a 2. If you want to ensure level 2 you can uncheck the negotiate box and enter a 2 for the version. However, this may make a connection impossible with older machines. In fact, if it is really old you may be forced to put a 1 as the version. If level 3 SSH is ever released you may get a new putty and enter a 3 here. However, if you put something higher than 2 here, you will not connect.

The SSH port has NOTHING to do with the port on the Edit Sessions screen. This is the port on which the SSH connection is made. Normally this is 22.

Show SSH Window should normally not be checked, no need to see it and it is not interesting except in cases noted above.

Connect Sleep Delay in Seconds – I find that locally 4-5 is very reliable. Remotely 10-15 is sometimes required. You will have to experiment. Also, if a connections fails once, you can always just try again and often it will make it, assuming your time is not silly-small.

Force IPV6 for SSH connect – check this if the SSH server connection must be IPV6 (normally this is automatically handled by plink). If you use this feature likely you will need to make changes in the Setup Separate SSH Server section.

Starting port number on localhost. A tunnel works by setting up a valid SSH connection and then accepting connections on one end of the tunnel and passing them through the SSH tunnel (meaning encrypted) and then forwarding the connection to a connection on the other end. The connection on the ABW side will then be to "localhost" and an arbitrary port number. It will pick the arbitrary port number from this field. If you have a lot of different servers you will save overhead a LOT by making these different, separated by say 10, for each server.

This is because if 8001 is used to connect to Banjo, and you then try and connect to Bear on 8001 … the Bear connection will fail on the first attempt. It will then try port 8002, 8003, etc. It does give up after 1000 ports. To be clear, it does not really hurt to leave them all at the default setting of 8001, but it will connect a little faster if you play games.

There are two "pure" Putty/plink option. One is to set compression. This works gloriously. The tunnel will be secure and compressed – so the bytes to and from the remote machine go down. The other is to use your own private key (generally a trusted one is the only reason to do this). Almost nobody needs this nor should they bother, but … the option is there.

The Setup Separate SSH Server from Telnet Server section requires a long discussion of tunneling and how it works and how it can be used. If you are not interested, skip to the next topic.

These three fields normally default to the analogous fields on the Edit Session screen. You may fill in any number of them to override. So if you just want to change the server, but have the same user id and password, that is valid.

Since ABW is simply using Putty to establish a tunnel this can easily be used in cases where you are connecting to a network that has one public SSH server that you connect to first, and then from that server you make connections to the actual servers you want access to. So say that ftp.vicsmba.com is a public server (it is) and Banjo and Bear and Solo and so forth are private machines with no way to access them from the outside. Normally one would connect via SSH (or in the old days telnet) to ftp.vicsmba.com and then from there telnet to the others. Hassle. With a tunnel you can set up an ABW session in which you would put banjo as the Host/IP address in Edit Sessions and then on the SSH Setup screen enter ftp.vicsmba.com as the Server Name. Done. Unless you have a different user id or password on the SSH server, in which case you enter these as well. When ABW connects with this session plink will contact ftp.vicsmba.com and negotiate an SSH session, and when complete, will then attach your ABW (talking to local host port 8001) to banjo port 23 – allowing ABW to then have a telnet connection to banjo that is secure. If you use the new option above to keep passwords by session, you can then make the same setup for every internal private machine, and the passwords and so forth will be kept apart.

If you are only connecting to one server and you want to "disable" telnet you can almost do so. Set your firewall to not allow port 23 at all. Set "Server Name" on the Setup Separate SSH Server to the external server name (like banjo). Set the "Host/IP Address" on the Edit Session to localhost. This means that the SSH connection will be to the normal server (say banjo) and then the telnet connection will be to "localhost 23" – what is important here is that the "Host/IP Address" is what the SSH server on the client is actually opening with the SSH session. This means that whatever you put here must be understood BY THE SERVER. If you

use the same name servers and/or hosts files these are normally the same when doing a simple connection to one machine. By putting localhost into the Edit Session you force the SSH server (on the server) to connect to localhost – which will NEVER BE SEEN by any firewalls or switches or whatever. It does not even see the network card. So this allows you to tell your firewall that port 23 is always blocked, yet it works fine within the server.

If you are connecting to multiple servers through a single SSH server you cannot completely remove telnet. You must allow telnet from that public server to the private server that is your ultimate destination. You can certainly make it more secure by removing the telnet client from the public server (uninstall it), using a MS Server for the SSH tunnel (new ones don't have telnet clients), and then setting your firewall to only allow telnet connections that originate from the SSH server. With no telnet client on the box at all (remember you uninstalled it) this means only the SSH tunnels are sourcing telnet. It is also possible to have the SSH public server tunnel to the final destination using SSH. This is a setup in SSH on that server and beyond the scope of this document – use Google. In that case, you would be using the technique described above as the SSH server would be forwarding you to the final destination. If this is done, you can also set the firewall to disallow all port 23 (telnet) traffic.

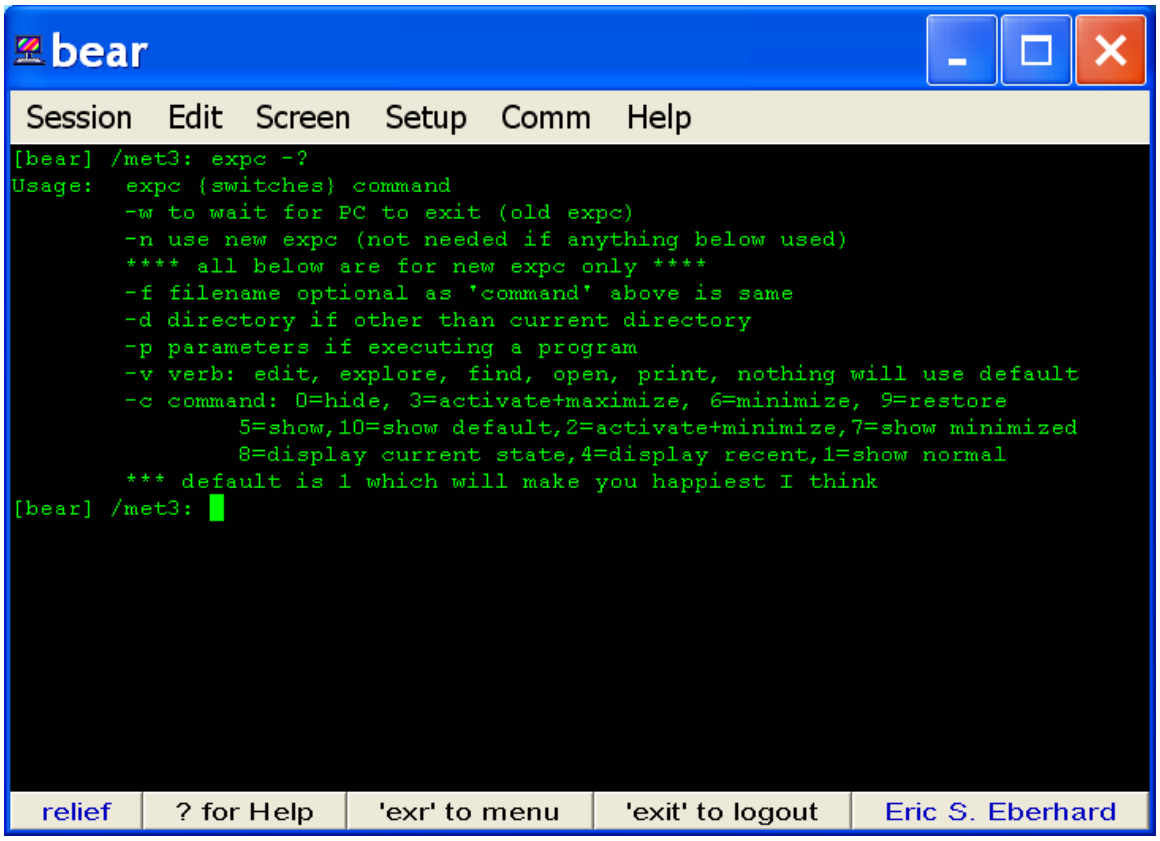
NOTE: completely disallowing telnet is probably ideal. In the multiple server environment this is difficult and likely overkill – but my advice is to try and do so. Also note that for PCI compliance even internal ABW clients should connect with SSH.

NOTE: you may need to consult with the network administrator and the administrator of your servers to set this up in the most secure manner. There are many different versions of SSH servers on many O/S and this document is not intended to cover these areas. However, since we use Putty/plink it is very easy to Google how to do it with Putty – there are many documents on this subject – and all will apply to ABW. In general, although we will try and be helpful, Alphabase is NOT going to be setting up your network, firewall, and servers, and Alphabase is NOT going to be responsible for telling you how to make your system PCI or medical records compliant. You need to consult with security experts. Specifically we are supplying a Putty/plink integration for ABW, but YOU are responsible for making sure your meet all security requirements. Our opinions above are guidelines to explain how the connections work, not the ultimate word on security.

ALL OF THE BELOW FEATURE REQUIRE MET 9.9(10) or higher:

1) **expc** has been greatly enhanced to use the Windows "OpenProcess" function which allows you to open files without specifying the program to open them

with. The PC will use the extension to detect the correct program to open the file. So an xls file will be opened with Open Office or MS Office, depending whatever the user has installed. You may pass the filename and directory separate from each other, and the parameters separate as well (which keeps confusion down). You may pass a verb for how it will open. And you can chose the state of the window, such as active or not, minimized or maximized, etc.



A VERY KEY part of this is that you may open just about anything … a file, a directory, a URL, whatever you like.

2) A macro to use the above expc from directly within BASIC (using an XCALL, not system calls to expc!). The possibilities are limitless. For example, let us say you would like in order entry to show the customer service representative the catalog page from your web site for an item. Easy. Pass the URL as the file name and you are done (well, you may have to store the name, but in our application we managed to call the file name the same as the part number). So:

   \EXPC http://www.vicsmba.com/ourpics.html

   Will open that URL all day long (and from the command line expc –n http://www.vicsmba.com/ourpics.html will do the same)

You may also want to place users into directories to look for files. This also is easy:

\EXPC c:\tmp

Will open that directory. If the verb is "explore" it will open with the explorer tree to the left.

This becomes very useful if you have Samba running, of course. Another use we have for it is to open the output of our reports as spreadsheets. We used to always create them and then make the user go hunting for the file (we did give them the name) through Samba. Now, instead, we can open the file with:

\EXPC FNAME="report.csv" DIRCODE="\\banjo\tmp"

This will (using Samba) open up on the server banjo in directory \tmp (a Unix computer) the file report.csv (a spread sheet) with the Windows program the user has selected to open spread sheets with (like Excel).

3) F8 Messages can be set to use the expc features and you can then keep your messages in .doc or .txt or whatever you like. If you put it in a doc file don't expect printing it and parsing it to be easy … you are on your own.

End of Document