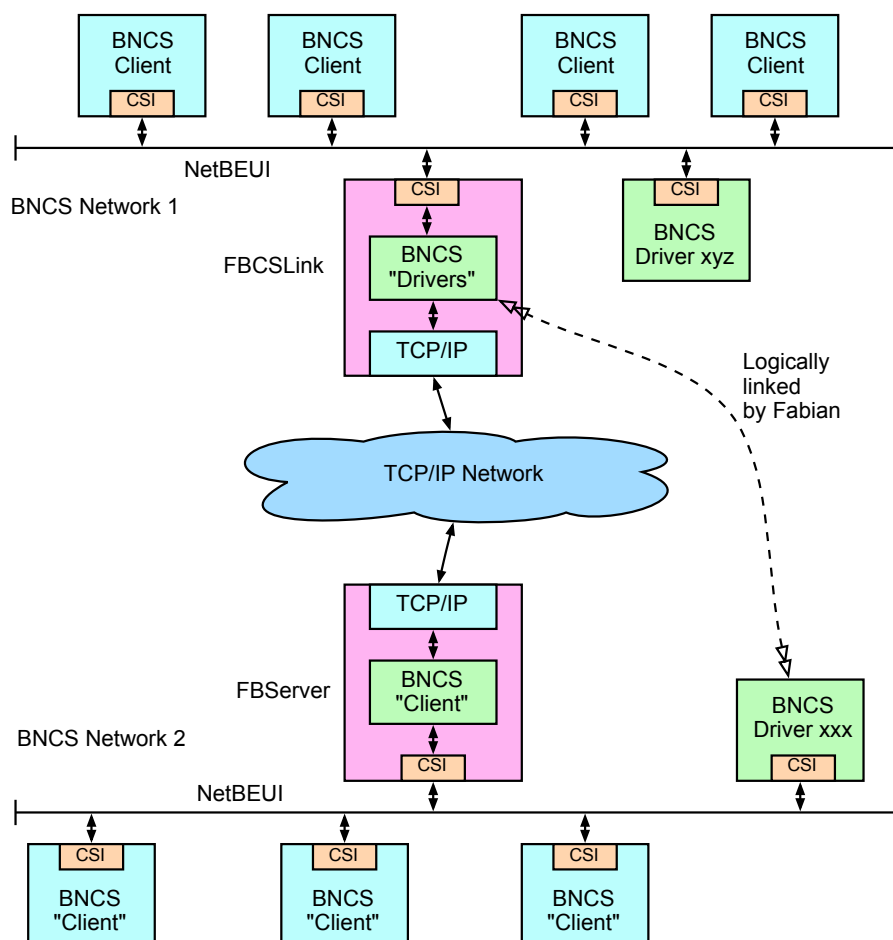


Using FABIAN for BNCS Inter-Network Linking

Introduction.

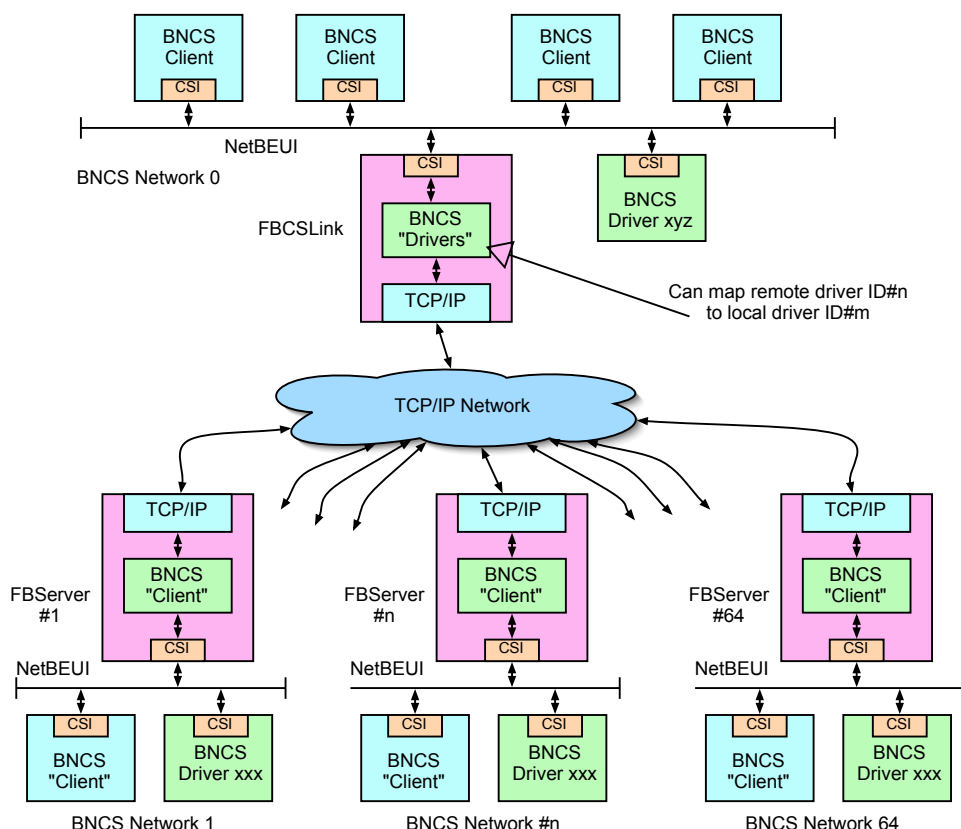
BNCS was originally designed to provide network communications using datagrams on a NetBEUI transport service. NetBEUI offers a fast and simple transport for the control messages that are the important data in a BNCS application. One of the features that makes NetBEUI fast and simple is the low overhead on each packet, but this also means that the datagrams can not be routed into another local area network. This is not a problem for a small, local to one room or building, installation. However, if it is desirable to control a device at a remote location, or to provide control in an enterprise-sized installation, this lack of routing becomes a problem. To get around this problem the BNCS development team adapted a module that was originally developed to manage regional booking information and real time messaging facilities to assist in the distribution of programme material. The name of the module is FABIAN, abbreviated from the full function description of "**F**ast **A**ccess **B**ooking **I**nformation **A**nd **N**otification **S**ystem". FABIAN can use TCP/IP protocol, and this can be both routed between networks in an ethernet based installation, or carried on serial communications circuits (possibly by telephone modem). The diagram below illustrates the use of FABIAN to link two networks.



Reference: Hxxxx

A Fabian TCP/IP Link requires, in addition to a standard BNCS installation, the **FaBian Control System Link** module (**FBCSLink.exe**) and the **FaBian Server** module (**FBServer.exe**). Both FBServer and FBCSLink interface via CSI to their local BNCS environment. FBServer appears to CSI just like any other client and FBCSLink appears like a driver. The FBCSLink and Fabian server will each have two network interface cards, one bound to NetBEUI traffic and the other bound to TCP/IP. Experience shows that having two different types of card in a station simplifies the process of identifying which network physical connection carries NetBEUI and which carries TCP/IP traffic.

A message from a client ApplCore panel in network 1 is passed via CSI to FBCSLink which is configured to redirect commands from the local BNCS environment to a remote Fabian server on BNCS network 2. The FBServer then passes it on to its local CSI for processing. Replies from a driver on network 2 are passed back via CSI to the FBServer, thence back to FBCSLink and in turn to the interested clients in network 1.



FBCSLink can maintain active connections with up to 64 Fabian servers at different sites. Remote device ID's can be translated into different local ID's if there is a conflict between the two areas or the system administrator wishes to logically group the remote sites.

In the event that the network connection between sites is lost FBCSLink will attempt to reconnect, progressively increasing the time between retries, until it attempts every 30 seconds. Connection from FBCSLink to FBServer will be restored automatically when the link itself is restored (within this 30 second window).

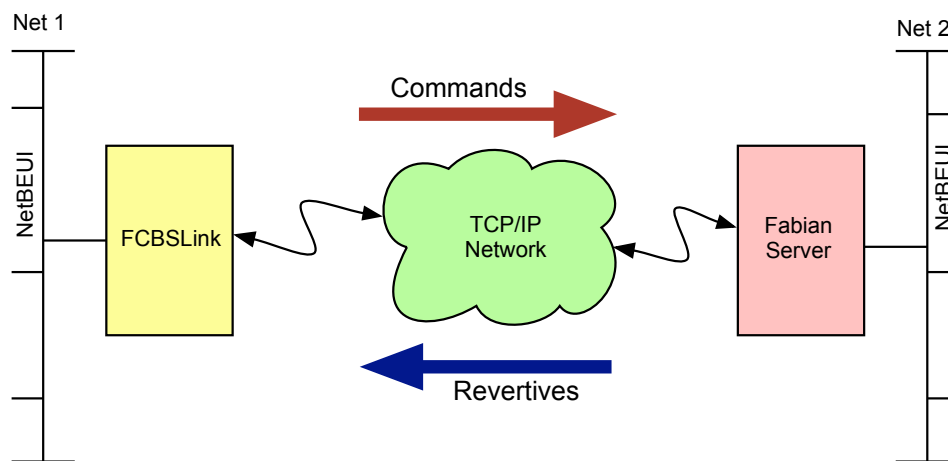
FBCSLink can also be configured to connect to an alternative FBServer. Once connection is lost to the primary server, either because the link has failed or the server itself has failed, then FBCSLink will attempt connection to the 'Alternative' server, the address of which is set in the FBCSLink.ini (initialisation) file. This connection will remain and no attempt will be made to reconnect to the 'first' server, until such time that the link or alternative server itself is lost.

The server requires a user name and password before it will allow access to control functions. FBCSLink.exe looks after the logging on procedure and, if successful, can be made to run a local application.

The BNCS modules, in general, form a versatile toolbox from which users can implement the control systems that are required to support their broadcast installations. The Fabian modules continue and extend this toolbox concept. The next few diagrams illustrate the flexibility of the FCBSlink and Fabian Server.

A Simplex System

A simplex system configuration is ideal for situations where equipment (for example a weather camera) at a remote site (Network 2) needs to be controlled from the local site (Network 1). There is no requirement for control of network 1 equipment from network 2.



The INI file for FCBSlink identifies the TCP/IP address of the Fabian server, provides the user name and password required for access to the server, and the password required to tell the server to pass control messages to its local network. The INI file also identifies the driver, or multiple drivers that are accessed through the Fabian link system. These driver entries include the ability to re-map driver numbers between the local and remote systems. So if the remote system has a driver number 436 that is to be accessed via the Fabian, and there is already a device 436 on the local network, FCBS link allows a new local device_id to be defined, automatically converting accesses to this local driver_id into device 436 when messages are sent over the link.

The INI file for the Fabian server holds the user names and passwords for the stations that are allowed to access the control network. It also identifies the drivers that may receive messages, and restrictions on the destinations that may be controlled through Fabian. The Fabian server can support up to 64 simultaneous accesses from FCBSlinks or other equivalent messaging generators.

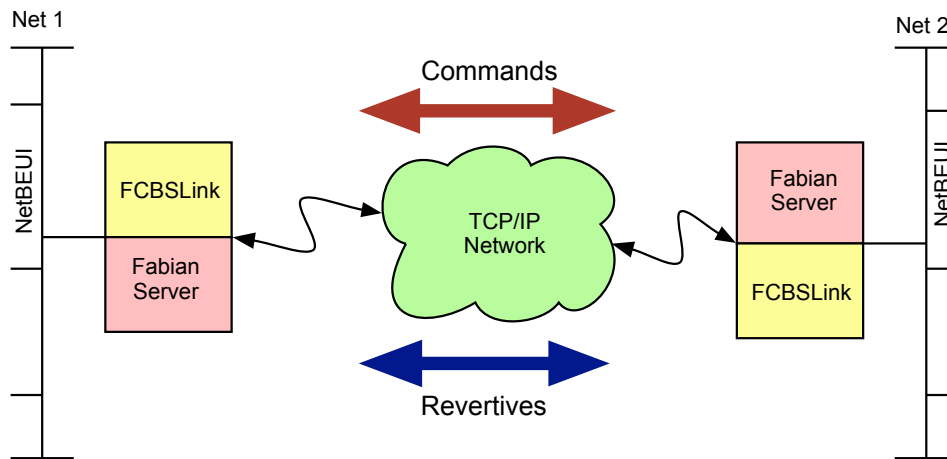
Once the link between FCBSlink and Fabian server is established, the link is continually checked for presence by using a message known as RUT (aRe yoU There). The response from the remote station is a RUR message (aRe yoU there Reply).

The status of the link can be monitored through a read-only infodriver that is provided as part of FCBSlink. The driver number of this infodriver device is provided in the FBCSlink.INI file. sixty four slots are available, with slot 1 issuing status data about link 1, slot 2 about link 2 etc.

A second infodriver is built into FCBSlink to enable the user to control the 64 links. Changing the value of a slot in this second infodriver will start (set slot to 1) or stop (set slot to 0) the associated link traffic.

A Duplex System

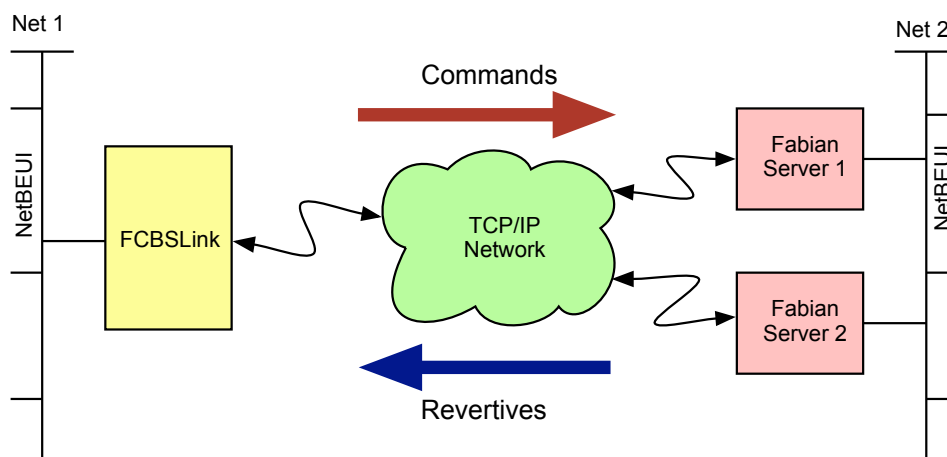
Consider a system that is built from a number of discrete networks. For example it may be sensible to have one network used for the controlled devices in a studio, and a second network for the controlled devices in a central area. This type of network structure can limit the network traffic bandwidth within each area, ensuring minimum latency of command and revertive traffic. However, there will be requirements to pass messages between the networks, with both networks originating traffic. For example, the central area may need to control some crosspoints on the studio router to select cue fees to remote locations, and the studio will need the ability to control crosspoints in the central area matrix to select outside sources to the studio mixer. This is where a duplex Fabian system is used.



Each network has a local FCBSLink and an FCBS server.

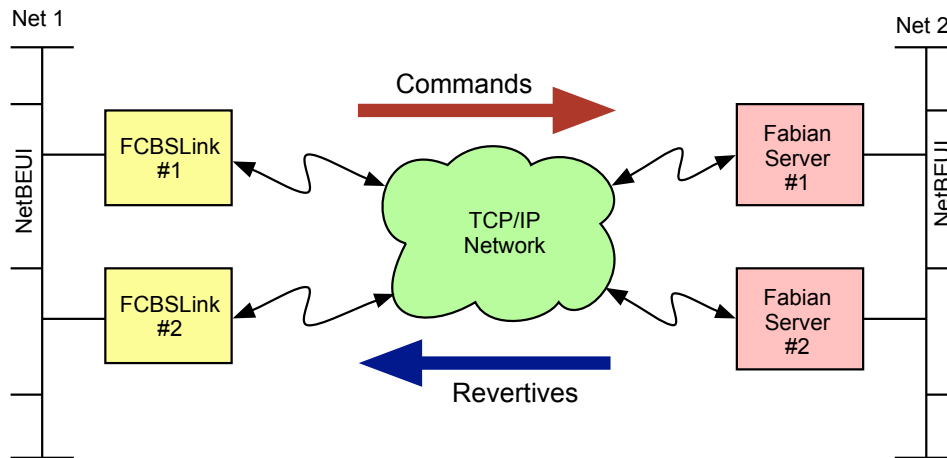
Alternative Fabian Servers

There are some systems that need more security of operation than are supported by the basic system. For those instances it is possible to use a main and reserve Fabian server. The TCP/IP address of the main and reserve systems are identified in the FCBSlink.INI file, as are the user names and passwords for the two Fabian servers. If the primary connection fails, FCBSlink switches to reserve connection. It stays with the reserve connection until such time as that files, when it will once more look to use the primary link.



Limiting single points of failure

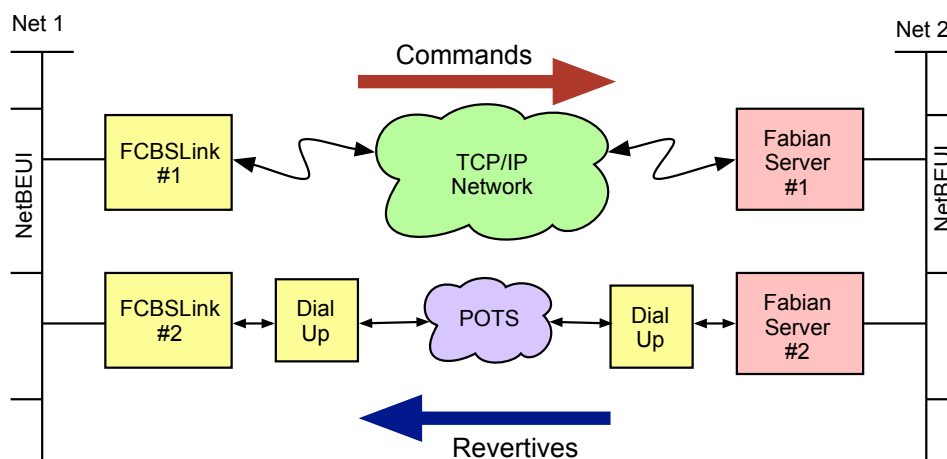
In the previous section there was diversity in the link systems, provided by the two server computers. However, there is still a single point of failure in the single FCBSLink processor in network 1. If this process or its host computer fails, then the control traffic will not get through. To avoid this point of failure it is possible to have two FCBSLink systems in network 1 and two Fabian servers in network 2.



CSI will police FBCSLink so that one instance will be in Tx/Rx and one in Rx only. Both instances of FBCSLink will be configured to use the same pair of Fabian Servers therefore either instance of FBCSLink can talk to either Fabian Server.

Alternative Paths to Fabian Servers

There is still a single point of failure in the TCP/IP network “cloud”. Any problem that stops the cloud from passing error free messages will kill the control traffic. This can be resolved by using two independent transport services, typically by using the ability of TCP/IP to be carried over a serial communications network such as offered by modems over the telephone system (Plain Old Telephone System).



Testing the Link to the Server

The Fabian system uses simple text commands and replies carried across the TCP/IP network. Therefore the server connection to its local network and the connection from a remote FCBS link can be tested by using a Telnet program on an appropriate physical link. To perform the test, you will need to know a user name and passwords for the server to be tested. The names and passwords are listed in the FBServer.ini file. The name and password data will be present for up to 64 users, each section starting in a similar fashion to the example below.

```
[User_001]
UserName=woodnorton
UserPass=training
ControlPass=cntlbncs
Fields=1,2,3,4,5
BNCSAccess=WRITEFILTERED
SendRUTs=1
ExpectRUTreply=0
DevWriteAccess_01=042:6,89-90,22-23
DevWriteAccess_02=200:1-16,21-36
DevWriteAccess_03=201:1-26
.....
```

Open a telnet session to the IP address of the Fabian server. This should return a few lines of text similar to the example below.

```
telnet> open 192.168.2.1
Trying 192.168.2.1...
Connected to 192.168.2.1.
Escape character is '^]'.
*****
FABIAN BNCS Server - V1.11.08 - Dec 6 1999 13:10:29 - (c)BBC 1995-99
WN Fabian 1
Up time = 27/11/2005 - 21:12:09 -- Type 'HELP' for more information
*****
Login:
```

This shows that there is a connection to the Fabian server. Before you can test the access to the control system, you must identify yourself to the server. this is done using the Set User Name (sun), Set User Password (sup), and Set User Control (suc) commands as show below. If your telnet client has local echo enabled, then you will see doubled letters (ssuunn rather than sun).

```
sun  woodnorton
sup  training
Ready...
suc  cntlbncs
Control access granted (Read/Write[Filtered])
```

You can now issue control commands us describe in the later section to prove full control.

This completes the summary of the Fabian system principles. The rest of this document describes the detailed configuration including the settings in FCBSlink.INI and FBserver.INI

FBCS-Link

Introduction

The Fabian Control System Link module **FBCSLINK.EXE** enables a Broadcast Networked Control System (BNCS) installation to control a remote BNCS installation over a TCP/IP link. **FBCSLINK.EXE** becomes a client on a remote Fabian Server. It intercepts switching and interrogation commands on the local BNCS network and, if it is configured to process them, passes the commands on to the server. The Fabian server is itself a client on the remote BNCS network.

FBCSLINK.EXE can maintain active connections with up to 64 Fabian servers at different sites. Remote device ID's can be translated into different local ID's if there is a conflict between the two areas or the system administrator wishes to logically group the remote sites.

In the event that the network connection between sites is lost., **FBCSLINK** will attempt to reconnect, progressively increasing the time between retries, until it attempts every 30 seconds..Connection from **FBCSLINK** to **FBSERVER** will be restored automatically when the link itself is restored.(within this 30 second window).

FBCSLINK.EXE can be configured to connect to an alternative **FBSERVER**. Once connection is lost to the server, either because the link has failed or the server itself has failed, then **FBCSLINK** will attempt connection to the 'Alternative' server.,the address of which is set in the file of **FBCSLINK.INI** This connection will remain and no attempt will be made to reconnect to the 'first' server, until such time that the link or alternative server itself is lost.

The server requires a user name and password before it will allow access to either booking or control functions. **FBCSLINK.EXE** looks after the logging on procedure and if successful can be made to run a local application.

Configuration

When **FBCSLINK.EXE** is started it looks for a file in the Windows directory called **FBCSLINK.INI**. If not found the file is created. The [Configuration] section will contain the following default values.

[Configuration]

Default	Example	Explanation
DebugMode=0	DebugMode=0	1 = Debug messaging enabled.All telnet messages between client and server are listed. 0 = Debug messaging disabled, concise startup, link status and close down messages shown.
Send Ruts=1	Send Ruts=1	1 = Sent and received RUTS shown in the debug window. 0 = No RUTS shown in the debug window.
InfoDriver=0	InfoDriver=0	Device id for built-in link status Infodriver. Driver will issue link status reverts from Slots 1 - 64 (1 slot per server).

Default	Example	Explanation
ActiveInfoDriver=0	ActiveInfoDriver=0	Device id for in built <i>control</i> infodriver. Slots 1 - 64 are used to control the link to each server. 1 = activate link 0 = deactivate link
AlertOnError=0	AlertOnError=0	1 = FBCSLink will display a message box will pop when a successful connection is made or a disconnection occurs. 0 = the message appears in the scrolling window. Note this stops the link execution until OK is pressed!
QuitOnLinkFail=0	QuitOnLinkFail=0	If set to 1, the application will close when connection to the server is lost.
CacheEnable=0	CacheEnable=0	This is a hang over from BNCS Version 1. FBCSLink now uses the CSI Cache.
CacheFilePath=C:\	CacheFilePath=C:\	See Above
UpdateDatabases=0	UpdateDatabases=0	1=Database service enabled. FBCSLink will take each redirection driver ID in turn and query a source/ destination once per second. If there is a discrepancy between the string returned and the one in the local system's database then the local system will be updated. 0=database service disabled.
CheckLinksByConnectionNumber=0	CheckLinksByConnectionNumber=0	Used when dual links are required for redundancy, and prevents both links being opened simultaneously. 0 = dual links are used to the same server. The reserve link will check whether the main link is active using the server IP address in its FBCSLink.ini file. If it is active the reserve link will not be established. This also works if the main link has been established to an alternative server using the AltAddress entry. 1 = dual links are used to 2 servers with different IP addresses. The reserve link does not have the main server IP address in its FBCSLink.ini file and so cannot inquire whether the link to the main server has been established. It therefore checks whether the link is active via the connection number [the list of servers to connect to] in its FBCSLink.ini file. Therefore there needs to be an entry for both servers in the FBCSLink.ini file.
AlwaysRegisterMaxIndices=0	AlwaysRegisterMaxIndices=0	A hang over from Version 1 InfoDrivers where the number of slots was allocated manually rather than fixed at 4096 as in V2. 1 = Fabian assumes the infodriver has 4096 slots.

After the configuration section there will be 64 sections labelled [Fabian_Server_nn] where 'nn' is between 1 and 64. Each section has the following default entries.

[Fabian_Server_nn]

Default	Example	Explanation
Name=NONE	Name=Manchester CTA	The name of the remote server or site. This is used in messages to the user regarding the status of a link.
Address=NONE	Address=110.192.113.123	The IP address of the Fabian Server
AltAddress=NONE	AltAddress=110.192.113.124	The IP address of an alternative Fabian Server which is used if the connection cannot be made to the first Fabian Server. FBCSLink will try both servers alternately until it makes a connection.
ProxyAddress=NONE	ProxyAddress=185.143.150.180	The IP address of the Proxy Server if the link goes via one.
ProxyPrompt=NONE	ProxyPrompt=Simon	When FBCSLink attempts to establish a Telnet session with the Proxy Server, the Proxy will return a series of text messages in response. FBCSLink will wait until the Proxy returns the characters specified here before it attempts a Telnet Login. This acts as a prompt that the server is ready to Telnet. When using the BBC Reith network the prompt is 'Simon'.
ProxyCommand=NONE	ProxyCommand=c<space>	Pre-fix at the start of the IP address so that the Proxy Server will accept the Server IP address. For the BBC Reith proxy this is 'c<space>
Port=23	Port=401	The default port address is the Telnet port (23). If this is already in use the recommended alternative is 401
UserName=User	UserName=MRREMOTE	Your user name on the remote server
UserPass=Password	UserPass=MRPASS	Your password on the remote server
ControlPass=Pass2	ControlPass=MRCTRL	A secondary password required before the remote server will allow you access to control functions
LogonExec=NONE	Logon Exec=Launch.exe	This is the application you wish to run as soon as FBCSLink has successfully connected to the remote server. e.g. Launch.exe
ExpectRUTreply=0	ExpectRUTreply=0	1 = FBCSLink will drop the connection if it does not receive a RUR from the server in response to its RUT.
BNCSRevertiveMode=Normal	BNCSRevertiveMode=Normal	Normal = revertives are verbose text via Telnet Packet = revertives sent as data packets (quicker - especially when polling a large router for example).
BiDirectionalLink=0	BiDirectionalLink=0	Set to 1 to prevent database howl rounds when a bi-directional link is used (FBServer and FBCSLink at each end of the link). See the section 'Database Changes via Fabian' for more details.
LinkActive=1	LinkActive=1	FBCSLink will attempt to open the link to the Server at startup. A link will be made to all servers listed in the FBCSLink.ini file where LinkActive=1. If the link state is changed using the Active InfoDriver slots, the LinkActive setting is changed to reflect the new state.
ConnectionOverride=0	ConnectionOverride=0	1 = Override switching between main and alternate servers.

Default	Example	Explanation
LockTimeToRemoteSystem=0	LockTimeToRemoteSystem=0	I = FBCSLink will periodically obtain the system time from the remote system and will synchronise the time on the FBCSLink CSI.
Devices=16	Devices=9	The number of remote devices that you wish to have access to. A corresponding number of device entries will be added to FBCSLink.ini
Device_nnn=Device ID, Local ID, Remote ID	Device_001=GRD,100,200	This would map a GRD which has the ID of 100 on the local network to a remote GRD which has the ID of 200 on its network. If no mapping is required i.e. by addressing local Device 100 you require to address remote Device 100 then make both Device numbers the same e.g. Device_001=GRD,100,100 To map an InfoDriver then substitute GRD with ID. To map a GPI then substitute GRD for GPID.

Next, FBCSLink will look for Dev_nnn.ini files for all the devices in the 'Device_nnn=...' list. If any of these Dev_nnn.ini files for the remote devices **are not found** on the FBCSLink machine a warning message will be given and those devices **will be ignored**. The ini files are required so that the maximum number of destinations, inputs/outputs or info slots can be read. FBCSLink needs this information so that it can maintain an internal tally table for each device.

Note: if the device that the dev_nnn.ini relates to is mapped via FBCSLink then the dev_nnn.ini file should have the Local ID.

FBCSLink maintains an internal tally table for each remote device. When a control panel polls for status FBCSLink will attempt to supply the information from its own tables. When a panel polls for the first time after the link is established FBCSLink will poll the remote driver via the Fabian Server. The revertives are stored and subsequent polls will be met by FBCSLink from its internal tables. The error corrected TCP/IP link should ensure that the local tables are kept up to date and data traffic on the link is kept to a minimum.

Setting Up a Link

Establish the TCP/IP link before running FBCSLink.exe which will then make up to 5 attempts to contact each of the remote servers. Connections should occur on the first try, but occasionally some settling time is required after the TCP/IP link is established.

Note. If Alternative Servers have been specified in the FBCSLink.ini file then the application will attempt to connect to the main first, and if connection cannot be obtained will attempt to connect to the alternative. If the 'alternative server' cannot be found then FBCSLink will try once again to connect to the main server. This process will continue until a successful connection is obtained.

Link Control

FBCSLink has two in built InfoDrivers (InfoDriver and ActiveInfoDriver) which allow control of the links to the Fabian Servers and provide link status information. The driver ID's are supplied in FBCSLink.ini.

InfoDriver

This InfoDriver shows the link status for each of the 64 possible servers in slots 1 to 64. The slot values can be:

- 0 = not connected
- 1 = connected
- 2 = Logged on
- 3 = inactive
- 4 = active

ActiveInfoDriver

This InfoDriver controls the connections to the Fabian Servers:

1. Server Section.

BNCSLink can connect to up to 64 servers. The link to each can be controlled by the 'LinkActive' entries in the FBCSLink.ini file which determines which servers FBCSLink will attempt to connect to at startup. Each link can be actively controlled by setting slots 1 to 64 in the InfoDriver whose ID is given by the 'ActiveInfoDriver' setting in FBCSLink.ini.

2. Main/ Alternative Address switching.

Each link can have a specified alternative server. Rules governing which server is selected when FBCSLink attempts to make a connection are set by using a slot in the range 1001 to 1064 of the ActiveInfoDriver.

0 = FBCSLink will toggle between server addresses until it can make a connection to one or other.

1 = FBCSLink will always attempt to connect to the main server address and toggling is inhibited.

2 = FBCSLink will always attempt to connect to the alternate server address and toggling is inhibited.

3. Main/Alternate Address indication.

Slots 2001-2064 of the ActiveInfoDriver indicate whether the main or alternate address is being used for each server connection.

1 = FBCSLink has connected to the main server address.

2 = FBCSLink has connected to the alternate server address.

Device Databases

Device Database Synchronisation via Fabian.

Database files for drivers on the remote network are usually also present on the local machines to allow local client panels to use the names they contain. If any of these databases are changed on the remote network, a mechanism must exist to keep the local databases in step with any changes on the remote system. If 'Update Databases' is set to '1' in FBCSLink.ini, FBCSLink will send a ARN (Appcore Router Name) command at regular intervals to the remote Fbserver for each database entry in all databases of the redirected drivers. The remote FBServer then responds with the correct name for the database entry. This process cycles round all the databases as long as the link is active. A by-product of this is that there may be a time delay between the database change on the remote system occurring and the local database receiving the new name.

Device Database Changes via Fabian

It is sometimes necessary for a local client panel to change a database entry for a device on a remote network via Fabian. When a client panel on the local network issues a Router Modify command it is picked up by all CSI's on the network. Any CSI that has the device file on its workstations will modify the database entry and will issue a 'database change' message to any clients it is hosting so that they can update as required by calling stringtable line 32000.

When a Router Modify is issued on a local network for a device database on a remote network, FBCLink must pick up this instruction and pass it to the remote network. FBCLink does not connect directly to the local network (it connects via CSI) so it cannot see the Router Modify instruction. Instead it picks it up indirectly by using the database change message issued by its CSI when a database change occurs. This means that the remote devices ini file must exist on the local FBCLink workstation to allow CSI to modify it and subsequently issue the database change message that FBCLink picks up**. This is then passed by FBCLink to the remote network where it is issued by FBServer as the appropriate Router Modify command.

*** remote database files are normally also present on the local machines to allow panels etc. to use the names in the databases.*

Database Howl round

A database howl round can occur when there is a bi-directional Fabian link between the two networks (i.e FBCLink and FBServer are both running on each of the local and remote networks). When a remote database change is initiated as described above, the remote FBServer issues the Router Modify. This is picked up by the CSI hosting FBCLink on the remote site which causes this FBCLink to pass the database change information back to the local network via the second link where it is issued by the local FBServer as a router modify, the whole cycle then repeats itself and a howl round ensues. This loop is broken by setting 'Bidirectional Link = 1' on the remote FBCLink.ini.

FABIAN RUT's

RUT = aRe yoU There

RUT's are simple messages that are sent between server and client at 20 second intervals to check that the link is still active.

FBCLink

FBCLink can send RUT's to FBServer. This is controlled by the SendRuts entry in FBCLINK.ini. The server will respond with a **RUR** (RUT Reply).

The reasons for this are:

1. Check that the link to the server is still OK.
2. Ensure that the TCP/IP ARP cache does not lose the address of the server. This can happen if there is very little activity on the link. An example of this might be Fabian control of a router where crosspoints are only changed very infrequently.

FBCLink.ini has an entry ExpectRutReply.

If this is set to '0' FBCLink will not close the link if it doesn't receive a reply from the Fabian Server.

FBServer

FBServer can also send RUTS to each of its clients, which will respond with an RUR if the link is active. This is controlled by the SendRuts and ExpectRutReply settings for each user in FBServer.ini

The main reason for this is to ensure that FBServer closes the port associated with a client if the link is broken. If this does not happen, the port will remain allocated. Server RUTS can be inconvenient if connecting to a Fabian server via Telnet, as FBServer will close the link if it does not receive RUR responses to its RUT's - this would involve sending RUR's manually via Telnet.

FB-Server

Introduction

The name FABIAN comes from its original intended function as a Fast Access Booking Information And Notification System supplying regional booking information and real time messaging facilities to assist in the distribution of programme material. Its main purpose now is to act as a means of linking BNCS control system networks.

Terminology

Throughout this section of the document:

<CR> means press carriage return.

Text in ***bold italics*** is either output or input from the user screen

Server Overview

The FABIAN Server is the hub of the system and performs 4 basic functions.

- TCP/IP link
- Database Engine
- Resource Manager
- Message Handler

TCP/IP Link

The server can manage clients wishing to access to the BNCS network (cluster) from another BNCS cluster or simple terminal emulation program. Device Id mapping can be used to prevent clashes between local and remote devices.

Database Engine

The database engine performs basic storage and retrieval of database records and fields. It has a capacity to store and manipulate a whole years bookings and associated information. There can be up to 999 bookings per day with up to 99 amendments per booking. All amendments are retained. A database record consists of 128 characters fields. Each field can be up to 128 characters in length. In addition to the fixed fields the database engine can store and retrieve text files up to 32k in length.

Resource Manager

The resource manager is a system for managing up to 26000 named resources and allocating them to bookings. Possible conflicts between resources and bookings can be prevented. Temporary resources are allowed.

Message Handler

A client can subscribe to up to 99 message notification groups. These are communication channels for inter-client messaging. Some channels are dedicated to particular functions such as AMS data, others are entirely user configured.

Mirror Servers

A server can optionally be set up to mirror another server. It logs on to the remote server and receives database change information which it uses to interrogate the remote machine. The local server overwrites bookings in the local database or inserts them at a predefined offset. Details on the configuration of the mirroring process is detailed in the following section.

Server Configuration

Setting Up

Assign TCP/IP protocol to your Network Card and specify an IP address. (This will have to be given by your system co-ordinator)

If you want the server to act as a TCP/IP link you will need to install a NetBIOS transport protocol stack in addition to the TCP/IP.

It is a good idea to set up a client machine with the above software at this time so that you can test the basic connectivity of the network using the TCP/IP 'Ping' utility. When you are happy that the machines are configured correctly and are able to 'ping' each machine from the other you can then proceed with the rest of the installation.

Install BNCS on the PC you wish to run FBserver on. Locate **FBserver.exe** which should be in `C:\bncs\bin\fabian` directory if you have set up your machine using the BNCS install disc. If not, it is available on the BNCS FTP site. Run CSI and then run **FBServer.exe** which will then create **FBserver.ini** file in the Windows/ WINNT directory (depending on the operating system you are using). Now you will now need to edit **FBServer.ini**, using a text editor such as Notepad, as detailed below.

The FBServer.ini file

The INI file is split into several sections. The first section contains general configuration information:-

[Configuration]

Default	Example	Explanation
ServerName=Anonymous	ServerName=Anonymous	Server name - displayed by any client or telnet sessions e.g. Exeter Local Radio
ServerPort=23	ServerPort=23	This is the TCP/IP port that clients must use in order to talk to the server. A PC may have several TCP/IP services of different types running and each one has its own unique port number. The Fabian server uses port 23 by default, which is also the telnet port number. If this is already in use on the PC you wish to run the Fabian server on then a good alternative is port 401.
LoginRequired=1	LoginRequired=1	If set to '0' no username or password is required.
DataPath=C:	DataPath=C:\FBSERVER	The path to the log files
CloseWithCSI=1	CloseWithCSI=1	Closes when CSI closes
DisableCSICache=1	DisableCSICache=1	Forces CSI cache off so that poll requests are always routed on to the network.
EnableLogging=1	EnableLogging=1	Opens hourly log files of link activity.
DebugMode=0	DebugMode=0	When set to '1' the server displays more verbose information about its activities. This flag is toggled by the 'Diagnostics' option on the main menu.
SystemCode=0	SystemCode=0	A system password to allow remote supervisor access. This is not currently used

Default	Example	Explanation
BNCSAccess=I	BNCSAccess=I	If set to 'I' the server will act as a BNCS control system host and allow 'ApplCore' style commands to be used. Note :A second password is required via the SUC command in order gain access and CSI.EXE must be running prior to running the server.
SchedNotifyGroup=I	SchedNotifyGroup=I	Used with Fabian Scheduler to set schedule group notification of changes.

The following sections of the file contain user details for up to 64 users or automatic clients. Each of the sections has the same layout:-

[User_001]

Default	Example	Explanation
UserName=---	UserName= yourname	User name (16 characters max)
UserPass=---	UserPass=vectra	User password (8 characters max)
ControlPass=---	ControlPass=fluff	Control password (8 characters max)
Fields=1,2,6,3,4,5	Fields=1,2,6,3,4,5	The fields that this user wishes to use in required order.
BNCSAccess=READONLY	BNCSAccess=READONLY	May be READONLY, WRITEFILTERED OR WRITEALL If READONLY - you will have no control of the devices on the FABIAN Server network. If WRITEFILTERED - the device access table below is used specify the device(s) & index range to be controlled. If WRITEALL - full control of all devices on the FABIAN Server network.
Fields=1,2,3,4,5	Fields=1,2,3,4,5	Fields=1,2,3,4,5 The Server database entries this user wishes to use, in order - see Database below.
SendRUTs=I	SendRUTs=I	Sends RUT's (aRe yoU There) to clients every 60 seconds
ExpectRUTreply=0	ExpectRUTreply=0	Expects RUT replies from clients - Disables the link if replies are lost - set to 0 for a BNCSlink client login.
DevWriteAccess_01=NONE	DevWriteAccess_01=042:6,89-90,22-23	Table of devices and indexes that may be accessed under READ & WRITEFILTERED
DevWriteAccess_02=NONE	DevWriteAccess_02=200:1-16,21-36	Device 200, indices 1-16 & 21-36 (GPI's)
DevWriteAccess_03=NONE	DevWriteAccess_03=201:1-26	Device 201 indices 1-26 (Infodriver)
DevWriteAccess_04=NONE	DevWriteAccess_04=5:16	Device 5 Destination 16 (Router)
DevWriteAccess_05=NONE	DevWriteAccess_05=103:9,11,13	...

Default	Example	Explanation
DevWriteAccess_06=NONE	DevWriteAccess_06=104:1-128	...
etc ..up to DevWriteAccess_16	etc.	etc.

The next section is the database configuration and determines the field names and sizes for up to 128 fields. The first 7 fields of a typical configuration is shown below.

[Database]

Field_001=Booking,3	Each entry consists of the field name and its size.
Field_002=Amendment,2	
Field_003=LineUp,4	
Field_004=Start,4	
Field_005=End,4	
Field_006=Status,2	
Field_007=Title,64	
etc.	

The last section is for the optional configuration of the server mirroring mechanism which is used to mirror the bookings database information. This is not relevant if you are using FBServer as a TCP/IP link but is shown here for completeness. For more information about the FBServer bookings facility, lookup the documentation on Fabian Booking process.

[Mirror]

MirrorEnable=1	'1' enables server mirroring, '0' turns it off.
Address=132.185.203.1	The IP address of the remote server to be mirrored
Port=23	The service port on the remote server. The default is 23
UserName=leedsmirror	Username to use on the remote server
UserPass=leedscta	Password to go with the above user name.
RemoteFields=1,2,3,4,5,6,7	The fields in the remote database that you are interested in.
LocalFields=1,5,6,7,8,9,10	The fields in the local database where the booking information is to be placed. For example data in remote field 3 would be put into local field 6.
Mode=INSERT	There are two Mode options. In OVERWRITE mode bookings in the local database are overwritten by those obtained from the remote database. In INSERT mode the bookings are inserted as the next free booking starting at the offset described below.
InsertOffset=600	In INSERT mode the local server will look for a free entry in the local database starting at InsertOffset and insert the booking there.

BNCSLink

Description

`bncslink.exe` was written so that it would be possible to run a simple BNCS panel application on a BBC "Desktopped" machine without complicated set-up.

This module is a simple FABIAN client application that is capable of supporting a single applcore (or 3rd party) panel. It replaces CSI as the interface between the panel and the network.

Features and Limitations

This application requires no special PC set-up at all and will run from one directory, or even off a floppy (this assumes that the PC is already set-up with Windows 95 and has a TCP/IP connection to a FABIAN server - any BBC desktopped machine that can see the BBC intranet). The program could even be run off a file server and the only thing copied to the local machine are skeleton `csi.ini` and `applcore.ini` files.

Both version 1 (`grd_xxx.ini`) and "version 2" (`dev_xxx.ini`) database files are supported. If both `grd_xxx.ini` and `dev_xxx.ini` exist for the same device number, the `dev_xxx.ini` file will be ignored.

This module has a few limitations.

Only one client application (panel) is supported. If the application calls for many panels it is probably too complex and so should be using a full BNCS installation.

There is no cache. Poll requests are passed directly to and revertives come directly from the FABIAN server.

Databases are not loaded into memory. Any database name is read from a compiled ini file each time (of the same format as the files CSI uses). In practice this has no serious hit on performance for the sort of simple panels that have been used so far.

Only databases 0 and 1 are supported.

With a client that is not permanently connected it is very possible that the databases will be out of date. This module will go round each of the databases that it uses and gradually update them - this does however mean that it has to be connected for sufficiently long to be able to do this. The length of time it takes to do this is dependant entirely upon the number and size of the databases used. All databases in the same directory as the application will be updated. Updating databases requires `fbserver.exe` v1.11.01 or later

If this client loses contact with the FABIAN server it displays a message box and exits. To reconnect the user must restart the program.

N.B. BNCSlink does not respond to RUT's, so disable `ExpectRUTreply` on the relevant login on the Fabian server by setting `ExpectRUTreply=0`.

On errors, the program displays a message box explaining the error and then exits (e.g. write access not enabled)

Set up

[Login]

Entry	Default	Notes
Username		User name on the Fabian server
Password		Password on the Fabian server
ControlPass		Control Password

[Panel]

Entry	Default	Notes
Path		Path to application to be run once connected to the FABIAN server. This parameter can also be supplied as a command line parameter (if a command line parameter is supplied this ini file setting is ignored).
DeviceWriteEnable	ALL	This can limit which devices this module will try and write to. This is a comma delimited list of device numbers or ALL for all devices

[Server]

Entry	Default	Notes
Address		IP address in the format 132.185.177.75

Required Files

The following files are required for the BNCSLink program itself (can be in the same directory as the application):

bbc_cc.dll

bwcc.dll

bncslink.ini must be in the same directory as the application

Database files (grd_XXX.ini or dev_XXX.ini) files should be ***IN THE SAME DIRECTORY AS THE APPLICATION***.

Additionally, Applcore requires the following files (can be in the same directory as the application):

aaplay.dll	Note: This is not required for panels that are bound to versions of ApplCore 2.20.14 (11/01/00) or greater
m2c.dll	Note: This is not required for panels that are bound to versions of ApplCore 2.01.00 (19/07/99) or greater

The following files are required in c:\windows. Skeleton files are created when this program is first run if they do not exist.

csi.ini
applcore.ini

Windows NT

This program will work under Windows NT from versions 1.03. Note: the file m2c.dll is not compatible with Windows NT. To use applcore panels on NT copy aaplay.dll, and rename the copy m2c.dll. This is sufficient to keep applcore happy.

Notes

This is a 16 bit application, and must be run from a "16 bit" path. i.e.

c:\somedir\bnclink.exe is OK,

but

\\rc456\c\somedir\bnclink.exe will not work

Upgrading to Windows 2000 and XP

There was a update applied to the BNCSlink.exe on 4th June 2003 to enable it to run correctly on Windows 2000 and Windows XP. If you are migrating your system from an unsupported operating system make sure you have the latest version of BNCSLink.exe which can be found on the BNCS FTP site - Please note BNCS User Name and Password required to access this link.

Fabian Command Reference

All commands consist of three letters followed by zero or more, space delimited, command dependent parameters. Valid commands with correct syntax are not acknowledged unless the nature of the command returns some output. Unknown commands result in an appropriate error message.

AppCore/BNCS Commands

The following commands are provided to control a BNCS environment via the Fabian server using AppCore style panel commands. Router, GPI and InfoDriver control is supported. An additional password is required before control system access is permitted.

GPI File

Syntax : AGF <Driver%>

Instructs the GPI driver <Driver%> to reload its Line Inversion table from the drivers INI file. Any changes you have made to any of the inversion bits will be immediately reflected in the system. In other words if you have changed the inversion bit for an i/o line then as soon as the table is reloaded the logical state of that line will change and the driver will reflect that change in state by sending a message to the network.

GPI Lock

Syntax : AGL <Driver%> <Min%> <Max%> <Switch%>

If <Switch%> = 1 then all outputs between <Min%> and <Max%> are locked and cannot be toggled again without first unlocking them with <Switch%> = 0 or changing the lock state manually at the driver.

GPI Poll

Syntax : AGP <Driver%> <Min%> <Max%>

Polls <Driver%> for status on all inputs or outputs between <Min%> and <Max%>. The information will be delivered into the stringtable offset that was set up when the the GPI Register command was executed.

GPI Register

Syntax : AGR <Driver%> <Min%> <Max%> <Stringtable Offset%> <Offset Mode%>

Registers with a GPI driver <Driver%>. Update information will be sent to your application every time the status of an input or output between <Min%> and <Max%> changes. The status of an input or output will be placed in S% and stringtable offset <Stringtable Offset%> executed if <OffsetMode%> is 0. If <OffsetMode%> is 1 then <Stringtable Offset%> is added to the input or output index and the resulting stringtable line is executed. For more information see the AppCore Guide.

GPI Switch

Syntax : AGS <Driver%> <Index%> <0 or 1>

Turns the given <Index%> on the GPI either on or off providing it has been configured at the driver as an output. <Driver%> is the number of the GPI you want to use.

GPI Unregister

Syntax : AGU<Driver%>

Following this command no further status updates will be returned to the application for the <Driver%>.

Info File

Syntax : AIF <Driver%> <Switch%> <Filename\$> <Var\$>

Requests <Driver%> to send the string <Var\$> to the file <Filename\$>. If <Switch%> is 0 the string will be added to the end of <Filename\$> if it exists, otherwise it will be created. If <Switch%> is 1 a new file will be created with the name <Filename\$>. Any previous file with the same name will be overwritten.

Info Hardcopy

Syntax : AIH <Driver%> <Var\$> <Switch%>

Requests <Driver%> to print <Var\$> on the logging printer. If <Switch%> is 0 the string will be added to a buffer. When sufficient lines of strings are held in the buffer to fill a page then the whole buffer will be sent to the printer. If <Switch%> is 1 it forces the buffer to be sent to the printer regardless of the number of lines it contains.

Info Lock

Syntax : AIL <Driver%> <Min%> <Max%> <Var%>

If <Var%> is 1 then all slots on <Driver%> between <Min%> and <Max%> are locked. The contents of a slot can not be changed when it is locked. Use <Var%> set to 0 to unlock a slot or range of slots.

Info Poll

Syntax : AIP <Driver%> <Min%> <Max%>

Requests <Driver%> to send all information contained between slots <Min%> and <Max%>.

Info Register

Syntax : AIR <Driver%> <Min%> <Max%> <Offset%> <Offset Mode%>

Requests the <Driver%> to register the calling application to be informed whenever strings in slots between <Min%> and <Max%> change. The information in a slot will be placed in S\$ and stringtable offset <Stringtable Offset%> executed if <OffsetMode%> is 0. If <OffsetMode%> is 1 then <Stringtable Offset%> is added to the input or output index and the resulting stringtable line is executed.

Info Unregister

Syntax : AIU <Driver%>

Requests <Driver%> to cease sending update information.

Info Write

Syntax : AIW <Driver%> <Var\$> <Index%>

Requests the <Driver%> to store the string <Var\$> in <Index%>.

Router Crosspoint

Syntax : ARC <Driver%> <Source%> <Destination%> [<Mask\$>]

Instructs a router driver <Driver%> to route <Source%> to <Destination%>.

Router Database

Syntax : ARD <DriverVar%> <Source/DestFlag%> <Index%>

When database change occurs as a result of the 'Router Modify' command the router driver will validate the source or destination name that has changed. If valid the change will be signalled to all instances of CSI in the system. Each CSI will save the change in the local database and signal all panel applications that a change has occurred by executing the DATABASE line of the stringtable. The DATABASE line is 32000.

To extract details of the change you must execute a Router Database command on the DATABASE line and supply variables in which to receive the change information. For example :-

RD A% B% C%

Immediately after executing this command A% will hold the driver number whose database has changed. B% will be '0' if the change is a source and '1' for a destination. C% will hold the index value for the source or destination that has changed. You can then use this information to extract the new database names if the change is applicable to your panel. The DATABASE line is executed once for every single database change.

Router Index

Syntax : ARI <Driver%> <Srce/Dest Switch%> <Srce/Dest\$> <Var%>

Returns the Index for a source or destination on <Driver%>. If the name given is a source then the <Source/Dest Switch%> must be 0 and for a destination it must be 1. <Var%> is the variable in which the index value is to be placed.

Router Modify

Syntax : ARM <Driver%> <Source/Dest Switch%> <Index%> <Name\$> <Flag%>

Instructs router driver <Driver%> to change its database. If <Source/Dest Switch%> is 0 then it is a source name change. If it is 1 then a destination is being changed. <Index%> is the index of the source or destination name that is changing and <Name\$> is the new name.

If <Flag%> is 1 it signals to the router driver to send a complete tally table dump in order to update any revertive source names that may have changed. If several database changes are being made this flag should be set to '0' and only be set to '1' for the very last command. If only destination names are changing then no revertive update is needed.

Router Name

Syntax : ARN <Driver%> <Source/Dest Switch%> <Index%> <Var\$>

Returns the name of a given source or destination <Index%> on a <Driver%>. The <Source/Dest Switch%> is 0 if searching for a source and 1 for a destination. If the name is found it placed in <Var\$>.

Router Poll

Syntax : ARP <Driver%> <Min%> <Max%>

Polls <Driver%> for the status of all destinations between <Min%> and <Max%>. The information will be delivered into the stringtable offset that was set up when the Router Registration command was executed. S% will contain the source number and S\$ will contain the name of the source as defined in the INI file for that router driver.

Router Register

Syntax : ARR <Driver%> <Min%> <Max%> <Offset%> <Offset Mode%>

Registers with router driver <Driver%>. Revertive information will be sent to your application every time the status of a destination between <Min%> and <Max%> changes. The source index will be placed in S% and stringtable offset <Stringtable Offset%> executed if <OffsetMode%> is 0. If <OffsetMode%> is 1 then <Stringtable Offset%> is added to the input or output index and the resulting stringtable line is executed. For more information see the ApplCore Guide.

Router Unregister

Syntax : ARU <Driver%>

Following this command no further status updates will be returned to the application for the Router

Get Commands

Get Current Amendment

Syntax : GCA

Returns the currently set amendment number. Valid number are between 0 and 99.

Get Current Booking

Syntax : GCB

Returns the currently set booking number. Valid number are between 1 and 999.

Get Current Day

Syntax : GCD

Returns the currently set day number. Valid number are between 1 and 31.

Get Current Field

Syntax : GCF

Returns the currently set field number. Valid numbers are between 1 and 128.

Get Current Month

Syntax : GCM

Returns the currently set month number. Valid number are between 1 and 12.

Get Current Year

Syntax : GCY

Returns the currently set year number. Valid number are between 1990 and 3000.

Get Database Extension

Syntax : GDE <Filename\$>

If its required to store data that exceeds the maximum field size of 128 characters it is possible to save the data to a file. The filename can, if desired, be stored for future reference in a field in the database using the 'Set Database Field' function. The files are stored by the server in the directory pointed to by the currently set month and day. Therefore the same filename can be used every day if required. The <Filename\$> parameter may be a complete path from the servers base directory if you wish to save the file elsewhere, but the path must be valid.

Get Database Field

Syntax : GDF

Returns the contents of the field pointed to by the currently set month, day, booking, amendment and field parameters.

Get Database Info

Syntax : GDI

Returns the status of the database for the currently set month and day. This is simply the number of bookings at present, but may be expanded if more information is required.

Get Database Lock

Syntax : GDL

Returns the contents of the field pointed to by the currently set month, day, booking, amendment and field parameters.

Get Database Owner

Syntax : GDO

Returns the the name of the client who has locked out the booking pointed to by the currently set month, day and booking.

Get Database Record

Syntax : GDR

Returns the the database record pointed to by the currently set month, day, booking and amendent. The fields returned will be those configured using the Set User Fields command. An error will be generated if no user fields are configured

Get Database Update

Syntax : GNA

Returns the state of the database update flag. When this flag is set to 1 the server will send a message to the client whenever a booking changes. If set to 0 no such notification will be sent. The default is 1.

Get Next Amendment

Syntax : GNA

Returns the next amendment number available after the currently set amendment number for the currently set booking number. If no more amendments exist then nd error message is returned.

Get Next Booking

Syntax : GNB

Returns the next booking number available after the currently set booking number. If no more bookings exist then an error message is returned.

Get Next Free

Syntax : GNF

Returns the next free booking number after the one currently set.

Get Resource Commitment

Syntax : GRC <Resource\$> <Start\$> <End\$>

List all the commitments for <Resource\$> for the currently set month and day between <Start\$> and <End\$>. Partial overlaps are included.

Get Resource Booking

Syntax : GRB

List all the resources allocated to the currently selected booking.

Get Resource List

Syntax : GRL <FirstLetter\$>

Returns a list of all resource names beginning with <FirstLetter\$>.

Get Resource Update

Syntax : GRU

Returns the state of the resource update flag. When this flag is set to 1 the server will send a message to the client whenever a resource commitment changes. If set to 0 no such notification will be sent. The default is 1.

Get System Help

Syntax : GSH <Optional\$>

This command is also invoked if the user types 'HELP'. If no further parameters are specified it returns the contents of the file FBSH_HLP.TXT in the servers default directory. If 'HELP COMMANDS' is specified it returns the contents of 'FBSH_COM.TXT'. If 'HELP GROUPS' is specified it returns the contents of 'FBSH_GRP.TXT'.

Get System Info

Syntax : GSI

Returns the banner page that a TELNET user sees when first connecting to the server.

Get System Time

Syntax : GST

Returns the system time at the server.

Get System Users

Syntax : GSU

Returns a list of connected users/clients. The list contains the user name, IP address and remote port number.

Get User Echo

Syntax : GUE

If the user terminal echo flag is turned on this command will return 1, otherwise it returns 0. The user terminal echo flag determines whether the server echos back a clients commands as they are typed in. Default is on (1).

Get User Fields

Syntax : GUF

Returns a comma delimited list of the fields required by the client. This set by the Set User Fields command. Whenever a database record is requested only the fields specified in this mask will be returned.

Get User Groups

Syntax : GUG

Returns a comma delimited list of the message notification channels to which the user has subscribed using the Set User Groups command.

Get User Name

Syntax : GUN

Returns the name by which the client is logged on.

Get User Type

Syntax : GUT

If the User Type flag is turned on this command will return 1, otherwise it returns 0. If set to 1 by a dedicated client the server will send blocks of data prefixed and suffixed by STX and ETX bytes respectively. If set to 0 no such bytes will be sent. The default is 0.

Set Commands

Set Current Amendment

Syntax : SCA <Amend%>

Sets the current amendment to be <Amend%>. Valid number are between 0 and 99.

Set Current Booking

Syntax : SCB <Bkg%>

Sets the current booking to be <Bkg%>. Valid number are between 1 and 999.

Set Current Day

Syntax : SCD <Day%>

Sets the current day to <Day%>. Valid number are between 1 and 31.

Set Current Field

Syntax : SCF <Field%>

Sets the current field to <Field%>. Valid numbers are between 1 and 128.

Set Current Month

Syntax : SCM <Month%>

Sets the current month to <Month%>. Valid number are between 1 and 12.

Set Current Year

Syntax : SCY <Year%>

Sets the current year to <Year%>. Valid number are between 1990 and 3000.

Set Database Extension

Syntax : SDE <Filename\$> <Data\$>

Writes <Data\$> to the file <Filename\$>. For a telnet client all data following the space after the <Filename\$> parameter up to the carriage return will be copied to the file. For a dedicated client that uses STX (ASCII 2) to start the command, all data up to the next ETX (ASCII 3) will be copied to the file.

Set Database Field

Syntax : SDF <Data\$>

Sets the field pointed to by the current month, day, booking, amendment and field to <Data\$>

Set Database Lock

Syntax : SDL <Switch%>

If <Switch%> = 1 then the current booking is marked as locked by the server. Fields for this booking can now only be written to by your client. Using <Switch%> = 0 removes the lock. Only one booking may be locked by a client at any one time. Other clients cannot steal your lock. Error messages are generated for invalid operations.

Set Database Record

Syntax : SDR <Data\$>

Sets the field pointed to by the current month, day, booking, amendment and field to <Data\$>. <Data\$> is a comma delimited list of fields which will be applied to the fields in the database record as they appear in the user field configuration as set by the Set User Fields command.

Set Database Update

Syntax : SDA

Sets the state of the database update flag. When this flag is set to 1 the server will send a message to the client whenever a booking changes. If set to 0 no such notification will be sent. The default is 1.

Set Resource Commitment

Syntax : SRC <Cmd\$> <Resource\$> <Start\$> <End\$>

Adds or deletes a commitment against the current booking. If <Cmd\$> = ADD a commitment is added. If <Cmd\$> = DELETE the commitment is removed. <Resource\$> is the resource name and <Start\$> and <End\$> are the start and end times respectively.

Set Resource New

Syntax : SRN <Resource\$>

Adds <Resource\$> to the list of available resources

Set Resource Remove

Syntax : SRR <Resource\$>

Removes <Resource\$> from the list of available resources

Set Resource Update

Syntax : SRU <Flag%>

Sets the state of the resource update flag. When this flag is set to 1 the server will send a message to the client whenever a resource commitment changes. If set to 0 no such notification will be sent. The default is 1.

Set System Time

Syntax : SST

Sets the system time at the server.

Set User Control

Syntax : SUC <Password2>

This is the extra password required before access is allowed to the control functions provided by the server. This is not the same password used with the SUP command. Without this second password only booking system functions will be available.

Set User Echo

Syntax : SUE <Flag%>

If <Flag%> = 1 then any characters sent to the server will be echoed back to the user. If set to 0 then there is no echo. Default is on (1).

Set User Fields

Syntax : SUF <Fields\$>

<Fields\$> is a comma delimited list of the fields required by the user. Whenever a database record is requested only the fields specified in this mask will be returned.

Set User Groups

Syntax : **SUG** <Groups\$>

<Groups\$> is a comma delimited list of the message notification channels to which the user wishes to subscribe. To unsubscribe to a single group use SUG with just the list of groups you wish to continue to subscribe to. To unsubscribe to all groups use SUG with no argument.

Set User Name

Syntax : **SUN** <Name\$>

This is the first part of the logon process. <Name\$> is your username.

Set User Password

Syntax : **SUP** <Password\$>

This is the second part of the logon process. <Password\$> is a valid password for your user name.

Set User Quit

Syntax : **SUQ**

Informs the server that you wish to close the connection. The server closes the session gracefully.

Send Commands

Send Group Message

Syntax : **SGM** <Group%> <Message\$>

Sends <Message\$> to all members of group <Group\$>. You cannot send messages to groups that you are not configured to receive messages from using the Set User Groups command.

<i>Author</i>	Andy Woodhouse			<i>Tic Reg. No</i>	
<i>File Name</i>	Fabian.pages				
<i>Version no.</i>	1	<i>File Date</i>	27/11/2005 21:39	<i>Print Date</i>	