BNCS Device Drivers

Riedel Artist Comms/Talkback Router

RiedelArtistDrv.exe

Doc 1.36, Driver 14.2

Written by Giles Moss

BNCS

Contents

BNCS

1 Overview

1.1 Description

The application **RiedelArtistDrv.exe** is a BNCS driver for the Riedel Artist communications/talkback router system.

The Artist router is an extremely flexible device so this application only exposes certain interfaces and functions. The BNCS driver has no knowledge of and therefore does not interfere with any of the other functions available in the Artist.

Ensure you read the *Notes* chapter of this document as it contains some important things to understand when using the driver.

1.2 Riedel Interface

1.2.1 Configuration

Configuration of the router is achieved through Riedel's *Director* software. This is very powerful and if you don't know how to use it then do not touch! Training and support is far beyond the scope of this document.

1.2.2 Remote Control

The BNCS driver interfaces to the Artist router via a gateway application developed by Riedel known as RRCS. Standing for *Riedel Remote Control System*, RRCS allows third-party control of certain aspects of the Artist router's configuration.

The connection between the BNCS driver and RRCS is a pair of TCP sockets. Conventionally both the BNCS driver and RRCS run on the same driver PC.

In turn, RRCS communicates with the Artist router via a TCP connection to port 8192.

1.3 Supported Firmware

This version of the BNCS driver was developed against Artist firmware **v6.40** and has been extensively tested against firmware **v6.60**. It will not operate with firmware versions earlier than v6.40.

Although Riedel traditionally maintain backwards compatibility when developing new firmware, extreme care must be taken if using this software with a different Artist firmware version.

Artist firmware **v6.90** is required if the trunking commands are enabled or if there are any ports defined in the VOIPCONFIG INI file.

1.4 Hardware Dongle

RRCS requires a hardware dongle before it will allow commands to query the actual router. Should the dongle not be available, the BNCS driver will fail to connect to RRCS and the debug log will report a "Dongle not available" error.

This dongle is a USB HASP key. Support for this is available from Riedel; they will have to provide a dongle with your copy of RRCS in the first place.

Driver software for the dongle is available from the HASP website (http://www.aladdin.com/support/hasp.aspx).

This hardware dongle requirement means that careful thought is required if you are intending to run RRCS on a virtualised driver workstation (basically; *don't*).

1.5 BNCS Interfaces

One BNCS external host is required for each interface so the BNCS driver acts as an external to multiple 32-bit InfoDrivers and GRDs. V3 and V4 environments are supported.

Overall configuration settings are read from the INI file corresponding to the XY GRD.

The driver connects with its BNCS host devices in the usual way, taking a command line parameter of [XY GRD number]

e.g. `\drivers\riedelartistdrv.exe` **nnn**

Note: All BNCS host drivers must already be running on the same machine.

If the command line parameter is not present or is invalid in some way, the driver will pop up an error message box and will not start.

1.5.1 Table of BNCS interfaces and how their device numbers are configured

Interface	BNCS Host Type	Device Number
XY crosspoint routing.	GRD	Supplied as command line argument to application.
Monitoring	InfoDriver	Defined in INI file.
PTI	InfoDriver	Defined in INI file.
Port Aliases	InfoDriver	Defined in INI file.
GPIO	InfoDriver	Defined in INI file.
Gain	InfoDriver	Defined in INI file.
Conferences	InfoDriver	Defined in INI file.
IFB	InfoDriver	Defined in INI file.
Extras	InfoDriver	Defined in INI file.

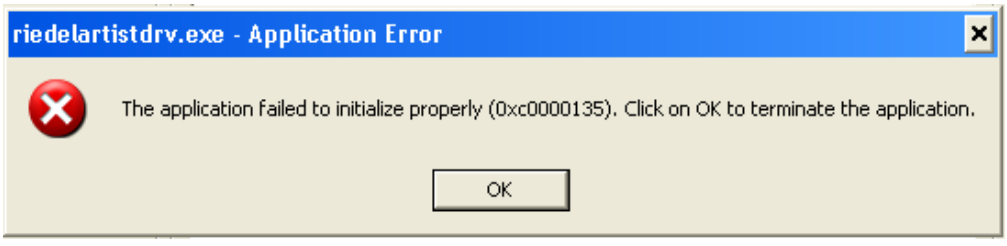
The driver checks to ensure device numbers are not duplicated between the command line and the configuration, i.e. all device numbers must be unique.

1.6 Software Requirements

This driver requires Microsoft's .NET framework v4.0 and Windows XP SP3 or later. It has also been tested on Windows Server 2008 R2.

1.6.1 Missing .NET

If the driver is started without the .NET framework installed, an error message similar to the following will be displayed.



1.7 Driver GUI

1.7.1 Screenshot

1.7.2 Title Bar

The driver's title bar contains:

- The primary device number (the one which it was started with, and the number of the XY GRD).
- The driver name.
- The current BNCS TxRx or RxOnly mode of the driver.

1.7.3 Indications

The driver GUI is divided into two main sections, the upper *status* section and the lower *debug log* section.

The *status* section is further divided into the statuses for the BNCS hosts and the statuses for the RRCS messaging interfaces.

The status bar for each BNCS interface displays information including the name of the interface, the device number of the host, how many messages have been transmitted to and received from the host, and how many requests the host has sent the driver. The background colour of the BNCS interface status bars is set according to the TxRx mode of the respective host: green for TxRx, dark red for RxOnly and grey for disconnected or unknown. In normal operation you should never see the indication with a grey background.

Similarly, the status bar for the RRCS message interface is separated into *outward* and *inward* interfaces. Outward messages are requests sent to RRCS by the driver. Inward messages are notifications received from RRCS. The background colour of the outward interface is green when the driver is happy it has communication with RRCS and RRCS can communicate with the Artist router, and is grey when this is not the case. The background colour of the inward interface is green when the driver is listening for connections from RRCS and grey when it is not. As is the case with the BNCS host statuses, in normal operation you should not see a grey background.

The RRCS status display also includes an indication of the loop times for message requests, and the length of the internal message queue.

1.7.4 Debug Log Window

Everything the driver does can be logged to debug window. This can be useful in troubleshooting situations or purely for your own interest. The amount of detail shown is controlled by the associated menu command and INI file setting (see section 2.1).

Logging is asynchronous so does not slow the driver operation down, and the debug window is limited to displaying the last 1000 messages to prevent it from taking too much memory.

1.7.5 Menu Commands

1.7.5.1 Tools

Reset: Commands to reset the statistics counters and clear the debug window.

Sync Router: Manually trigger a router synchronisation. See section 4.1.

Show Debug Tools: Displays a window with some power user tools.

Clear Queue: Clears any queued Configuration Changes, and any pending requests that might be waiting. See section 5.6.3.

1.7.5.2 Settings

Debug Window Level: Controls how much detail is displayed in the debug window.

Log to File Mode: Controls whether a text log file is written and if so, how it is named and over-written.

Always On Top: Keeps the driver window on top of other windows.

1.7.5.3 BNCS

Connect to BNCS: Manually kick off the connect process, if no BNCS drivers are connected.

Working Mode: Force the mode of the host XY GRD to TxRx or RxOnly.

1.7.5.4 Help

About: Displays an information window.

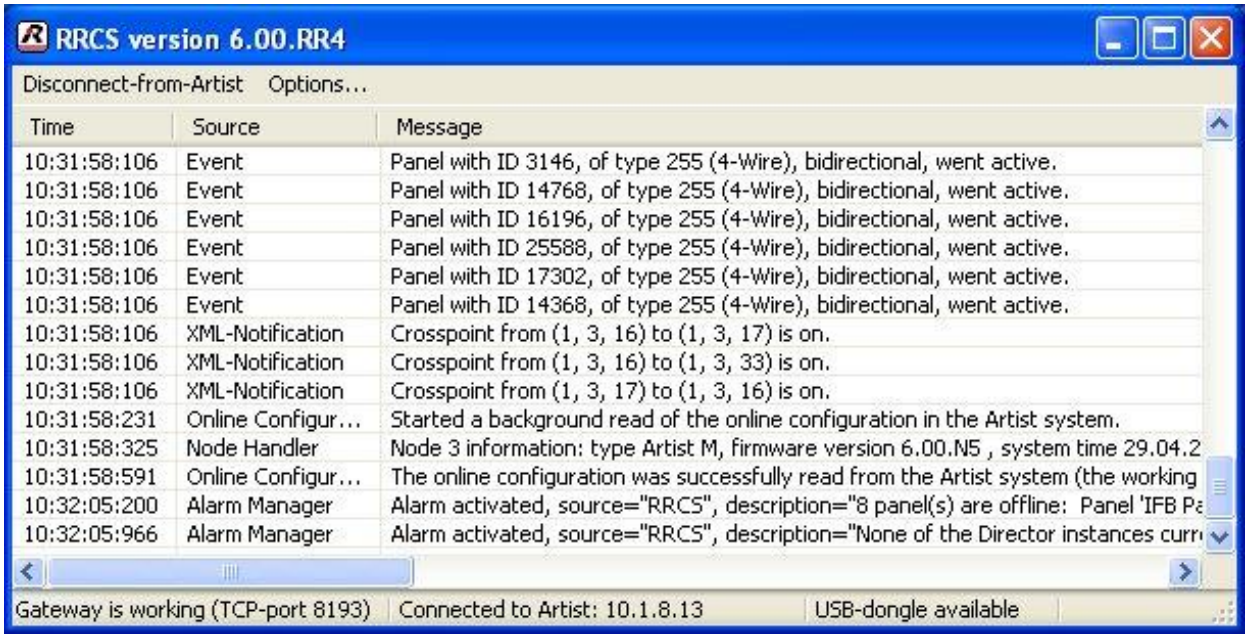
Open Logfile: If the BNCS driver is configured to write the debug log to a text file, this function will open that text file in Notepad.

Open Startup Error Log: If there were any errors encountered during driver startup, errors are separately written to a log file which can be opened via this function.

1.8 RRCS Interface Application

The driver communicates with the Artist router through Riedel's RRCS software.

1.8.1 Screenshot



1.8.2 Configuration

RRCS requires little configuration, and it is all available from the Options menu. This opens up a properties window with two tabs, *Logging* and *Router Options*. The Logging tab lets you choose how much detail is displayed in the log. The Router Options tab lets you specify

the IP address of the Artist frame to connect to and whether RRCS should automatically attempt to connect to the Artist on start up.

Ensure the *Autoconnect to Artist* checkbox is ticked. All other checkboxes on the Router Options page should be left un-ticked.

1.9 Object IDs

Within the Artist all objects (ports, conferences, etc.) are assigned a unique 32-bit ID number. This number is treated as an internal handle to that object by the Artist firmware.

The driver needs to know the object IDs for most things it's manipulating, and as such the INI files are full of object ID definitions.

Care must be taken to avoid specifying the same object ID in more than one place across the entire INI file configuration.

Object IDs can only be extracted from a router running the configuration in which they're defined. Don't ask why, as the IDs must exist in the offline configuration file, but Riedel haven't made the information directly available to us.

To help with driver configuration, there is a tool to extract all relevant object IDs from the router and copy them to the clipboard. See section 6.

2 Driver setup

2.1 INI file settings

The setup INI file is shared with the XY GRD. Most settings are stored under the [RiedelArtistDrv] section with the exception of the *NAME* setting, which is stored under the [GRD] section.

Several settings point to further configuration files required by the different interfaces.

The BNCS driver will create this file if it doesn't exist and will populate it with default values.

Item	Description
LOGLEVEL	The amount of detail logged to the debug window. Settings can be 0 (logging off) to 6 (extremely fine detail). Defaults to 3 which results in sufficient logging for most use.
LOGFILEPATH	The folder in which to store application log files. Defaults to c:\bncslogs
DEBUGLOGENABLE	Controls whether the driver writes its debug log output to file. Defaults to 0 (no logging). Other allowed values are: 1: Log file rotated daily, over-written once a month. Filename DEV_nnn.dd 2: Log file rotated daily, never over-written. Filename DEV_nnn.yyyyMMdd 3: Log file rotated hourly, over-written once a day. Filename DEV_nnn.HH 4: Log file rotated hourly, never over-written. Filename DEV_nnn_yyyyMMdd.HH
NAME	Text string displayed in the coloured bar at the top of the driver window. Defaults to "Riedel Artist Driver – Device nnn".
PORTCONFIG	**Additional configuration file** defining the port mapping between sources/destinations on the XY GRD and Riedel subscribers. Defaults to "RiedelPorts.ini" in the same folder as the GRD INI file. Further information in section 2.2.
MONITORCONFIG	**Additional configuration file** defining the monitoring outputs of the Artist router. Defaults to "RiedelMonitors.ini" in the same folder as the GRD INI file. Further information in section 2.3.
GPICONFIG	**Additional configuration file** defining the mapping between General Purpose I/Os in the Artist and slots in the GPI InfoDrivers Defaults to "RiedelGPIOs.ini" in the same folder as the GRD INI file. Further information in section 2.4.

Item	Description
CONFCONFIG	**Additional configuration file** defining the Artist conferences to be modified. Defaults to "RiedelConfs.ini" in the same folder as the GRD INI file. Further information in section 2.5
IFBCONFIG	**Additional configuration file** defining the Artist IFBs to be modified. Defaults to "RiedelIFBs.ini" in the same folder as the GRD INI file. Further information in section 2.6
GROUPCONFIG	**Additional configuration file** defining the Artist groups for call-to-group commands. Defaults to "RiedelGroups.ini" in the same folder as the GRD INI file. Further information in section 2.7.
LOGICCONFIG	**Additional configuration file** defining any Logic Sources the driver should control. Defaults to "RiedelLogicSources.ini" in the same folder as the GRD INI file. Further information in section 2.8.
MIXERCONFIG	**Additional configuration file** defining any mixer entities the driver should control. Defaults to "RiedelMixers.ini" in the same folder as the GRD INI file. Further information in section 2.9.
VOIPCONFIG	**Additional configuration file** defining any VoIP cards or ports whose properties driver should control. Defaults to "RiedelVoip.ini" in the same folder as the GRD INI file. Further information in section 2.11.
PORTACTMONCONFIG	**Additional configuration file** defining any ports whose active status should be monitored by the driver. Defaults to "RiedelPortActMonConfig.ini" in the same folder as the GRD INI file. Further information in section 2.12.
RRCSIP	The IP address of the PC on which RRCS is running. Defaults to 127.0.0.1.
RRCSPORT	The port number on which RRCS is listening. Defaults to 8193.
RRCSREVIP	The IP address of the interface on which to listen to reverse messages from RRCS. Defaults to 127.0.0.1.
RRCSREVPORT	The port number that the application should open in order to listen to messages sent from RRCS. Defaults to 8194.
SYNCONCO	Flag controlling whether the application performs a router synchronisation automatically on a RxOnly to TxRx mode changeover. Set to YES or NO. Defaults to YES.

Item	Description
TIMESTAMPSENDSTR	Flag controlling whether SendString revertive messages are pre-pended with the current time. Set to YES or NO. Defaults to YES.
NOXYTOCONFMEMS	Flag controlling whether the driver prohibits XY GRD crosspoints to ports that are currently a listen member of a conference. Set to YES or NO. Defaults to NO.
LINKDBTOPORTALIAS	Flag controlling whether the driver links names from BNCS source/destination name databases to Riedel port aliases. Further information in section 3.4.3. Set to YES or NO. Defaults to NO.
MONCLEARLABELDB	Specifies a database (on the monitoring interface's device number) to use for default labels for monitoring destinations. Further information in section 3.2.2.9. Set to NO or a number from 0 to 9. Defaults to NO.
SPLITIFBSHORTCMDS	Flag controlling whether, on processing an IFB shortcut command, the driver sends *only* the property which has been specified by that shortcut command ("YES") or whether it fills out the request with ports/labels from its internal tables ("NO"). Further information in section 3.8.4. Set to YES or NO. Defaults to YES. Needs to be set to YES if a single IFB construct is to be targeted by both the IFB and mixer interfaces.
SYNCRRCSCMDSONLY	Flag controlling whether the driver sweeps *all* commands off ports during its synchronisation process ("NO"), or whether it *only* removes commands it (or a previous driver instance) created ("YES"). Set to YES or NO. Defaults to YES.
GORXONRRCSLOSTRTR	Flag controlling whether the driver should immediately switch to RxOnly when RRCS reports it has lost comms with the router (so that a second instance of the driver may take over) or not. If not, the driver will report to CSI it has no comms to the router so the BNCS redundancy mechanism will eventually kick in and request that the driver goes RxOnly, which it will. This slightly experimental setting is intended to prevent driver changeovers if the RRCS-to-router communication failure is only momentary. Set to YES or NO. Defaults to NO.
DEBUGCONFPORTS	Flag controlling whether the driver logs additional debugging information during a conference command. Further information in section 3.7.6. Set to YES or NO. Defaults to NO.
MONITORDEVICE	The device number of the monitoring control InfoDriver. Must be a positive integer between 1 and 999 and must not be the same as the ID with which the application is already running. Defaults to one larger than the base device.
PTIDEVICE	The device number of the PTI interface InfoDriver. Must be a positive integer between 1 and 999 and must not be the same as the ID with which the application is already running or any of the other DEVICE IDs. Defaults to 2 larger than the base device.

Item	Description
ALIASDEVICE	The device number of the Alias interface InfoDriver. Must be a positive integer between 1 and 999 and must not be the same as the ID with which the application is already running or any of the other DEVICE IDs. Defaults to 3 larger than the base device.
GPIODEVICE	The device number of the GPIO interface InfoDriver. Must be a positive integers between 1 and 999 and must not be the same as the ID with which the application is already running or any of the other DEVICE IDs. Defaults to 4 larger than the base device.
GAINDEVICE	The device number of the Gain interface InfoDriver. Must be a positive integer between 1 and 999 and must not be the same as the ID with which the application is already running or any of the other DEVICE IDs. Defaults to 5 larger than the base device.
CONFDEVICE	The device number of the Conference interface InfoDriver. Must be a positive integer between 1 and 999 and must not be the same as the ID with which the application is already running or any of the other DEVICE IDs. Defaults to 6 larger than the base device.
IFBDEVICE	The device number of the IFB interface InfoDriver. Must be a positive integer between 1 and 999 and must not be the same as the ID with which the application is already running or any of the other DEVICE IDs. Defaults to 7 larger than the base device.
EXTRASDEVICE	The device number of the Extras interface InfoDriver. Must be a positive integer between 1 and 999 and must not be the same as the ID with which the application is already running or any of the other DEVICE IDs. Defaults to 8 larger than the base device.
CROSSPOINTPRI	The priority with which all XY crosspoints should be set. Allowed values are: BELOWSTANDARD STANDARD (default) HIGH PAGING EMERGENCY
LOGGETALIVES	If set to YES, the log will include regular GetAlive queries from RRCS. These are sent by RRCS to ensure the BNCS driver is still responding, and are sent once per second. As such, the debug log will quickly fill up with worthless messages so this flag can be used to inhibit the logging of such messages. Defaults to NO.

Item	Description
LOGFUNCTIONERRORS	If set to YES, the Alarm slots (see section 3.16.1) will include error messages from failed commands to the Artist in addition to port and node failures. Note that errors relating to malformed commands or out of range data (e.g. setting a volume to an impossible number) are not displayed. It is expected that the panel will not allow such malformed commands to be sent. Set to YES or NO. Defaults to YES.
ONLINENODEMONITOR	A comma-delimited list of Artist nodes whose online status should be monitored. Defaults to 0, meaning this function is disabled.
NETTRUNKADDRESS	The trunking address of this Artist ring. Defaults to 0, meaning trunking functions are disabled. Note if this is non-zero, **RRCS** must be version **6.90.RR1** or later.
LOGXML	If set to YES, all XML commands sent to/from RRCS are logged to the debug output. This can result in extremely large log files so should not be enabled in regular operation. Defaults to NO.

2.1.1 Sample INI file

```
[RiedelArtistDrv]
LOGLEVEL=3
LOGFILEPATH=c:\bncslogs
DEBUGLOGENABLE=0
PORTCONFIG=c:\bncs\mcr\config\riedelports.ini
MONITORCONFIG=c:\bncs\mcr\config\riedelmonitors.ini
GPICONFIG=c:\bncs\mcr\config\riedelgpios.ini
CONFCONFIG=c:\bncs\mcr\config\riedelconfs.ini
IFBCONFIG=c:\bncs\mcr\config\riedelifbs.ini
GROUPCONFIG=c:\bncs\mcr\config\riedelgroups.ini
LOGICCONFIG=c:\bncs\mcr\config\riedellogicsources.ini
MIXERCONFIG=c:\bncs\mcr\config\riedelmixers.ini
VOIPCONFIG=c:\bncs\mcr\config\riedelvoip.ini
RRCSIP=127.0.0.1
RRCSPORT=8193
RRCSREVIP=127.0.0.1
RRCSREVPORT=8194
SYNCONCO=YES
SYNCRRCSCMDSONLY=YES
GORXONRRCSLOSTRTR=YES
DEBUGCONFPORTS=NO
MONITORDEVICE=877
PTIDEVICE=878
ALIASDEVICE=879
GPIODEVICE=880
GAINDEVICE=881
CONFDEVICE=882
IFBDEVICE=883
EXTRASDEVICE=884
CROSSPOINTPRI=STANDARD
TIMESTAMPSENDSTR=YES
LOGXML=NO
LOGGETALIVES=NO
LOGFUNCTIONERRORS=YES
ONLINENODEMONITOR=2,3,4
NETTRUNKADDRESS=1

[GRD]
NAME=Riedel Driver (4W Ring)
```

2.2 PORTCONFIG configuration file

2.2.1 Address

Within the Artist router, all subscriber and panel ports are addressed by a dotted decimal structure comprising four fields for Network, Node and Port.

The port address within a node is calculated using the formula:

```
Port address = ((Slot no. – 1) * 8) + Port no. on Client Card – 1
```

For example, Port 5 on Slot 2 has the port address ((2 – 1) * 8 + 5 – 1) = 12. Therefore the PORTCONFIG entry:

```
1.2.12
```

Refers to a subscriber in Artist network 1, node 2, card 2, port 5.

Note: The Artist network number is always 1. Even Riedel admit it can never be anything else.

2.2.2 Port Type

A subscriber port can be of a variety of times, detailed in the enumeration:

Type	Description
1	2-wire input only
2	2-wire output only
3	4-wire split
4	4-wire
5	Panel
6	Panel, using second audio channel
7	Codec (e.g. ISDN)
8	Pool port

Subscriber ports are defined in the Riedel configuration as either:

- 4-wire ports, in which the input and output halves are related (e.g. the input and output of an ISDN codec), which relates to our enumerated type 4.
- Split 2-wire input and output ports, in which there is no relation between the input and output halves of the port, which relates to our enumerated type 3.

It is not possible to define just an input or an output port in the Riedel configuration, it will insist upon its other half being present. The driver's port type supports such a concept; however, to exclude half a port from driver control should it prove necessary.

Split ports appear as separate input and output ports in the Artist configuration, but are referenced by the same GRD index in the driver.

The driver pays careful attention to the port type and will only allow certain operations to certain halves of the port. For example, you would not be able to create a call-to-port command to a port which has no audio output, since the command would be nonsensical.

2.2.3 Pool ports and codecs

Pool ports are virtual ports within the Artist that are tied to a shared physical resource. For example, an ISDN or TBU unit may be physically connected to an audio port, but has two virtual pool ports for each half of the stereo pair. Similarly, a single VoIP 4-wire port can have a separate pool port for an individual SIP incoming number. The Artist intelligently uses pool ports to trigger commands or route audio based on an external event such as an incoming telephone call and some properties of it, such as the caller ID.

A single pool port, therefore, has an underlying physical port (where the audio actually goes) and a number of virtual ports on top of it on which commands (e.g. to join a conference) can be placed. Many pool ports can share the same underlying physical port.

The port types for pool port usage are:

> 7: The underlying codec. Has a physical audio port so can have audio routed to/from it by the XY GRD, but cannot be placed directly into a conference (for example). In regular use this need never be mapped in the configuration but the port type is included for completeness or future development.
>
> 8: Pool port. Has a physical audio port (to allow XY GRD audio routes) and can be placed into a conference (for example). Care should be taken when making XY GRD routes to a pool port because obviously the physical audio destination may be shared between multiple pool ports.

Pool ports, therefore, can be treated as regular ports for conference purposes, but care should be taken if treating them as audio ports because the underlying physical port may be shared.

2.2.4 INI File Entry (regular ports)

Port mappings are defined in the INI file under the section `[MAP]` and are in the format:

```
nnnn=x.x.x|PORTTYPE
```

where `nnnn` is a number between 1 and 2000 (the maximum number of XY ports is 2000) and the `PORTTYPE` is one of the values from the Port Type enumeration.

The port mapping INI file is essential. The BNCS driver will not start if the file is missing, incorrectly formatted or contains no valid port mappings.

Note: ObjectIDs for ports are read from the Artist when the driver connects.

2.2.5 INI File Entry (pool ports and codecs)

Codec and pool ports Port are defined in the INI file under the section `[POOLPORTS]` and are in the format:

```
nnnn=x.x.x|PORTTYPE|OBJECTID
```

where `nnnn` is a number between 1 and 2000 (the maximum number of XY ports is 2000), the `PORTTYPE` is one of the appropriate values from the Port Type enumeration and `OBJECTID` is the object ID of the codec or pool port. `x.x.x` is the net.node.port address of the underlying physical port.

The ObjectID has to be explicitly set here because it cannot be looked up by the driver.

Note: If using an ISDN on a two-channel port, take care when setting the net.node.port for the pool port. If the pool port is associated with the second channel of the codec, the port value will need to be incremented accordingly. Usually (i.e. in the case of two-channel panels), the port corresponding to the second channel of a port is ignored by the driver.

2.2.6 Sample PORTCONFIG INI file

```
[MAP]
0001=1.1.1|4
0002=1.1.2|4
0003=1.1.3|4
(etc…)
```

For further information see *XY Crosspoints* (section 3.1).

2.2.7 Panel and Monitoring Ports

In addition to mapping regular 4-wire ports, the XY GRD port configuration *must* include mappings for all panel ports and 4-wire destination monitors.

For further information see *XY GRD mapped ports* (section 2.3.5).

2.3 MONITORCONFIG configuration file

The Artist router supports various different types of monitoring output. To make use of a monitoring destination it must first be defined in the INI file. When a monitor is requested, the driver determines the nature of the request and the nature of the destination and makes the appropriate request to the Artist router.

2.3.1 INI File Entry

The INI file is populated with one definition section for each monitor destination. The INI file section is `[MONITOR-nnnn]` where `nnnn` is a number between 1 and 4000.

2.3.2 Monitor Types and INI file parameters

Different monitoring destination types require different settings.

Item	Item	Applicable Monitor Type(s)
TYPE	The type of monitoring destination. Allowed values are: 4WD: 4-wire destination port (i.e. an XY crosspoint destination) KEY: A single key on a panel. IGNORED: Overlook this entry quietly (allows a dormant configuration entry to exist).	ALL
DESTINATION	The crosspoint destination. Must be a port on the Artist router that is *not* otherwise defined in the XY GRD configuration. In the standard Artist port definition of Net.Node.Port, for example "1.2.3".	4WD

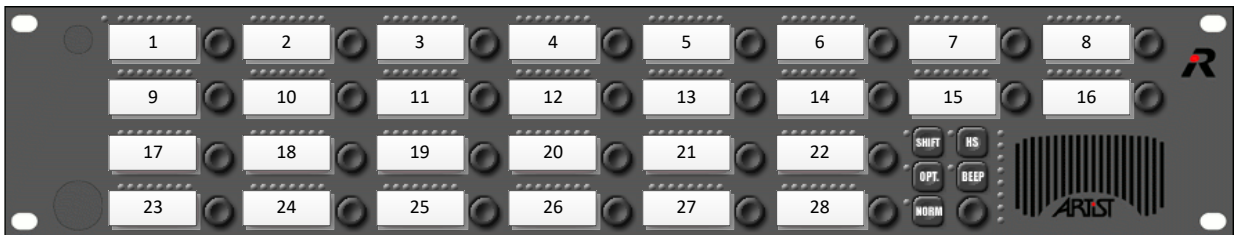
Item	Item	Applicable Monitor Type(s)

KEY	The address of the key.
	This is similar to the standard Artist port definition but includes additional parameters. The format is
	`Net.Node.Port.ExpansionPanel.PageNumber.Key`
	The `ExpansionPanel` should be zero if the key is on the main panel, or between 1 and 6 if the key is on an expansion panel (expansion panel number is allocated in Director).
	The `PageNumber` should be 1 for the main page or 2 for the shift page.
	The `Key` is the number of the key on the panel, the maximum value depends on the panel but is usually 28 or 32.
	For example, the configuration
	`1.2.33.2.1.15`
	Refers to a key on network 1, node 2, port 33, expansion panel 2, page 1, key 15.
		KEY

The MONITORCONFIG INI file must exist, however it can be blank.

2.3.3 Key Button Layout

The position of the keys on a panel is illustrated in the following graphics. In panels with differing numbers of keys, they are always numbered from the top left and are in row order.



New-style smart panels number in columns:

2.3.4 Sample MONITORCONFIG INI file

```
[MONITOR-0001]
TYPE=4WD
DESTINATION=1.3.2
```

This sample file entry defines a monitoring destination as an XY crosspoint destination on the Artist port on net 1, node 3, port 2 (client card 1, port 2 – see the PORTCONFIG configuration file notes, above).

For further information see *Monitoring (InfoDriver presentation)* (Section 3.2).

2.3.5 XY GRD mapped ports

2.3.5.1 4-wire destination monitors

Monitoring destinations configured as 4-wire destination monitors are, in truth, just 4-wire destination ports. The XY GRD must contain a destination for these ports.

During start up, the driver attempts to find the corresponding XY GRD subscriber port number for all 4-wire destination monitors. If this is not possible, the driver will log an error and the monitoring destination will be rejected from the configuration.

2.3.5.2 Panels

An Artist panel key exists in a panel, which itself exists on an Artist subscriber port. For monitoring destinations configured as panel keys, the address of the Artist subscriber to which the key's parent panel is connected must be defined in the XY GRD mapping.

During start up, the driver attempts to find the parent subscriber port number for all defined keys. If this is not possible, the driver will log an error and the key will be rejected from the configuration.

You can configure as many panel keys on a panel as you like, but the panel should only be mapped once in the XY GRD.

2.4 GPICONFIG configuration File

In a similar way to the XY GRD, the driver requires a mapping between GPI index in the GPI and GPO InfoDrivers and the physical GPIO port in the Artist.

Port mappings are defined in the INI file under the `[INPUTS]` and `[OUTPUTS]` sections and are in the format `nnnn= NET.NODE.PORT.SLOT.GPIO` where `nnnn` is a number between 1 and 2000.

NET and NODE refer to the physical node on the network in the same way as the XY GRD mapping.

If the GPI is hosted on a *panel*, `PORT` is the port number from `0` to `127` on the Artist frame to which the panel is connected. `SLOT` is ignored and `GPIO` is the index number of the GPIO on the panel (`0..2`, usually).

If the GPIO is hosted on a physical *GPIO card* in the Artist frame, then `PORT` must be set to `128`. In this case, `SLOT` refers to the location of the card in the frame according to the following table, and `GPIO` is the index of the GPIO on the GPIO card (`0..15` for the 16 GPIOs).

SLOT	Description
0..15	GPIO card in slots 1 to 16
16	GPIO card in bay X
17	GPIO card in bay Y
20	GPIO card in bay A

For example, the GPIO definition

```
0001=1.2.128.16.3
```

addresses the fourth GPIO on a GPIO card in bay X of node 2.

The GPICONFIG INI file must exist, however it can be blank.

2.4.1 Sample GPICONFIG INI file

```
[INPUTS]
0001=1.1.1.1.1
0002=1.1.1.1.2
[OUTPUTS]
0001=1.1.1.1.4
0002=1.1.1.1.5
```

For further information see *General Purpose I/O (InfoDriver presentation)* (Section 3.5).

2.5 CONFCONFIG configuration file

In order to manipulate the members of an Artist conference, that conference must already exist as an object in the Artist configuration.

Conferences are defined in the INI file under the `[Conferences]` sections and are in the format `nnnn=x|y` where `nnnn` is a number between 1 and 700, `x` is the object ID of the conference and `y` is the (optional) name. Note that the maximum number of conferences supported by the Artist is usually 350.

The CONFCONFIG INI file must exist, however it can be blank.

2.5.1 Sample CONFCONFIG INI file

```
[Conferences]
0001=3568452125|Gallery A
0002=2154784596|Technical
```

For more information, see *Conferences (InfoDriver presentation)* (Section 3.7).

2.6 IFBCONFIG configuration file

In a very similar way to the conferences, in order to manipulate the members of an Artist IFB, the IFB must already exist in the Artist configuration.

The INI file is populated with one definition section for each IFB. The INI file section is `[IFB-nnnn]` where `nnnn` is a number between 1 and 700 (note that the maximum number of IFBs supported by the Artist is usually 350).

The IFBCONFIG INI file must exist, however it can be blank.

2.6.1 Multiple Ports

An IFB construct comprises three elements:

- Mix Minus (or clean feed): Audio usually sent to the destination until a panel speaks over it.

- Output: The destination port.
- Input: Audio source that a panel key's call-to-IFB function should monitor.

The IFB interface can accept multiple ports for each of these, so, for example, you could send a single IFB to multiple destinations simultaneously. This is achieved using Artist Group constructs.

For example: If an IFB output is a single port, the IFB construct is configured with that port as the output parameter. Conversely, if the IFB output is a group of ports, the IFB construct is configured with a group as the output parameter, and the ports are added to that group. The driver handles the group membership and switching the IFB constructs to/from group operation as needed.

The group construct must be pre-defined in the Artist configuration, and the object ID of the group entered into the IFB configuration file. If the Object ID is zero, the driver will only allow a single port for the IFB's input, mix minus and output.

2.6.2 Sample IFBCONFIG INI file

```
[IFB-001]
NAME=STD1OS1
OBJID-IFB=1254785698
OBJID-GROUPOUT=789456123
OBJID-GROUPIN=5647985
OBJID-GROUPMXM=123469875
[IFB-002]
NAME=STD1OS2
OBJID-IFB=5969498
OBJID-GROUPOUT=56984663
OBJID-GROUPIN=7854513
OBJID-GROUPMXM=98411265
```

For more information, see

IFBs (InfoDriver presentation) (Section 3.8).

2.7 GROUPCONFIG configuration file

In order to create a call-to-group function on a key, that group must already exist as an object in the Artist configuration.

Groups are defined in the INI file under the `[Groups]` sections and are in the format `nnnn=x` where `nnnn` is a number between 1 and 700 and `x` is the object ID of the group.

The GROUPCONFIG INI file must exist, however it can be blank.

It is permitted to define a group in both the GROUPCONFIG.INI and as an IFB port group (see Section 2.6, *IFBCONFIG configuration file* for details).

2.7.1 Sample GROUPCONFIG INI file

```
[Groups]
0001=65798414
0002=126157825
```

For more information, see *Key configuration (type KEY)* (Section 3.2.2).

2.8 LOGICCONFIG configuration file

In order to target a logic source from a key, or to remotely control it, the logic source must already exist as an object in the Artist configuration.

Logic sources are defined in the INI file under the `[LogicSources]` section and are in the format `nnnn=x` where `nnnn` is a number between 1 and 2000 and `x` is the object ID of the logic source.

The LOGICCONFIG INI file must exist, however it can be blank.

2.8.1 Sample LOGICCONFIG INI file

```
[LogicSources]
0001=45847981
0002=561798416
```

For more information, see *Key configuration (type KEY)* (Section 3.2.2) and Remote Logic Sources (Section 3.14).

2.9 MIXERCONFIG configuration file

The Artist can provide a simple audio mixing function. A 'mixer' is a composite entity involving several objects in the Artist.

The INI file is populated with one definition section for each Mixer. The INI file section is `[MIXER-nnnn]` where `nnnn` is a number between 1 and 500.

The mixer entity requires one entry in the IFB table (defined as IFB number) and two groups (one for inputs and one for outputs, defined as group object IDs). The configuration also defines the number of inputs and outputs that the mixer can use, which can be between 1 and 20. Note that the inputs to the mixer are placed into the MixMinus section of the IFB Table.

If the mixer is configured with a single output, a group is not needed.

The MIXERCONFIG INI file must exist, however it can be blank.

2.9.1 Mixer/IFB concurrency

It is possible to specify the same IFB construct number for both a standard IFB and a mixer's IFB.

If an IFB is defined in both locations, the IFB interface will only allow control of the Label and Input properties of the IFB construct. The mixer interface must be used to control the Mix Minus and Output properties.

2.9.2 Sample MIXERCONFIG INI file

```
[MIXER-0001]
NAME=Presentation
IFBNUMBER=3
OBJECTID-MIXMINUSGROUP=45619817
OBJECTID-OUTPUTGROUP=0
INPUTS=4
OUTPUTS=1
```

For more information, see *Mixers* (section 3.12)

2.10 REMOTESCONFIG configuration file

The driver can remotely control specified logic sources and keys, setting them to active/pressed or inactive/not-pressed. The logic sources and keys targeted for remote control must be defined in the REMOTESCONFIG file, under the sections as follows.

Up to 500 remote logic sources and 500 remote keys can be defined.

2.10.1 Remote Logic Sources

Remote logic sources are defined in the INI File under the `[RemoteLogicSources]` section and are in the format

```
nnnn=x
```

where `nnnn` is a number between 1 and 500 and `x` is the number of the logic source (from the LOGICCONFIG configuration file).

2.10.2 Remote Keys

Remote keys are defined in the INI File under the `[RemoteKeys]` section and are in the format

```
nnnn= Net.Node.Port.ExpansionPanel.PageNumber.Key
```

where `nnnn` is a number between 1 and 500. The key definition is in the same format as for a monitor key (see Section 2.3.2). The key address has no connection to those defined as monitor keys.

2.10.3 Sample REMOTESCONFIG INI file

```
[RemoteLogicSources]
0001=1
0002=34
0003=53
[RemoteKeys]
0001=1.3.122.0.1.22
0002=1.3.122.0.1.23
```

For more information, see

Remote Logic Sources (Section 3.14) and *Remote Keys* (Section 3.15).

2.11 VOIPCONFIG configuration file

The driver can remotely manage the VoIP related properties on specified client cards and ports. The client cards and ports targeted for remote management must be defined in the VOIPCONFIG file, under the sections as follows.

Up to 20 VoIP client cards and 160 VoIP ports can be defined.

2.11.1 VoIP client cards

VoIP client cards are defined in the INI File under the `[VoipClientCards]` section and are in the format

```
nnnn=x,y
```

where `nnnn` is a number between 1 and 20, `x` is the Artist node number and `y` is the number of the client card in the Artist node.

2.11.2 VoIP ports

VoIP ports are defined in the INI File under the `[VoipPorts]` section and are in the format

```
nnnn= x,y
```

where `nnnn` is a number between 1 and 160, `x` is the Artist node number and `y` is the number of the port in the Artist node, where ports start at 1.

2.11.3 Sample VOIPCONFIG INI file

```
[VoipClientCards]
0001=2,1
0002=2,2
[VoipPorts]
0001=2,24
0002=2,25
```

For more information, see *VoIP Client Card Properties* (section 3.19) and *VoIP Port Properties* (section 3.20).

2.12 PORTACTMONCONFIG configuration file

The driver can monitor the PortActive status of selected ports.

PortActive status is relatively useless for AIO ports, but can show if VoIP ports are connected, if MADI ports have a MADI signal, etc.

The ports targeted for such monitoring must be defined in the PORTACTMONCONFIG file as follows. Up to 200 ports can be configured for monitoring.

2.12.1 Monitored ports

Monitored ports are defined in the INI File under the `[PortActivityMonitor]` section and are in the format

```
nnnn= x,y
```

where `nnnn` is a number between 1 and 200, `x` is the Artist node number, and `y` is the number of the port in the Artist node, where ports start at 0.

2.12.2 Sample PORTACTMONCONFIG INI file

```
[PortActivityMonitor]
0001=1.1
0002=10.3
```

For more information, see *Port Active Monitor* (section 3.21).

3 Interface Functions

3.1 XY Crosspoints (GRD presentation)

The driver supports the setting of simple crosspoints. A crosspoint made on the GRD makes a permanent (i.e. not key- or vox- controlled) route from a source port to a destination port.

Crosspoints are single and unidirectional, in other words to route a bidirectional 4-wire you will need to make two crosspoints, one from source to destination and a second from destination to source.

Only one crosspoint may be active to each destination at any time. Therefore, when making a route to a destination anything previously routed to that destination will be automatically de-routed.

The presentation to BNCS for panel development is a standard GRD.

The PORTCONFIG configuration file defines the mapping between the GRD source/destination number and the Artist subscriber address. Up to 2000 subscribers can be configured for the XY interface.

3.1.1 Usage

A client panel should request that a route is made via the RouteCrosspoint command.

```
RC 660 1 15
```

3.1.2 Database Updates

Changes to the source and destination name databases (databases 0 and 1 respectively) are caught by the driver, but names on Artist panels, where visible, are only changed when re-asserted.

3.1.3 Revertives

The revertive is sent when the driver receives confirmation from the Artist router that the request to set the crosspoint was successful.

3.1.4 Crosspoint Multiplicity

Although the XY GRD presentation only permits a single crosspoint to be active to each destination at once, the driver only has control of crosspoints set through it. In other words, any crosspoints that are created by other functions within the Artist router (e.g. keyed “call to port” functions) will not be affected by the XY GRD function.

It is therefore strongly recommended that destinations to be controlled by the XY GRD function are not controlled by any other function, especially manually forced or set functions via Director.

3.1.5 Park Source

There is no park source as such. Instead, the BNCS driver treats a source of “0” as “off”. Therefore, in order to “park” a destination, simply route “0” to it.

3.1.6 Route Prohibition

The driver will not allow XY GRD routes to be made to destinations configured as input-only ports, panels or 4-wire destination monitors.

Similarly, XY GRD routes can only be made *from* ports configured as 2-wire inputs or 4-wire (split or normal) ports.

3.1.7 Route prohibition if member is listening to a conference

If the NOXYTOCONFMEMS flag is set in the INI file, the driver will not allow crosspoints to be made to a destination that is currently a listen member of a conference.

3.2 Monitoring (InfoDriver presentation)

Monitoring functions are presented via an InfoDriver.

The slots correspond to the monitoring destination index specified in the MONITORCONFIG configuration file, for example the monitoring destination defined in the INI file under section [MONITOR-0024] is controlled by slot 24 in the InfoDriver.

Due to the underlying differences between monitoring destination types, not all monitoring functions are available to every monitoring destination.

3.2.1 XY monitoring (type 4WD)

The most basic form of monitoring destination is a 4wire destination. This is the destination half of a standard 4wire subscriber port.

3.2.1.1 Usage

To monitor a 4wire **source**, write the BNCS source number (mapped in the PORTCONFIG configuration file) and an "s" character to the InfoDriver slot corresponding to your monitoring destination.

For example, to monitor BNCS source 143 to monitoring destination 24, write "`143,s`" to slot 24 of the InfoDriver.

```
IW 660 '143,s' 24
```

To monitor a 4wire destination, write the BNCS destination number and a "d" character to the InfoDriver slot corresponding to your monitoring destination. This feature is known as a "Port Clone". When activated, the Artist creates a copy of all crosspoints made to the monitored destination to the monitor*ing* destination.

For example, to monitor what is leaving BNCS destination 385 to monitoring destination 16, write "`385,d`" to slot 16 of the InfoDriver.

```
IW 660 '385,d' 16
```

Note: Port cloning is a CPU-intensive task within the Artist system so these destination monitors should be used sparingly. The driver limits the number of active port clones to 20.

To switch the monitoring off, simply write "`0`" or a blank string to the InfoDriver slot.

As with GRD destinations, anything previously monitored to a destination is cleared when a new monitoring request is sent.

3.2.1.2 Second Channel Monitoring

To monitor the second channel of a port, use "O" for source monitor or "E" for destination monitor. The driver only allows this if the target port has a second channel.

3.2.1.3 Monitoring/XY GRD Port Duality

The underlying Artist router function for the 4wire monitoring destination is the XY crosspoint.

Regular control of XY crosspoints is provided by the XY GRD, however the monitoring destinations provide additional functionality over and above what the GRD interface can support. For this reason 4wire destination ports can *either* be defined as XY GRD destinations *or* monitoring destinations, but not both.

3.2.2 Key configuration (type KEY)

The Key type monitoring destination provides advanced control over what commands are placed on a panel key. This function actually reconfigures the Artist to suit the request so changes made will be visible in Director.

Any number of commands can be placed on the key, although the length of the InfoDriver slot is 255 characters so the monitoring string must fit within this.

3.2.2.1 Usage

The string written to the InfoDriver is delimited list of options as follows:

`{FUNCTION}[,{FUNCTION}]*{MARKER}*{LABEL}*{KEYPROPERTIES}`

Each `{FUNCTION}` defines a single key function, and comprises of the following delimited list:

`FunctionName|Value|Flags`

The `FunctionName` is what you want to happen. `Value` contains the function's target. `Flags` contain modifiers that control various properties of the function.

3.2.2.2 Shortcut Processing

Instead of sending the whole delimited string, individual commands can be added or removed by using the following shortcuts:

Shortcut code	Description
&ADD={FUNCTION}	The key function {FUNCTION} is added to the key. Existing functions are left unchanged. If the function already exists, it is updated if necessary.
&REMOVE={FUNCTION}	The key function {FUNCTION} is removed from the key.
&LABEL={TEXT}	The key's label text is updated to {TEXT}. If {TEXT} is empty, the label is cleared and the Artist will display the default label for the command(s) on the key.
&MARKER={MARKER}	The marker assigned to the key is updated to {MARKER}. If {MARKER} is 0, the custom marker is cleared and the default marker for the command(s) on the key will be indicated.
&KEYPROPERTIES={KEYPROPERTIES}	The key's properties are updated with {KEYPROPERTIES}.

3.2.2.3 Function Types

The following values are permitted for the FunctionType value:

Item	Description and Flags detail
CALLTOPORT	Creates a "Call to Port" function on the key. Value: The BNCS port number to target. Flags: "U" enables the Riedel's "AutolistenFromDest" option which automatically creates a listen-to-port function on the key. "L" and "V" create an always- or vox-triggered (respectively) call-to-port command from the target port back to the key's panel.
TRUNKCALL	Creates a "Call to Port" function on the key targeting a trunked port. Value: The target trunked net and port number in the form "Net.Port".
CALLTOGROUP	Creates a "Call to Group" function on the key. Value: The group number to target (the number as defined in the GROUPCONFIG.INI file). Flags: "I" enables incoming call alert to the key.
LISTENTOPORT	Creates a "Listen to Port" function on the key. Value: The BNCS port number to target.
CONFERENCE	Creates a "Call To Conference" function on the key. Value: The conference number to target. Flags: "T" to give the panel talk privilege in the conference. "L" to give the panel listen privilege in the conference. At least one of T or L must exist or the function will be rejected.
IFB	Creates a "Call To IFB" function on the key. Value: The IFB number to target.

Item	Description and Flags detail
LOGIC	Creates a "Logic" function on the key. This triggers a logic source in the Artist. Value: The number of the logic source to target (the number as defined in the LOGICCONFIG.INI file).
SENDSTRING	Creates a "Send String" function on the key. Value: The string to send.

3.2.2.4 Second Audio Channel

Commands can be configured by additional flags to use the second audio channel of the source or destination, providing the target has a second audio channel configured.

The only port type that can have a second audio channel configured is a panel.

The flags controlling the second audio channel are "S" for the source's second channel and "D" for the destination's. It's not always obvious what the source and the destination are for a particular function, so the following table summarises:

Item	Source Second Channel Audio Flag "S"	Destination Second Channel Audio Flag "D"
CALLTOPORT	The command sends audio from the local panel's second audio channel. If the L or V flags are specified (always- or vox-triggered call from the destination port back to the panel respectively), that audio is routed **to** the panel's second audio channel.	The function targets the second audio channel on the destination port. If the L or V flags are specified (always- or vox-triggered call from the destination port back to the panel respectively), that audio is routed **from** the destination port's second audio channel. Only valid if the destination is a two-channel panel.
TRUNKCALL	The command sends audio from the local panel's second audio channel.	The function targets the second audio channel on the destination port.
CALLTOGROUP	Audio is routed to/from the local panel's second audio channel.	(not supported)
LISTENTOPORT	The function targets the second audio channel on the target port. Only valid if the target is a two-channel panel.	Audio is routed to the local panel's second audio channel.
CONFERENCE	Audio is routed to/from the local panel's second audio channel.	(not supported)
IFB	Audio is routed to/from the local panel's second audio channel.	(not supported)
LOGIC	(not supported)	(not supported)
SENDSTRING	(not supported)	(not supported)

3.2.2.5 Label

The label is the text that is displayed on the key. If this is empty, they key's text will be automatically generated by the Artist.

3.2.2.6 Marker

Markers are the pattern of LEDs above the key. Up to 255 markers can be defined (some are defined by the system and cannot be changed); these are system-wide and are configured using Director. Each marker should have a unique number, with lower numbers taking greater priority.

When a marker is placed on the key the Artist adds it to the stack of marker(s) already present. If the newly-added marker has a lower priority than those already there, the new marker will not be displayed. Therefore any custom markers should be defined at high priorities.

3.2.2.7 Key Properties

The driver allows manipulation of some properties of the key.

The properties are presented as a pipe-delimited set of property statements, for example:

```
KM|D0
```

Only one value of each property type is permitted.

Key Property Type	Value	Description
K	M	Key Mode -> Momentary. Sets the key mode to momentary action.
	L	Key Mode -> Latching. Sets the key mode to latching.
	A	Key Mode -> Auto. Sets the key mode to automatic, which determines the mode based on what commands are present.
D	1	Speaker Dim -> Yes. The panel speaker is dimmed while the function is enabled.
	0	Speaker Dim -> No. The panel speaker is *not* dimmed while the function is enabled.

The property string can be blank, in which case a monitoring request will leave the existing properties unchanged.

3.2.2.8 Clear All

Writing "0", an empty string or "&CLEAR"* to the slot clears all commands from the key. This can also be used to clear a stuck command from a key, should that happen (note: this should not happen, and that the driver sync process should also clear stuck commands).

* Please note that "&CLEAR" is more destructive and will remove absolutely all commands from the key, including those placed there by Director. "0" and an empty string will only remove commands from the key placed there by RRCS/BNCS.

3.2.2.9 Default Clear-Label

If the key is cleared using the Clear All command AND the MONCLEARLABELDB INI file setting is configured with a valid database number, the key will be labelled with the name taken from the corresponding database entry.

For example, MONCLEARLABELDB is set to "2" in the INI file. Monitoring destination 5 is cleared using the Clear All function:

```
IW 602 '0' 5
```

The driver removes all commands from the key, and labels the key with the name taken from device 602, database 2, index 5 (".noname."). The revertive will include this name:

```
IR 602 '*0*.noname.' 5
```

This function is intended to allow default placeholder names on keys when no commands are present, in order to identify the keys as "under BNCS control" rather than "unused".

Note: If the key is cleared by writing the full monitoring string, e.g "*0*", the name is *not* set to its default. In this case it is up to the panel to send the name with the string, e.g. "*0*.noname.".

The name is quietly truncated at 8 characters. Database changes are caught while the driver is running, although the key will not show the new name until it is next cleared.

3.3 PTI (InfoDriver presentation)

The Artist router is complicated and a source can be feeding a variety of destinations via a number of interfaces. For example, a source could be routed to a destination on the XY GRD and also be routed to a monitoring destination. To facilitate panel development, the PTI InfoDriver presents all forward routes for every source and contributing sources destination in a standard format.

3.3.1 Slot Allocation

Slots 1 to 2000 contain the forward route indication (the true meaning of "push to indicate") for the ports treated as a source (i.e. input to something).

Slots 2001 to 4000 contain the contributing source indication for the ports treated as a destination (i.e. the output from something).

3.3.2 Slot Contents (Forward Route Indication)

The data in the slot is presented as a set of comma delimited fields which are in turn delimited with pipe characters. Each pipe-delimited field is the PTI for a particular interface function as follows.

```
XY GRD|Monitoring|Conference(as talk)|IFB(mix minus)|Mixer(input)
```

For example, if source 5 is routed to destinations 1, 2, 4 and 5 via the XY GRD, routed to monitoring destination 4, is a talk member of conference 6, is the mix-minus input to IFB 7 and is not in the input group for any mixers, slot 5 will contain:

```
1,2,4,5|4|6|7|0
```

If the source is not routed anywhere via a particular interface function, the PTI string for that function will contain "`0|0|0|0|0`".

3.3.3 Slot Contents (Contributing Source Indication)

The data in the slot is presented as a set of comma delimited fields which are in turn delimited with pipe characters. Each pipe-delimited field is the contribution from a particular interface function as follows.

```
Conference(as listen)|IFB(as output)|Mixer(output)
```

For example, if port 5 is a listen member of conference 1, is in the output group of IFB 5 and is not in the output group of any mixer, slot 2005 will contain:

```
1|5|0
```

If the port is not a listen member of any conference, the slot will contain "`0|0|0`".

3.3.4 Port Clear

You can clear all references to a given port with a single client request.

This is triggered by writing to the forward route indication slot corresponding to the port you wish to clear.

Command	Purpose
&CLEARSCE	Clears the port from places it appears as a source: • From XY routes where it is the source. • From all conferences in which it has talk privileges†. • From all IFBs where it appears in the Input or Mix-Minus groups. • From all keys on which it is the target of a Listen-To-Port. • From all mixers where it appears in the Input group.
&CLEARDEST	Clears the port from places it appears as a destination: • An XY route to the port is removed. • From all conferences in which it has listen privileges†. • From all IFBs where it appears in the Output group. • From all keys on which it is the target of a Call-To-Port. • From all mixers where it appears in the Output group.
&CLEAR	Clears the port from all of the above.

† If the port is in a conference with talk and listen privileges, either &CLEARSCE or &CLEARDEST will remove the port completely; it will not adjust the port's existing membership.

3.4 Port Aliases (InfoDriver presentation)

An Alias can be applied to each port in the Artist (whether a subscriber or a panel), and is used as the display text whenever that port is monitored to a panel.

3.4.1 Slot Allocation

The slots in the Port Aliases InfoDriver correspond to the sources defined in the XY GRD.

Source port aliases start at slot 1. Destination port aliases start at slot 2001.

For ports configured as bi-directional (4-wire ports or panels), the Artist system only supports one alias. For these ports the source port alias is used.

3.4.2 Slot Contents

The data in the slot is simply a string of text which has a maximum length of 8 characters. Strings longer than this are truncated. Write an empty string to clear the Alias (in which case the Artist will display the configured ShortName on the key).

A change of Alias is reflected on all panel keys more or less immediately.

3.4.3 Database Link

If the LINKDBTOPORTALIAS INI file setting is set to YES, the driver will automatically populate the ports' aliases with names taken from the BNCS XY GRD's source and destination name databases. The names are synchronised during a driver synchronisation process, and the driver responds to database changes received while it is running.

3.5 General Purpose I/O (InfoDriver presentation)

The Artist router can be equipped with General Purpose Inputs and Outputs (GPIs or GPOs). The GPIO interface is presented to BNCS via an InfoDriver.

The GPICONFIG configuration file defines the mapping between the slot number in the InfoDriver and the Artist GPIO addresses.

3.5.1 Slot Allocation

GPIOs are mapped according to the entries in the GPICONFIG configuration file.

GPIs start at slot 1. GPOs start at slot 2001.

3.5.2 Slot Contents

The InfoDriver slot will contain "1" if the GPIO is asserted or "0" if it cleared. When writing, "1" is treated as an assert and anything else is treated as a clear.

Slot contents are updated with the live status of the GPIOs from the Artist, though this might not be instantaneous as it will depend on how busy the Artist controllers are.

3.6 Input/Output Gain (InfoDriver presentation)

The Gain InfoDriver provides an interface to the input and output subscriber gains.

3.6.1 Slot Allocation

The slots in the Gain InfoDriver correspond to the sources defined in the XY GRD.

Source gains start at slot 1. Destination gains start at slot 2001.

3.6.2 Slot Contents

The slots contain the numeric value of the gain of the associated port. The range is -18.0 to +18.0dB and can be adjusted in 0.5dB steps. "MUTE" is also acceptable. The units are omitted, as is the '+' symbol for positive gains.

3.7 Conferences (InfoDriver presentation)

The conferences InfoDriver is split up into blocks of 1000 slots. Each block has a different function and within each block, each conference appears at the same index. For example, Conference 1 (as defined in the INI file) uses slots 1, 1001 and 2001.

3.7.1 Slot Allocation

Slot (offset)	Purpose
1 to 350	**Conference members**, see below.
1001 to 1350	**Conference label string** This is the string that appears on panel keys when they are configured to monitor this conference. It can be formed of any characters and can be a maximum of 8 characters in length (it will be truncated if necessary). Note: A changed label string is only sent to the Artist when the members of a conference change. If only changing the label, you should re-send the conference member string to the members slot after writing the label to the label slot. If you are changing the label and the members, change the label first.
2001 to 2350	**Panel members**, see below.

3.7.2 Conference Member String

Slot 2 of the InfoDriver block contains the member string for the conference. This is a comma delimited string of members and conference settings.

Within the comma delimited string, individual conference members are pipe delimited, and are in the following arrangement:

```
Port|Privilege|ProtectionGroup
```

`Port` is the BNCS port number from the XY GRD. `Privilege` is whether the member should join the conference with Talk privilege (`T`), listen privilege (`L`) or both (`TL`). Note that if neither `T` or `L` is present, the member will be ignored.

The `ProtectionGroup` parameter is much the same as the *Room Code* function in the Artist. Usually, in a conference, a given subscriber hears audio from all the other

conference subscribers. By placing two (or more) subscribers in the same protection group, they will not hear audio from each-other. This function can be used, for example, where there are a main and a backup audio circuit and it would be confusing if audio spoken on one returns from the other. Alternatively, another use could be where the go and return legs of a subscriber are on different Artist subscriber ports (yet both ports are not configured as split ports in the Artist configuration). The protection group number is unique per conference and can be between 1 and 9, or 0 to disable the feature.

For example, the following member string defines a conference with three members: Port 1 has full talk and listen privileges, port 8 can talk only (bit odd but technically allowed) and port 17 can only listen.

```
1|TL|0,8|T|0,17|L|0
```

3.7.2.1 Shortcut Processing

Instead of sending the whole delimited string of conference members, ports can be added or removed by using the following shortcuts:

Shortcut code	Description
&ADD={PORT}	The port {PORT} is added to the conference with Talk and Listen privileges (and is not placed in a protection group). If the port already exists with lesser privileges, the privileges will be upgraded to Talk/Listen.
&REMOVE={PORT}	The port {PORT} is removed from the conference if it was present.
&REMOVE=EVERYTHING	All ports are removed from the conference. Also, all keys with a call-to-conference command targeting the conference are removed.

3.7.2.2 Re-send members

As mentioned previously, the label is only updated when the member string is updated. If you only wish to change the conference label your panel may not necessarily know what the members of that conference should be.

To avoid having to poll for the member string first, your panel can simply write "&RESEND" to the conference member string slot. This has the same effect as writing the member string back to the slot without changing it, and will update the label if necessary.

3.7.2.3 Vox members

By default, conference membership is added as an "always" command in the Artist. This means that audio from member port(s) with Talk privileges is *always* routed to member ports with Listen privileges. This works fine but does result in crosspoints in the router that may not strictly be necessary.

The conference membership command can, instead, be added as a "vox" command in the Artist. This means that audio is only routed through the conference when activity is detected on the talking ports. Crosspoints are therefore saved.

To specify a member as a vox member, supply the conference member string with the `V` flag as well as the `T` and/or `L` flag(s). Clearly there's no point specifying the flag combination `LV`, because the vox concept only applies to incoming (`T`) audio.

For example, the following member string defines a conference with three members: Port 6 has full talk and listen privileges, port 55 can talk only (bit odd but technically allowed),

with the crosspoint(s) only made if/when there is vox activity on the port, and port 109 can only listen.

```
6|TL|0,55|TV|0,109|L|0
```

3.7.3 Panel members

Artist panels can be members of a conference; however they are added through the Monitoring interface (by creating a call-to-conference function on a panel key).

The panel members slot contains a read-only comma-delimited list of panels* which are members of the conference. This is updated whenever the monitoring interface command creates or removes a call-to-conference command.

*The number displayed for the panel is that panel's corresponding port in the XY GRD. See section 2.3.5 for further information.

For example, the following panel member string indicates that a call-to-conference command exists on at least one key on each of the panels mapped to the XY GRD ports 101 and 102.

```
101,102
```

3.7.4 Multi Membership

A port can be a member of as many conferences as it likes (although there's almost certainly a limit within the Riedel that you'll hit eventually).

3.7.5 Performance Note

Conference membership is changed using the configuration change system. A new membership string triggers a configuration change request which can take some time to complete. During this time you will be unable to make further changes to the conference membership (although the command(s) are queued by the driver). If the panel is changing multiple members, therefore, it is better to send one message to the Artist driver with all the changes at once.

3.7.6 Additional Debugging

Ports are added to and removed from conferences by the driver creating/removing "call to conference" commands on the "always" or "vox" virtual functions of the port(s). In regular operation, the driver should keep in sync with what is on the port in 'real life' and all will be well.

Should there be problems adding or removing conference members, an additional debugging function can be enabled by setting the DEBUGCONFPORTS INI file setting accordingly.

This function interrogates the Artist for the complete command list for the port(s) affected during a conference operation. The output of the query is logged to the debug log for later analysis, but the idea is that you can see for certain that call-to-conference commands are created and removed from the intended port(s) during the conference operation.

3.8 IFBs (InfoDriver presentation)

An IFB is entirely defined in a single InfoDriver slot, so IFB 1 is configured through slot 1 of the IFB InfoDriver, etc.

3.8.1 IFB member string

The member string for an IFB looks like:

```
Input port(s)|Mix minus port(s)|Output port(s)|Label
```

All ports are entered as BNCS port indices corresponding to the XY GRD mapping. If specifying multiple ports, the ports should be comma-delimited.

3.8.1.1 Input Ports

The source port(s) a panel key should listen to when configured to call to this IFB. Also known as *IFB Listen Source(s)*.

3.8.1.2 Mix-Minus Input Ports

The source(s) that feed the IFB destination until interrupted by an operator talking via a panel. Also known as *Clean Feed*.

3.8.1.3 Output Ports

The destination port(s) to which the IFB should be sent.

3.8.1.4 Label

The text string that appears on panel keys when they are configured with a call to this IFB. It can be formed of any characters and can be a maximum of 8 characters in length (it will be truncated if necessary).

3.8.2 Usage

The member string should be written to the InfoDriver. For example:

```
IW 661 '1|2|3,4,5|ManUvEvt' 12
```

This configures IFB 12 to have the label "ManUvEvt", port 1 as the input, port 2 as the mix-minus source and ports 3, 4 and 5 as outputs.

Note: If this IFB is also targeted by a mixer, the mix minus and output ports cannot be set through the IFB interface.

3.8.3 Shortcut Processing

Instead of sending the whole delimited string, individual elements of the IFB can be manipulated by using the following shortcuts:

Shortcut code	Description
&INPUT={PORT(S)}	The input member(s) are changed to the new one(s).
&INPUTADD={PORT}	The port is added to the current input members(s).

Shortcut code	Description
&INPUTREMOVE={PORT}	The port is removed from the current input members(s).
&MIXMINUS={PORT(S)}	The mix-minus member(s) are changed to the new one(s).
&MIXMINUSADD={PORT}	The port is added to the current mix-minus members(s).
&MIXMINUSREMOVE={PORT}	The port is removed from the current mix-minus members(s).
&OUTPUT={PORT(S)}	The output member(s) are changed to the new one(s).
&OUTPUTADD={PORT}	The port is added to the current output members(s).
&OUTPUTREMOVE={PORT}	The port is removed from the current output members(s).
&REMOVEALL={PORT}	The port is removed from input, mix-minus and output member(s).
&LABEL={LABEL}	The label is modified.
&REMOVE=EVERYTHING	All ports are removed from the IFB. Also, all keys with a call-to-IFB command targeting the IFB are removed.

Note: If modifying a group, all members of the group need to be specified at once. It is not possible to add/drop individual members from the group.

3.8.4 Split IFB Shortcut Commands

By default, when shortcut processing commands are used, the Artist is only updated with the property has changed. For example, if the label is changed with "&LABEL=xxx", the input, mix minus and output properties are left entirely alone, permitting them to be set via alternative means.

If this is not desirable, and the driver should instead fill in the properties that were not specified in the shortcut processing request before updating the Artist, configure the SPLITIFBSHORTCMDS setting to NO.

3.8.5 Fighting Destinations

As all ports are defined in the XY GRD it may be tempting to make an XY route to the same destination that has an IFB assigned to it. This will work and the Artist will mix the audio leaving the output port. Confusion may result!

3.9 Extras (InfoDriver presentation)

The Extras InfoDriver exists to mop up all the little functions that didn't deserve an InfoDriver of their own. The functions are detailed in their own sections, but this table serves to show which slots in the Extras InfoDriver do something.

Slot	Function	See section for further details…
1	*Send String* output slot.	3.10
2	*Configuration re-send* function.	4.1.3
3	*Sync alert* function.	4.1.4
4	*Crosspoint Adjust* function.	3.11
5	Workstation number of the driver currently in TxRx (cleared on RxOnly workstation).	
6	*Running Status* display.	3.17
101-120	Alarm messages.	3.16.1
201-220	Node online status indication.	3.16.2
301-325	Port Probe.	3.13
400-499	*Trunkline activity* function.	3.18
501-1000	Mixers.	3.12
1001-1500	Remote Logic Sources.	3.14
1501-2000	Remote Keys.	3.15
2001-2020	VoIP client card properties.	3.19
2021-2180	VoIP port properties.	3.20
2201-2399	PortActive Monitors	3.21

3.10 Send String

Send String is an Artist key function that can be created on a key using the Monitoring interface. Simply, whenever the key is pressed the pre-configured string is sent to BNCS. No information on which panel or key sent the string is delivered, so the string should contain this information explicitly if needed.

The last string sent is stored in the Extras InfoDriver, slot 1. This is cleared during a router synchronisation.

For example:

```
01-Dec-2010*10:58:03|This is the string that BNCS wrote..
```

3.10.1 Timestamp

By default, the time a message is received from the Artist is pre-pended to the message string. To disable this, use the TIMESTAMPSENDSTR INI file setting.

3.11 Crosspoint Adjust

The Crosspoint Adjust function enables low level control of any crosspoint in the Artist.

To use, write to the Extras InfoDriver, slot 4 in the following format:

```
Command|Source,Destination[|Volume]
```

The Command is "SET", "KILL" or "VOLUME".

If "VOLUME", you must additionally specify the `Volume` parameter. `Source` and `Destination` are the BNCS port numbers between which lies the crosspoint that needs adjusting.

To specify the second audio channel of a port, append "`:2`" to the port number. Note this does *not* check whether the port truly does have a second channel of audio configured, it only serves to target the Artist port number that is one port higher than that which is mapped to the port. Second channel can only be requested if the Artist port is an even number; remember the Artist numbers ports starting at 0 so the first port of a two-channel port will always be 0, 2, 4, etc.

No revertive comes from the Crosspoint Adjust slot because there is no easy way to show the status of all 1024x1024 crosspoints in BNCS. This, therefore, is a one-shot "set and forget" command that is only for use where desperately needed.

For example, to set a crosspoint between ports 55 and 207:

```
SET|55,207
```

To mute the volume between ports 328 and 910 (perhaps 910 is a panel and you want to effectively press the encoder to mute a listen):

```
VOLUME|328,910|MUTE
```

The volume is specified as a dB value from MUTE to 12.5 (dB).

Note: This command sets the *single volume* of the crosspoint, not the *conference volume*. Conference volumes are 'intelligently' handled by the Artist and setting the conference volume of a single crosspoint in a conference can lead to unexpected results so is not supported.

3.12 Mixers

The Artist can provide very simple audio mixing capabilities through the BNCS mixer interface. This allows for a number of sources to be routed to a number of destinations simultaneously, and also supports independent volumes for each input.

Up to 500 mixers can be defined (see *MIXERCONFIG configuration file*, section 2.9).

Slots 501 to 1000 of the Extras InfoDriver correspond to the configuration slots for the 500 mixers.

3.12.1 Slot Contents

The input ports, volumes and output ports are all delimited into one slot per mixer in the following format:

```
InputPort:Volume[,InputPort:Volume…]|OutputPort[,OutputPort…]
```

All ports are XYGRD ports, though the driver does not allow input-only ports to be added to the output side of the mixer, and vice-versa.

The volumes are specified as dB values from MUTE to 12.5 (dB).

The number of entries in the input and output strings is controlled by the mixer's INPUTS and OUTPUTS properties in the configuration file and is a fixed value. If the string sent to the mixer has more ports than are configured, the surplus ports are ignored. Conversely, if the string has less ports, zero-placeholders are added.

For example:

```
121:0.0,122:MUTE,0:MUTE,133:10.0|501,503
```

This mixer is configured with 4 inputs and two outputs. Port 121 is in the input group at 0.0dB, port 122 as mute, the third entry is blank, and then port 133 is at +10dB. The output of the mixer is routed to ports 501 and 503.

3.12.2 Shortcut Processing

The mixer function also supports the following shortcut commands:

Shortcut code	Description
&INPUT#={Port}	Set the port for mixer input #.
&INPUT#={Port:Vol}	Set the port for mixer input # and also set its volume.
&INPUT#={*vol*:Vol}	If the literal string "*vol*" is specified in place of the port, the driver only applies a volume change.
&OUTPUT#={Port}	Set the port for mixer output #. Note # must be specified even if there is only 1 output.
&REMOVEIN={Port}	The port is removed from the mixer's input, without concern for which input # it's at.
&REMOVEOUT={Port}	The port is removed from the mixer's output, without concern for which output # it's at.
&REMOVE={Port}	The port is removed from the mixer's input and output, without concern for which input or output # it's at.

3.12.3 Unwanted mixing

Care should be taken to avoid duplicating crosspoints. If a pair of ports is placed in a mixer, do not also make an XYGRD crosspoint between them. The driver will happily allow this but you may find odd volumes resulting. So just don't.

3.13 Port Probe

The Port Probe function examines a specified port (audio input/output port or pool port) and reports what commands are present on its *always* and *vox* virtual functions.

The command list is queried from the Artist and therefore contains all commands, not just those created by the driver.

The list is automatically updated if the driver believes the commands on the port have changed. (The driver only creates or removes commands from a port when adding or removing that port to/from a conference.)

Note: The Port Probe function only runs on the Working driver. The command detail slots on the Standby driver will be blank (and are refreshed when the driver switches from Standby to Working).

3.13.1 Slot layout

Slot	Function
301	The port to target by the Port Probe. This is a port index from the XY GRD and must correspond to an audio or pool port.
302	The number of commands found on the port's virtual functions.
306-325	20 slots, containing the detail of the command(s) found. One command per slot. If more than 20 commands are on a port's virtual functions, only the first 20 will be shown. Write "&DELETE" to remove manually the command from the port.

3.13.2 Deleting commands

A single command can be manually removed from a port by writing "`&DELETE=x`" to the slot in which that command is described, where `x` is the port number that is the target of the Port Probe (this is a check to ensure the Port Probe is still looking at the port the operator intends before a command is deleted).

The command list is automatically refreshed after the deletion so the success (or not) of the request will soon be visible.

Some command types cannot be removed by the driver but all common ones (such as *call-to-conference* and *DimXP* commands, the only sorts that the driver is likely to place on a port's virtual functions) can be.

Care MUST be taken with this command. It is an *engineering tool* only. No checking is done before the command is removed. It is entirely possible to remove a call-to-conference command that is legitimately joining a port to a conference. The Conference interface's InfoDriver will still show that the port is in the conference even though, in the Artist, it is not. The Port Probe function is designed to track down erroneous or stuck commands and to permit their manual removal.

3.14 Remote Logic Sources

Logic sources are intangible objects inside the Artist. A logic source can be set to either on or off, and acts as an input into the Artist's internal logic functions. The logic source can be routed to a logic destination, and commands placed on that destination. This is a way of providing neat trickery!

The configuration of logic destinations, commands and source to destination routing is achieved through the Director software.

The Driver allows the state of pre-defined logic sources to be remotely controlled.

Up to 500 remote logic sources can be defined (see *REMOTESCONFIG configuration file*, section 2.10).

Slots 1001 to 1500 of the Extras InfoDriver are mapped directly to these logic sources.

Write "`1`" to set the logic source to "on". Write "`0`" for "off".

The InfoDriver will send the revertive on a successful change of logic source state. Note this information does not come from the Artist, so if a logic source is asserted from within the Artist, the driver will not show the status.

3.15 Remote Keys

In a similar way to remote logic sources, keys predefined for remote control can be 'pressed' or 'released'.

Up to 500 remote keys can be defined (see *REMOTESCONFIG configuration file*, section 2.10).

Slots 1501 to 2000 of the Extras InfoDriver are mapped directly to these keys.

Write "`1`" to press the key. Write "`0`" to release it.

3.16 Rudimentary Alarms

3.16.1 Alarm Messages

The Artist is aware of errors in its running configuration, for example if a client card has failed. When an alarm condition develops a notification is sent to the driver. The notification system is not especially comprehensive but does serve as an indication of trouble.

The last 20 alarm messages are stored in slots 101 to 120 of the Extras InfoDriver. The time at which the event occurred is pre-pended to the string. The slots are allocated in a round-robin system and are re-set to slot 101 at driver start up.

In the following example, the alarm is indicating that the upstream fibre interface from the controller card in Node 2 has failed.

```
01-Dec-2010*10:58:03|UpstreamFailed (Node: 2)
```

3.16.2 Node Online Check

As part of the alarm notification mechanism, the driver received events informing of Artist node controller failures. These notifications are used to keep track of the online status of node(s) in the Artist ring.

The ONLINENODEMONITOR INI file setting controls this function. If set to `0`, the function is disabled. Otherwise, the setting should be set to a string containing the Artist node numbers whose online status should be tracked.

The online status of the tracked nodes is stored in the Extras InfoDriver in slots 201 upwards. Slot 201 is the status for the first node in the INI file string, slot 202 is the second, etc.

A value of `1` suggests the node is online. `0` suggests it is offline.

3.17 Running Status

The running status of the driver is presented in slot 6 of the Extras InfoDriver. This includes a timestamp and one of the following string describing the operational mode of the driver.

String	Meaning
Connected to BNCS	The driver has connected to the BNCS host drivers but has yet to start communicating with the Artist.
Artist Starting	The driver has connected to the Artist and is performing various start up tasks.
Artist Failure	The driver was not able to connect to the Artist at start-up.
Syncing	The driver has connected to the Artist and is performing its router sync.
Ready	The driver has connected to the Artist and is ready for use having performed its start-up sync if required.
Closed	The driver has closed and disconnected from the BNCS host drivers.

3.18 Trunkline Activity Probe

If the INI file setting NETTRUNKADDRESS is set to non-zero, a snapshot of trunkline usage can be obtained from the Artist.

Write "`Update`" to Extras InfoDriver slot 400 to trigger the driver to query the Artist for the trunkline usage *at that instant*.

Note, no counts are kept nor are any averages calculated. This is simply a view of what the trunklines are doing at this moment.

The driver is only concerned with trunk lines leaving the Artist ring to which it is connected. Each of these trunk lines is represented in a slot between 401 and 499.

The contents of this slot are:

```
DestNetTrunkAddr|DestPortTrunkAddr|TrunkInUse
```

`TrunkInUse` can be `0` (not in use) or `1` (in use). If in use, the usage is delimited into the slot following a `*`.

3.18.1 Trunkline Usage

A trunk can be used for one or more calls. Each call using a trunk line is listed as a comma-delimited set of pipe-delimited numbers.

```
TrunkCommand|SourcePort
```

Where `TrunkCommand` identifies the active call type and `SourcePort` identifies the BNCS index of the port that initiated the command.

3.18.2 Examples

Slot Contents	Meaning
2|3|0	This is a trunk line to Trunked Net 2, on its port 3, and it is not currently in use.
2|3|1*1|589	This is a trunk line to Trunked Net 2, on its port 3, and it is currently in use for a single call-to-port command initiated from the port at BNCS index 589.
2|3|1*4|4,4|589	This is a trunk line to Trunked Net 2, on its port 3, and it is currently in use for two call-to-conference commands initiated from the ports at BNCS index 4 and 589.

3.19 VoIP Client Card Properties

The driver can manage the VoIP properties of VoIP-108 G2 client cards.

The client cards to be managed are assigned to Extras InfoDriver slots in the 2001 to 2020 range in accordance with the `VoipClientCards` mapping section in VOIPCONFIG.INI.

In the InfoDriver slot the VoIP parameters are presented as key-value pairs.

3.19.1 Client card key-value pairs

Key	Value	Description
ia	0 or 1	Obtain IP address automatically, yes or no
da	0 or 1	Obtain DNS address automatically, yes or no
i	x.x.x.x	IP address e.g. 192.168.42.120
m	x.x.x.x	Subnet mask e.g. 255.255.255.0
g	x.x.x.x	Default gateway e.g. 192.168.42.1, or blank
dp	x.x.x.x	Primary DNS server, or blank
ds	x.x.x.x	Secondary DNS server, or blank
h	Hostname string	Client card DNS hostname (up to 30 characters, a-z, A-Z, 0-9 and hyphen)
p	1024-65535	TCP port

3.19.2 Usage

To change a parameter, write its key value pair to the InfoDriver slot. The driver then executes a Configuration Change to effect the change.

Multiple key value pairs can be written simultaneously and get included with the one change, this is the preferred way to change multiple parameters at once.

3.19.3 Client card slot example

```
ia=0,da=0,i=192.168.42.160,m=255.255.255.0,g=192.168.42.1,dp=,ds=,
h=client01,p=5060
```

3.20 VoIP Port Properties

The driver can manage the VoIP properties of ports.

The ports whose VoIP properties are to be managed are assigned to Extras InfoDriver slots in the 2021 to 2180 range in accordance with the `VoipPorts` mapping section in VOIPCONFIG.INI.

In the InfoDriver slot the VoIP parameters are presented as key-value pairs.

3.20.1 Client card key-value pairs

Key	Value	Description
h	x.x.x.x	IP address of remote SIP host e.g. 192.168.42.160
s	string	Remote SIP ID (up to 30 characters) e.g. “port2”
l	string	Local SIP ID (up to 30 characters) e.g. “port2”
a	Enumerated 0-9 (?)	Audio codec. Experiment with Director to find out what the enumeration means
k	Enumerated 0=20ms 1=40ms 2=80ms 3=160ms	Audio packet size in ms
f	Enumerated 0=80ms 1=160ms 2=320ms 3=640ms 4=1280ms 5=2560ms 6=5120ms	Audio buffer size in ms Note the minimum value permitted is 4 * packet size; the Artist silently sets the buffer size to this if the driver requests it smaller.
v	0 or 1	Voice activity detection, on or off.

3.20.2 Usage

To change a parameter, write its key value pair to the InfoDriver slot. The driver then executes a Configuration Change to effect the change.

Multiple key value pairs can be written simultaneously and get included with the one change, this is the preferred way to change multiple parameters at once.

3.20.3 Port slot example

```
h=192.168.42.160,s=port2,l=port2,a=0,k=20,f=160,v=1
```

3.21 Port Active Monitor

The driver can monitor the PortActive status of selected ports.

The ports to be monitored are assigned to Extras InfoDriver slots in the 2201 to 2399 range in accordance with the `PortActivityMonitor` mapping section in PORTACTMONCONFIG.INI.

3.21.1 Slot contents

Slot Contents	Meaning
(empty)	No PortActive status is known.
0	The port was reported as OFFLINE.
1	The port was reported as ONLINE.

3.21.2 Nature of the PortActive notifications

RRCS only delivers the driver a notification of a change in port status when that status actually changes. There is no way to determine the state of a port on driver startup. As such the data contained within these slots is not cleared on startup nor refreshed during a driver sync.

On a port whose status regularly changes (e.g. a VoIP port) this won't matter as an updated status will be notified soon enough. On ports whose status rarely changes (e.g. on a MADI port) it may make this function less than completely useful.

3.21.3 Resetting a notification

It is possible to clear the last-reported notification by writing "&CLEAR" to the slot. The contents will then be cleared awaiting the next notification from RRCS.

4 Advanced Features

4.1 Router Synchronisation

4.1.1 Full Synchronisation

The driver includes a synchronisation function to ensure the Artist router configuration matches that which is represented in the tally tables of the various BNCS host drivers.

The synchronisation function is always run on driver start-up, and can be optionally triggered when the driver's mode switches from RxOnly to TxRx.

Due to the complexity of the remote control interfaces there are a large number of requests made to the Artist router during a synchronisation. Therefore the synchronisation function takes a while to complete. During this time the driver will not respond to new requests from BNCS. These requests will not be queued and therefore will not be actioned.

The synchronisation of a complex system may take upwards of a minute.

4.1.2 Synchronisation Errors

When the synchronisation function completes it summarises any errors and warnings it encountered.

4.1.2.1 Sync Errors

An error is logged if the driver could not send a batch of configuration changes to the Artist. This is a major error because it means a large number of configuration items (conferences, for example) may not be correctly initialised. The most likely outcome of this failure is stuck conference members, or things stuck on panel keys.

The debug log should be inspected for the error and the situation resolved as soon as possible.

4.1.2.2 Sync Warnings

A warning is logged if the driver could not change something during the synchronisation, but that thing is not operationally critical. Port gains fall into this category: often the Artist is incorrectly configured and the gain can not be set by the driver.

4.1.3 Mini Synchronisation (configuration re-send)

If, for whatever reason, the online configuration across multiple Artist nodes gets out of step, it may be necessary to effectively re-send the correct configuration into the Artist system.

Note that this should not happen in regular operation, but may occur if a node is offline or reboots while a configuration change is in progress.

Write "`&RESEND`" into slot 2 of the Extras InfoDriver to trigger.

4.1.4 Sync Alert

As has been mentioned, the synchronisation function is essential and restores the Artist router to a known state so the driver can control it successfully. Due to the limitations of the Artist router's API, the synchronisation function is intrusive and will result in commands vanishing from keys for a few seconds before they're restored. Obviously, if this were to happen when the system is being heavily used the operators would get a shock, OBs might lose comms to the studio, and all manner of badness would ensue.

By default, a router synchronisation will be triggered when an RxOnly driver takes over as TxRx. This can theoretically happen at any time, so to prevent an intrusive synchronisation happening at any time the SYNCONCO INI file setting can be used to stop this automatic synchronization. However, in this case the Artist router may end up running with a slightly different configuration to that which the driver expects. This can cause problems with commands being unable to be placed on or removed from keys, or stuck crosspoints, and other funnies.

To provide an alert that the driver should be manually synchronised with the router, slot 3 of the Extras InfoDriver contains a sync alert function.

In regular use this slot will be blank. If the driver changes over and is stopped from performing a synchronisation through the SYNCONCO INI file setting, a timestamp and alert will be written to this slot. For example:

```
2011-03-01 11:54:22 – Driver Changeover RxOnly->TxRx. Advise driver sync ASAP!
```

A full synchronization can be then manually triggered at a convenient time by writing "`&SYNC`" to the slot.

The slot is cleared during a synchronisation.

4.2 Keep Alive

Once the driver has connected successfully to RRCS, a keep-alive timer is started. Every 15 seconds this makes a test call to RRCS to check it is still responding correctly.

If this call fails twice in a row, the driver concludes that RRCS has failed in some way. In an attempt to recover, the driver sets itself to RxOnly, in the hope that another instance of the driver is running on the network and will take control. The driver then terminates its message processing function and attempts to re-connect to RRCS.

5 Notes

5.1 Node Controller IP Addresses and how RRCS Connects

Artist frames are designed to have dual node controller cards. The two cards decide between themselves which one is in control and which one is the standby. There is no way to manually change this and to all intents and purposes it does not matter.

There is a designed-in limitation around the IP addresses of the node controller cards: The primary card must be assigned an IP address with an even number as the last octet, and the secondary card must be one higher. For example, if the primary card is configured with the IP address 10.1.8.12, the secondary card can only be assigned 10.1.8.13.

When the IP address is configured in RRCS, the address of the primary controller card is entered and you tick the checkbox labelled "This is an Artist with a redundant node controller card". RRCS then auto-populates the IP address of the secondary controller.

Both node controller cards must be connected to the network and RRCS must have communication with both of them. During a connection attempt, RRCS first attempts to connect to the primary controller card and asks for control. If the primary controller card is the active one, control is granted immediately. If the node is running with the secondary controller as the active one, RRCS will not be granted control and will wait 2 seconds before timing out and attempting to connect to the secondary controller.

If RRCS cannot connect to the primary controller card at all, it stalls at the first step and will never try to connect to the secondary controller card.

5.1.1 Concurrent Connections

Each Artist node can support up to a maximum of 4 Director, RRCS or Trunk Navigator connections. If any more try to connect, they are forbidden until one of the existing connections closes.

It would be most sensible, therefore, to configure RRCS to connect to a node that has no other connections.

Similarly, the main and reserve instances of RRCS should connect to different nodes.

5.2 RRCS View

The Artist router is a modular system that can have multiple frames connected together in a ring formation. The RRCS remote control software connects to a single CPU card in one Artist frame.

Therefore, the view of the Artist router that an instance of RRCS has is limited to the view of the router that the CPU card has. This is somewhat esoteric but is an important concept to understand. In regular operation you will not need to consider it but in certain fault conditions a failure to understand this can hamper your faultfinding efforts.

If RRCS cannot action a request the command will fail, no revertive will be sent to BNCS and an explanatory error message will be displayed in the debug log.

5.3 RRCS Hardware Communication Errors

Some notes on what happens to the driver when RRCS gains or loses communication with the Artist frame:

- If a connected instance of RRCS loses communication with the Artist it will alert the driver which correspondingly switches to Standby mode.
- If RRCS then re-discovers the Artist, the driver will attempt to switch to Working mode.
- Similarly, when RRCS closes it alerts the driver that it has lost communications with the Artist. The driver will switch to Standby mode. If RRCS crashes or closes without sending this alert, the keep-alive function will detect the failure and trigger a reconnection (as detailed in section 4.2).
- If RRCS does not have communication with the Artist when the driver attempts to connect for the first time, the connect attempt will fail.

5.4 BNCS Driver Shutdown

As the driver is an external to multiple BNCS host drivers, and all are essential for correct operation, some features are included to protect against the situation where an incomplete set of BNCS host drivers is running.

- During start up the driver connects to each BNCS host driver in turn. If one is not found, the start up procedure will pause.
- In normal operation, when the driver shuts down it disconnects from all BNCS host drivers but leaves them running. The host drivers are also set to RxOnly so another instance of the driver will take control.
- Both the driver and **all** BNCS host drivers will terminate if **any one** of its BNCS host drivers is closed. This behaviour is designed to reduce the likelihood of the driver running without all required BNCS host drivers, or a driver machine being left running with an incomplete set of BNCS host drivers.
- Holding SHIFT while closing the driver will also terminate all BNCS host drivers.

5.5 BNCS Redundancy

The driver can be run as a TxRx/RxOnly driver pair. In this case the mode (TxRx or RxOnly) is determined from the XY GRD. The driver will keep the other BNCS host drivers in step.

The Riedel gateway RRCS is correspondingly switched to its *Working* or *Standby* mode. In standby mode only a small subset of commands are supported so this driver only forwards requests to the Artist router when in TxRx mode.

For maximum redundancy, the two instances of RRCS should be configured to communicate with different CPU cards in different Artist nodes.

When this driver is switched from Standby to Working, therefore, it performs a router synchronisation to ensure the Artist router is configured in the way that the BNCS host driver tally tables indicate.

5.5.1 TxRx Retry Period

Although a full discussion of the BNCS driver redundancy mechanism is outside the scope of this document, it is necessary to briefly summarise it here.

The mechanism is managed by the instance of CSI running on the driver workstation which keeps track of all hosted drivers. A driver (InfoDriver or GRD) can be in 'TxRx' mode (working) or RxOnly (standby). The mechanism attempts to ensure that only one instance of the driver is in TxRx mode on the network at one time, although multiple RxOnly instances are permitted.

If a TxRx driver closes properly it alerts the network forcing an election between the RxOnly drivers. One will take over in TxRx quickly, usually well within a second.

If the TxRx driver does not close cleanly for whatever reason, the redundancy mechanism will detect its absence (and trigger the re-election) but this can take a long time. The more drivers running on a workstation, the longer this time will be. The minimum time will be the number of drivers running multiplied by the time period at which CSI checks the mode of its hosted drivers.

The workstation running this driver will have multiple BNCS drivers running, one for each interface function. It is therefore necessary to configure CSI with a shorter than default time period for its mode checking.

The setting that controls this mechanism is the `TxRxRetryPeriod` property under the `Misc` section of CSI.ini or bncs_system.ini in a V4.5 system.

This defaults to 60, which on a driver workstation with 9 devices running (as will be the case with this driver) could result in a delay of over 10 minutes before a failed driver is spotted.

It is recommended that this setting be reduced to 5s at the most, which will mean the delay before a failed driver is spotted is reduced to about 45 seconds.

5.5.2 Manual Intervention

The GUI has commands for setting the working mode of the driver, available under the BNCS menu.

5.6 Remote Interface Performance

It is useful to understand, at least a little, the inner workings of the Artist system.

5.6.1 Configuration or running status?

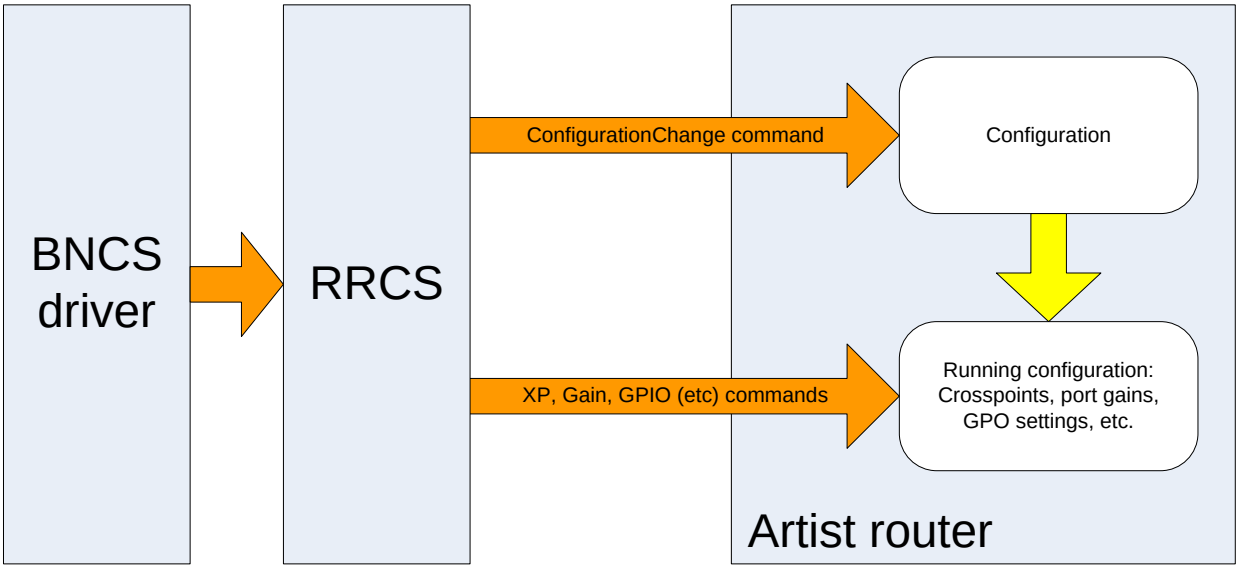
The Artist system maintains an online configuration: a set of elements containing details of all the hardware on in the system, what's expected on each port, panels, commands on keys, etc. The configuration can be inspected, modified and saved by the Director software.

The Artist also has a running status, which is the crosspoints, port gains, etc. Some of this is visible in Director (the crosspoint view, for example) but a lot is hidden (port gains, etc).

Some configuration elements alter this running status. For example, if a subscriber is in a conference, the *configuration* will hold a call-to-conference command for this subscriber. When this subscriber talks, the vox function on the port will (effectively) create crosspoints to other conference members. These are held in the *running status*.

RRCS API commands are split into two sections. ConfigurationChange commands alter the base configuration of the system and the changes made will be visible in Director. All other commands, such as SetXp commands, only modify the running status of the system. This distinction is important because of the time a ConfigurationChange takes to implement.

The command paths are illustrated in the following diagram:



5.6.2 Configuration Change Queue

The configuration is stored in the controllers on every node on the Artist fibre ring. When the control software (RRCS or Director) makes a change to this, it first must lock each node in the ring to ensure only a single software instance makes changes at once. If it didn't do this, there would be a danger of different nodes having different configurations, which would quickly lead to major confusion. The locking, uploading and unlocking of the nodes can take a long while, perhaps a second, to implement.

Obviously this is a long time to wait for a revertive, but, even worse, if 10 commands were sent, it would take the driver 10 seconds to process them all. To help mitigate this, the driver has a buffer of configuration changes:

When a driver request results in a need to alter the configuration, the driver creates a 'configuration change set' which is a list of what it has to do to implement the request. The driver then waits 150ms to see if any more configuration change requests will be created. This buffer means that commands sent simultaneously will all be sent to the Artist at the same time, effectively meaning the one-second delay is only experienced once. Up to 10 different commands will be buffered in this way.

A negative with this is that should any one configuration change not work for some reason, all changes that were sent to the Artist at the same time are rejected. In this case the BNCS panels will receive the previous revertive again to indicate that the new request has failed.

The driver will process other requests (with exceptions, see the *Sequential Commands Limitation* section) while it is waiting for RRCS to complete a configuration change request.

5.6.3 Sequential Commands Limitation

Many of the driver interface commands require the driver to precisely know what the current status of a configuration object is. For example, in conferences, the driver must keep track of which ports have *call-to-conference* commands configured in order to be able to delete the commands at a later stage.

When a request is received, the driver compares the current configuration of the object (e.g. a conference) with the new configuration that will result from the request it's just received. The driver then forms a list of changes to make to the Artist configuration in order

to satisfy the request. The list of changes is wrapped into a *configuration change set* and sent to the ConfigurationChange queue. When the configuration change is completed, remember that's perhaps a second later, the driver's internal tally tables are updated.

Crucially, this means that while a configuration change is in progress for a particular object (a conference, an IFB, a panel key) no other changes are allowed to that object.

Consider trying to remove two subscribers from a conference. You send a request to the driver to remove the first subscriber. The driver happily goes off to do this, but before it completes you send a request to remove the second subscriber. Because the driver is already processing a request for the conference, the second request will have to wait.

The second request is stored in a queue by the driver. This queue of pending requests is processed once the in-progress configuration change is completed. With every request taking a second (or more), when requests stack up the driver will take some time to process them all.

It would be much better, therefore, to send all the changes you wish to make in a single driver request. All configuration change commands can change multiple things at once (the conference command, for example, just accepts a new string of subscribers that could be entirely different from those that were present originally; there's no need to send a different command for each subscriber add/remove).

5.6.4 Key takeaway points:

- Commands that affect the configuration of the Artist take a long time to complete.
- The driver makes an attempt to buffer these commands together, so if multiple commands (altering different aspects of the configuration) are sent simultaneously, they will all be sent to the Artist at the same time.
- The driver only permits one change to each configuration object (conference, IFB, panel key) at a time. If a command is already in progress, any further commands will be queued until the first completes.
- The driver only permits one configuration change to happen at once, though each configuration change operation can contain a large number of individual changes (providing each change is to a separate object).

5.7 Ring Stability

It is vital that the core fibre ring and all Artist nodes on it are stable.

For all the configuration change operations, RRCS has to gain a write-lock of all connected nodes. Each node on the ring knows about what other nodes are present, so when RRCS requests a configuration change, it is told which nodes are online and therefore need to be updated. RRCS duly attempts to gain a write-lock on each of these nodes. When this is successful, the configuration change is made.

If a node is apparently online (according to the node to which RRCS is connected) but that node is misbehaving in some way, RRCS may fail to lock the node. This results in a long timeout and a failed configuration change. Generally, if RRCS cannot lock the required nodes, Director won't be able to either.

A node can fail in this mysterious misbehaving way due to a faulty controller card, or a faulty fibre. Whatever the reason, the ring must be stable or the configuration change mechanism will fail to work correctly.

5.8 Markers

A marker is Riedel's term for the pattern and colour of LEDs that appear above the key. The driver allows custom markers to be placed on keys through the monitoring interface.

Markers are defined in the Artist's configuration with the Director software. One set of markers is defined across the whole ring of Artist nodes. Some markers are pre-defined but you can customise them to suit your requirements. There are also some user markers that can be used for whatever unique and exciting things you can think of.

The driver cannot change the look of the markers. It can only specify which one is put on a key.

When the driver is expecting a marker you need to feed it a numeric value that identifies that marker in the configuration.

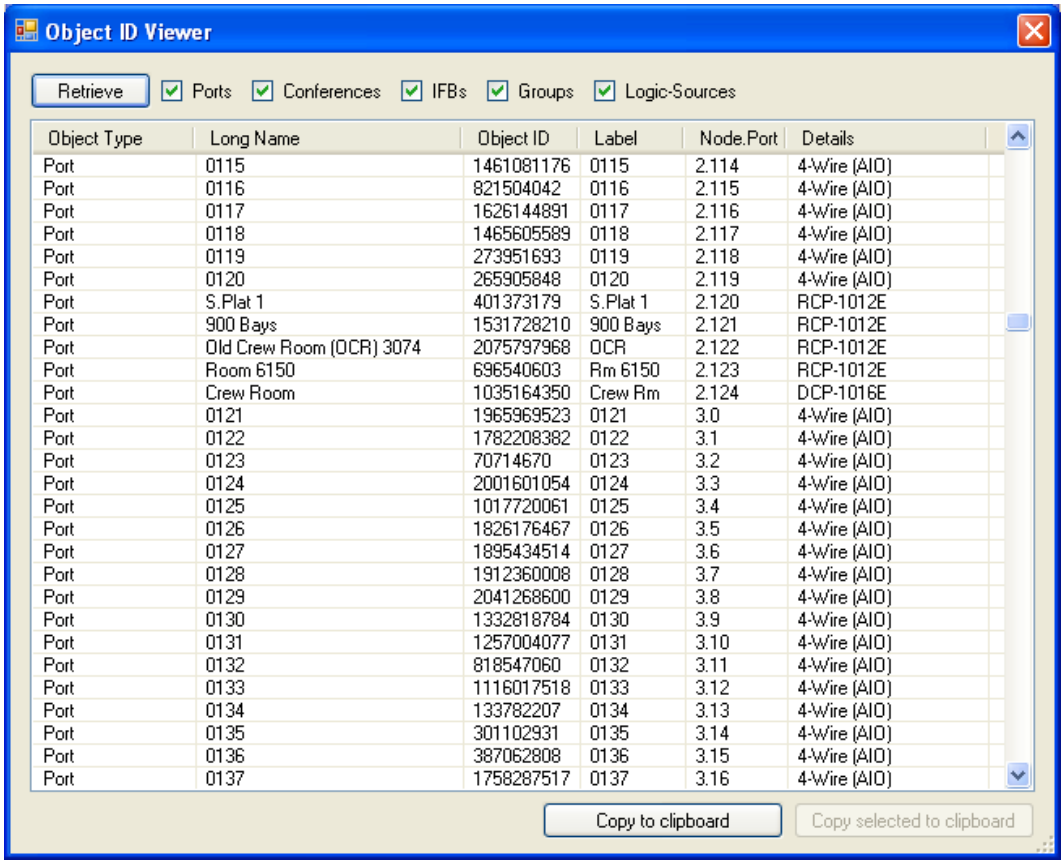
Note: the list of Markers in Director starts counting at 1, whereas the driver expects markers to start counting at 0. So you need to subtract 1 from the marker number as displayed in Director before sending to the driver.

Markers are all assigned a priority. It is possible there will be multiple markers on a key (for example if there are two commands on the key, each command causes a marker to be displayed). The marker with the highest priority is displayed. Therefore, to ensure your custom markers appear when you request them, these markers should be set to a very high priority.

6 Object ID Tool

Much has been mentioned about the need to specify Artist object IDs in the driver configuration files. Section 1.9 of this document introduces the concept and explains why they're needed.

The driver has a built-in function to retrieve these object IDs. Select Tools -> Show Object IDs.



Object Type	Long Name	Object ID	Label	Node.Port	Details
Port	0115	1461081176	0115	2.114	4-Wire (AIO)
Port	0116	821504042	0116	2.115	4-Wire (AIO)
Port	0117	1626144891	0117	2.116	4-Wire (AIO)
Port	0118	1465605589	0118	2.117	4-Wire (AIO)
Port	0119	273951693	0119	2.118	4-Wire (AIO)
Port	0120	265905848	0120	2.119	4-Wire (AIO)
Port	S.Plat 1	401373179	S.Plat 1	2.120	RCP-1012E
Port	900 Bays	1531728210	900 Bays	2.121	RCP-1012E
Port	Old Crew Room (OCR) 3074	2075797968	OCR	2.122	RCP-1012E
Port	Room 6150	696540603	Rm 6150	2.123	RCP-1012E
Port	Crew Room	1035164350	Crew Rm	2.124	DCP-1016E
Port	0121	1965969523	0121	3.0	4-Wire (AIO)
Port	0122	1782208382	0122	3.1	4-Wire (AIO)
Port	0123	70714670	0123	3.2	4-Wire (AIO)
Port	0124	2001601054	0124	3.3	4-Wire (AIO)
Port	0125	1017720061	0125	3.4	4-Wire (AIO)
Port	0126	1826176467	0126	3.5	4-Wire (AIO)
Port	0127	1895434514	0127	3.6	4-Wire (AIO)
Port	0128	1912360008	0128	3.7	4-Wire (AIO)
Port	0129	2041268600	0129	3.8	4-Wire (AIO)
Port	0130	1332818784	0130	3.9	4-Wire (AIO)
Port	0131	1257004077	0131	3.10	4-Wire (AIO)
Port	0132	818547060	0132	3.11	4-Wire (AIO)
Port	0133	1116017518	0133	3.12	4-Wire (AIO)
Port	0134	133782207	0134	3.13	4-Wire (AIO)
Port	0135	301102931	0135	3.14	4-Wire (AIO)
Port	0136	387062808	0136	3.15	4-Wire (AIO)
Port	0137	1758287517	0137	3.16	4-Wire (AIO)

6.1 Usage

Ensure the driver is connected to RRCS.

Select the object types you wish to view using the check boxes.

Click "Retrieve". This can take a few seconds.

The list will be populated with the objects, their names and their object IDs. In the case of subscriber ports, more information is provided such as the node.port address (net is assumed to be 1) and the details of what the port actually is.

6.2 Clipboard

The contents of the list can easily be copied to the clipboard. The data is placed on the clipboard as comma-separated variables so can simply be pasted into a spreadsheet for further processing.

Copy to clipboard copies the entire list. You can also copy just a selection (shift-click to select multiple objects).

BNCS

7 Version history

7.1 Software Versions (Released)

Version No	Date	Details	Name
1.0.0	May 2010	Initial release for testing. Supports Customers' Project 1 requirements.	Giles Moss
1.1.0	Dec 2010	Release for testing. Supports Customers' Project 2 requirements as much as it can, with the features available in Artist firmware v6.20.	Giles Moss
1.2.0	April 2011	Feature complete for Customers' Project 2.	Giles Moss
1.3.0	May 2011	Second channel audio commands added.	Giles Moss
1.4.0	June 2011	Less-destructive handling of commands on key and virtual positions to allow the driver to coexist less argumentatively with Director.	Giles Moss
1.5.0	July 2011	DebugTools now includes port names. Sync errors separated into errors and warnings.	Giles Moss
1.6.0	September 2011	Debugging functions added for conference port members. RxOnly driver correctly imports tally tables on startup. Bug with IFB revertive string order fixed.	Giles Moss
1.6.1	October 2011	Bug fix to main/reserve redundancy operation.	Giles Moss
1.7.0	October 2011	Remote control of logic sources and keys. Logic-commands on keys. Crosspoint Adjust feature. Various new port types supported.	Giles Moss
1.8.2	November 2011	Pool ports supported (needs RRCS 6.30.RR12 or greater) Port Probe function added. Mixers with single outputs don't need a group. IFB information added to PTI string. IFB and mixer information added to contributing member string. IFBs and mixers can target the same IFB construct. SPLITIFBSHORTCMDS setting added. LOGFUNCTIONERRORS setting added. Bug fix to stop sync process clearing but not re-asserting IFB and mixer group membership. Further bug fix to main/reserve redundancy operation. Workstation number of TxRx machine is announced via Extras InfoDriver.	Giles Moss
1.8.3	November 2011	Crosspoint Adjust now supports 2nd channel audio.	Giles Moss

1.8.4	November 2011	Number of mixers increased to 500 (vice 200). Mixers' "shortcut command" functionality added. Keep Alive check now also confirms RRCS' Artist connection.	Giles Moss
1.8.5	January 2012	Bugfix: Output ports' contributing source "PTI"s were not updated after an IFB change. Now they are.	Giles Moss
1.8.7	February 2012	Bugfix: Presence of a stuck "&RESEND" in a conference slot now doesn't kill startup processing. Bugfix: Ports defined twice in INI file now doesn't kill startup processing.	Giles Moss
1.8.8	March 2012	Bugfix: Using shortcut commands to set key labels now doesn't wipe key command.	Giles Moss
1.8.9	May 2012	Improvements to monitoring functions.	Giles Moss
1.8.10	June 2012	Bugfix: Volume regex in mixer shortcut command. Main application lozenge now reflects TxRx status	Giles Moss
1.8.12	August 2013	New INI file setting GORXONRRCSLOSTRTR. DigitalGiles code library update. Driver now uses .NET 4.0	Giles Moss
1.8.13	January 2014	Environment variables can be used in the INI file paths to additional configuration files. RRCS version checking improved.	Giles Moss
1.8.14	March 2014	Mixer shortcut processing handles a volume of "mute" and the ability to not specify the port number if you only want to change the volume. Mixer configuration now requires OBJECTID-**MIXMINUS**GROUP instead of -**INPUT**GROUP	Giles Moss
1.8.15	October 2014	Mixer processing now drops a MUTEd port from the group, instead of setting its volume to MUTE. Mixers no longer allow a port to exist twice in the same group.	Giles Moss
1.9.0	May 2015	Key function 'CALLTOPORT' now supports L and V flags for reverse-call functions from the called port back to the panel. IFB interface allows individual ports to be added or removed. PortClear function added to PTI interface. Monitor configuration accepts an "IGNORED" type to allow dormant configurations. Conferences & mixers can be defined with a name (IFBs and monitors already could). Startup error log file written (can be easily launched in Notepad from GUI menu). *Panel Key Examiner* function added to Debug Tools.	Giles Moss
1.9.1	May 2015	PortClear enhancements.	Giles Moss
1.9.2	May 2015	&REMOVE=EVERYTHING shortcut commands added to conferences and IFBs.	Giles Moss
1.9.7	July 2015	Added &REMOVE(IN|OUT) shortcut commands to mixers.	Giles Moss

1.9.8	July 2016	Debugging improvements.	Giles Moss
1.10.1	August 2016	*Running Status* display added. RRCS version requirement dynamically changes depending on what startup options are specified (i.e. trunking requires 6.90.RR1).	Giles Moss
1.11.0	May 2017	Supports manipulation of VoIP properties on client cards and ports.	Giles Moss
1.12.0	April 2019	Actually enforce the 8-character limit of Port and Conference alias strings.	Giles Moss
1.13.0	April 2019	Adds Port Activity Monitor function.	Giles Moss
1.14.0	April 2019	Adds TRUNKCALL key function.	Giles Moss
14.1	April 2019	Adopted a new numbering system. "One number getting bigger". There's no point having the Major number as "1" all the time because when is it realistically going to change? Now we have Major.Minor where Major denotes new features and Minor denotes fixes. In the assembly the third and fourth parts of the version relate to the build date and time. Includes bug fix to null labels (seen in MRM) and a startup race condition in the driver internals.	Giles Moss
14.2	April 2019	Permit comments in INI files (use ; at the start of a line)	Giles Moss

7.2 Document version

Version No	Date	Details	Name
0.1	April 2010	Initial Draft	Giles Moss
0.2	April 2010	Added some more necessary settings and tweaked text in response to user comments.	Giles Moss
1.0	May 2010	Final version to coincide with release 1.0 of the driver.	Giles Moss
1.1	May 2010	Reviewed	Andrew Prince
1.2	May 2010	Information added on GUI indications and functions.	Giles Moss
1.3	May 2010	Added initial information about Phase 2 functionality. Work in progress, there will be changes and refinements as our understanding of how the Artist behaves improves.	Giles Moss
1.4	May 2010	Reviewed	Andrew Prince
1.5	May 2010	Updated following discussions with Customer and Riedel. Reviewed use of LongName and Object ID properties where appropriate. Better explanation of some of the interfaces.	Giles Moss

1.6	Dec 2010	Updated to document how the driver actually works with release 1.1.0 This differs in a few ways from what was originally envisaged, resulting from API, performance and functional issues, changes and requests encountered during development.	Giles Moss
1.6.1	Dec 2010	Added information on a couple of conference related features. Added section on the Object ID tool.	Giles Moss
1.6.2	Jan 2011	Conference slot layout and configuration file format changes.	Giles Moss
1.6.3	Jan 2011	Correction to KEY ASSIGN InfoDriver slot string format	Giles Moss
1.7	Mar 2011	Many small updates including: • Call-to-group function. • Logic sources. • Sync alert. • Change to the ObjectID requirement in the MONITORCONFIG.INI file. • Added a note about markers.	Giles Moss
1.8	Mar 2011	Many small updates including: • ObjectIDs no longer required in RiedelPorts or RiedelMonitor INI files. • Conference re-send and shortcut processing commands. • Second audio channel support for panel key functions. • Database changes tied to port aliases. • ObjectID tool is now a built-in function in the driver. • Added a note about ring stability.	Giles Moss
1.9	Apr 2011	Addition of Key Property manipulation.	Giles Moss
1.10	Apr 2011	Updated to reflect change in how ports are treated.	Giles Moss
1.11	Apr 2011	Added section on mixer configuration and use. Driver release 1.2.0	Giles Moss
1.12	May 2011	Added second channel support to 4-Wire destination monitoring. Driver release 1.3.0	Giles Moss
1.13	May 2011	Tweak to call-to-port panel function behaviour. Driver release 1.3.4.	Giles Moss
1.14	June 2011	Driver now plays nicer with Director-created commands in various places. Bug fix to reported port gain value. Driver release 1.4.0	Giles Moss
1.15	July 2011	Supports driver release 1.5.0	Giles Moss
1.16	October 2011	Corrected some errors in the documentation. Supports driver release 1.6.0	Giles Moss

1.17	October 2011	Supports driver release 1.7.0	Giles Moss
1.18	November 2011	Corrected detail on GPI definitions. Supports driver release 1.8.0	Giles Moss
1.19	November 2011	Further bugfixes and modifications for driver release 1.8.2.	Giles Moss
1.20	November 2011	Modifications for driver release 1.8.3	Giles Moss
1.21	November 2011	Modifications for driver release 1.8.4	Giles Moss
1.22	January 2014	Modifications for driver release 1.8.12	Giles Moss
1.23	January 2014	Modifications for driver release 1.8.13	Giles Moss
1.24	March 2014	Modifications for driver release 1.8.14	Giles Moss
1.25	October 2014	Modifications for driver release 1.8.15	Giles Moss
1.26	May 2015	Modifications for driver release 1.9.1	Giles Moss
1.27	May 2015	Modifications for driver release 1.9.2	Giles Moss
1.28	July 2015	Modifications for driver release 1.9.7	Giles Moss
1.29	July 2016	Modifications for driver release 1.9.8	Giles Moss
1.30	August 2016	Modifications for driver release 1.10.1	Giles Moss
1.31	May 2017	Modifications for driver release 1.11.0 Added key layout for new smart panels	Giles Moss
1.32	April 2019	Modifications for driver release 1.12.0	Giles Moss
1.33	April 2019	Modifications for driver release 1.13.0	Giles Moss
1.34	April 2019	Modifications for driver release 1.14.0	Giles Moss
1.35	April 2019	New driver numbering adopted with release 14.1.	Giles Moss
1.36	April 2019	Modifications for driver release 14.2	Giles Moss