

# **BNCS**

## **Virtual Router Driver**

**Author: Simon Dowson**

---

### **Introduction**

This document will be continuously updated throughout the development of the VRD module. The version release number is shown above and should correspond to the version number of VRD.EXE that you are using. This is the second revision to the original proposed specification and the original documents should be referred to whilst reading this one.

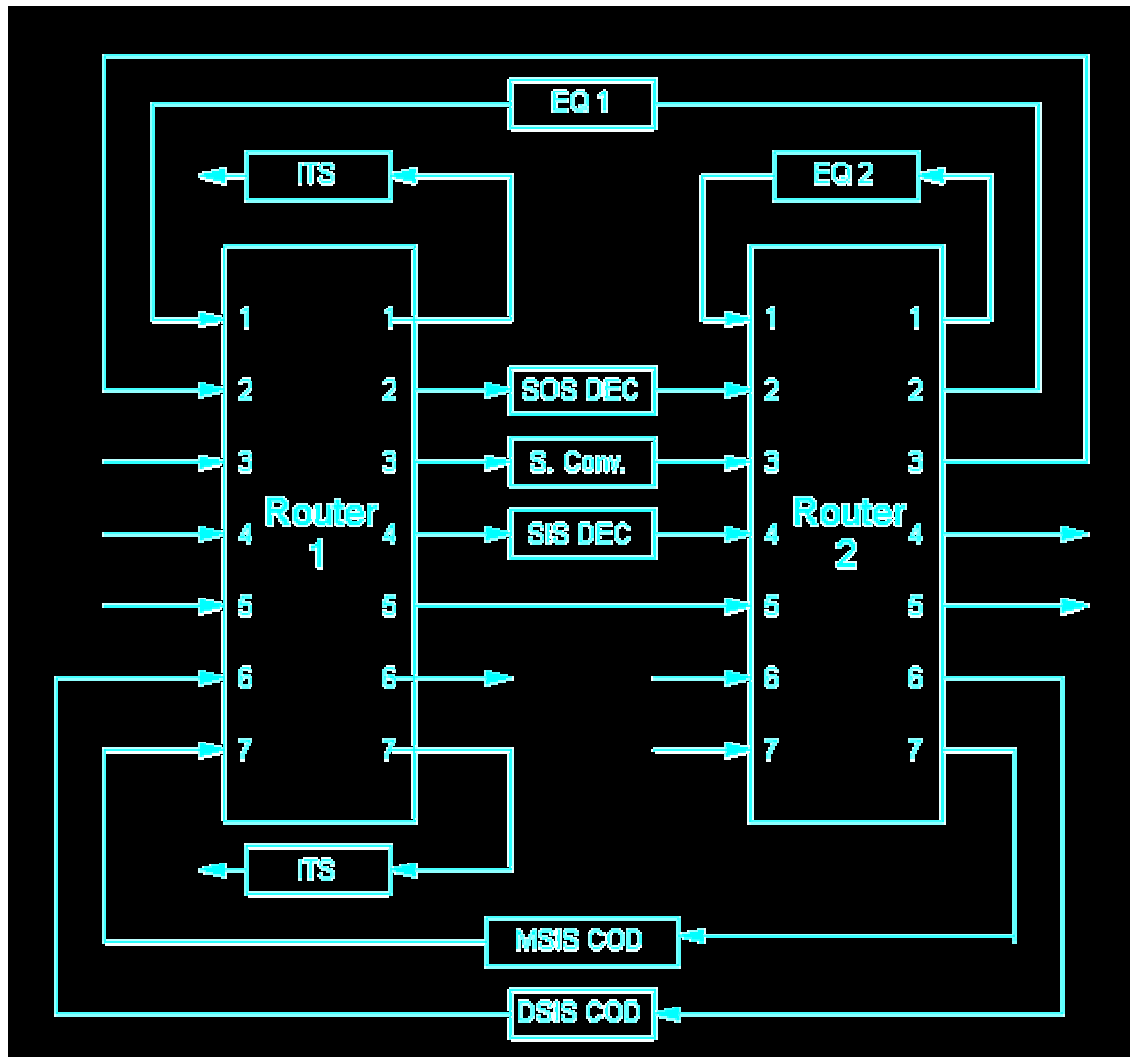
Development of this module has been incremental so that it could be enhanced to meet users evolving requirements. This version (1.03) is the latest in that evolution and the design will likely be frozen at this point. Further bug-fix versions will be released if necessary.

### **Overview**

The Virtual Router Driver (VRD) is a BNCS module which allows up to 8 cascaded routers to be treated as a single router. All interconnecting tielines and resources are managed by the VRD to enable users to select sources without regard to the routes that need be set up in order to get the source signal to its target destination. Routes or part routes can be shared or private. Sources can be selected from the first router in chain or any intermediate router.

A VRD appears to the system as an ordinary router driver. It has source and destination names that mirror the source and target routers. It responds with revertives messages, not only about the virtual source data, but route and resource information as well.

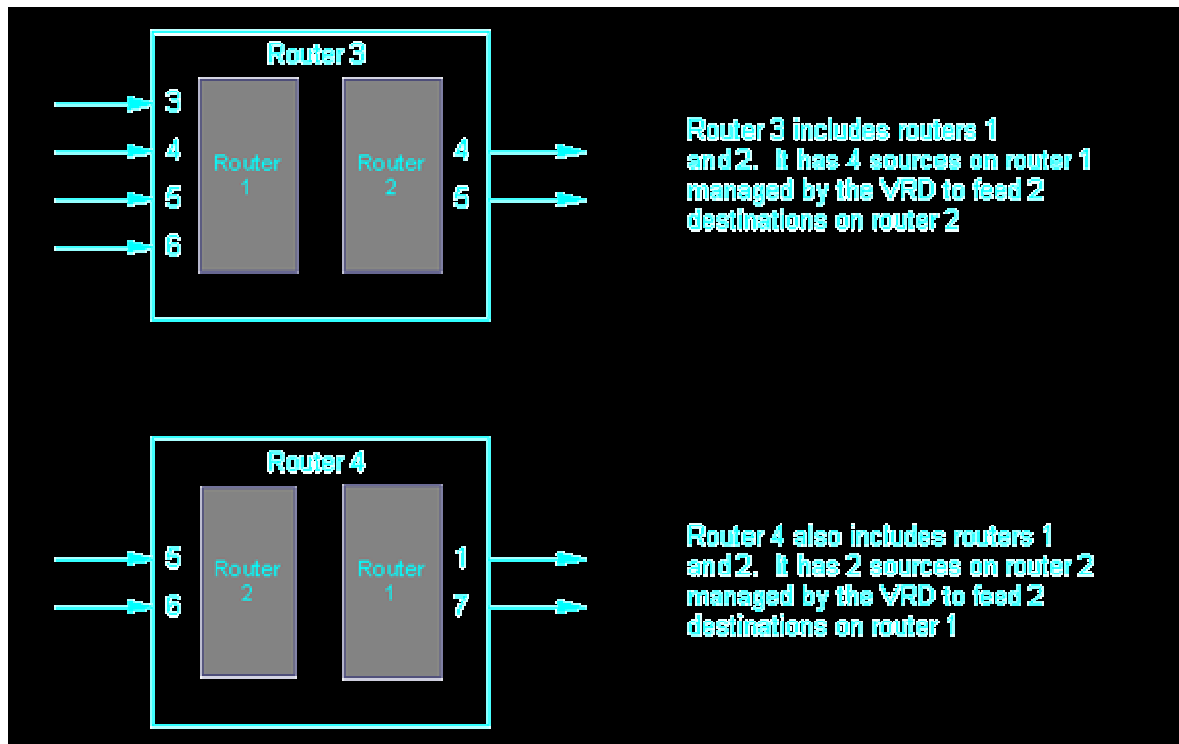
The diagram below shows just two routers forming two VRD's, one of which will be used as an example throughout the configuration sections of this document.



### Example VRD

A VRD allows two or more routers to appear to BNCS as a single router. This does not mean that all the destinations or resources associated with those routers have to be part of the VRD. Only the destinations specified in the VRD configuration file are used by the VRD. Other destinations can be used as normal or form part of another VRD.

In this example the routers 1 and 2 are configured to appear as two VRDs, namely routers 3 and 4.



This is not a 'real world' example , but it does show the potential complexity. There may be the temptation to build a complex virtual system to make up for deficiencies in hardware. This should be avoided where possible. The best approach is to keep the implementation of VRD's as simple as possible.

## Terminology

A **virtual crosspoint** consists of all the real crosspoints used in a route.

The **foremost router** is the first router in the VRD chain.

The **source router** is the first router used in a virtual crosspoint

The **target router** is the last router in the VRD chain.

The **target destination** is the final destination on the target router in the VRD chain. It is this destination index number that is used as a **user** number to register the usage of all the tielines involved in the through route.

A **virtual crosspoint** exists if there is a through route from the source router to the target router. The virtual crosspoint is only valid if the routes involved are managed by the VRD. If any crosspoint is changed within the virtual crosspoint, other than by the VRD, then the virtual crosspoint is deemed invalid. Ownership and usage counts for all crosspoints that made up the virtual crosspoint are deleted.

## Specifications

The current maximum values apply as of this release, but may be subject to change:

-

Maximum number of routers within a VRD = 8

Maximum number of tieline destinations per router = 32

Maximum number of resource types = 31

## **Resource Types**

This version of the VRD can manage 31 different resource types. The following are defined for the purposes of this document. The names are not important and you can allocate any numbering scheme you wish. A direct tieline, however, must have a 0 for a resource type.

Direct tieline, no equipment. 0

Mono SIS Coder 1

Mono SIS Decoder 2

Stereo SIS Coder 3

Stereo SIS Decoder 4

SOS Decoder 5

Standards Converter 6

## **Configuration**

The configuration is achieved in a few simple steps. VRD.EXE will attempt to initialise the INI file with default values where possible. You can setup a VRD using any of the three methods below.

1. Create an INI file from scratch using a text editor.
2. Clone an existing VRD INI file and modify it.
3. Allow VRD.EXE to create a new one for you.

The latter is preferable because it is more likely to show up errors in tieline allocations and is the one used in the example that follows. The process is split up into three simple stages and involves starting the VRD three times and then editing the INI file to configure the options to suit your requirements. Each time you start the VRD it will read the INI file and then, based on the information you have given it, add new sections to the INI file for further editing. The three stages are :-

1. Setting up the real router ID's that comprise the virtual router and mapping the virtual sources onto real ones.
2. Allocating the destinations used as tielines on each router.
3. Specify the interconnections and resources in each tieline.

During the initialisation process VRD will appear full screen and a scrolling window of progress information will be presented until initialisation is complete or fails. If

initialisation completes successfully then VRD will minimize itself after a short delay, otherwise it will remain maximised so that the reason for failure can be examined.

## **Stage 1**

Decide on the router driver number you wish to use for the VRD and then start VRD.EXE using this number as a command line parameter. The example shown below is for VRD 3

VRD.EXE 3

VRD.EXE will look for a file called GRD\_003.INI. If the file is not found then it will be created with the first set of default values. Close down VRD.EXE since it has now created the skeleton INI file and we need to manually configure it before we can run it again.

At the moment the new INI file will have just one section called [VRD]. This section will contain the following entries as described below :-

### **[VRD]**

Default	Configuration	Explanation
Workstation=0	Workstation=27	The workstation ID for the computer hosting the VRD.EXE application. Each computer in the system must have a different workstation number. It is important to set this parameter as configuration will not proceed without it.
DebugMode=0		If this is set to '1' VRD will produce more verbose output both during initialisation and normal operation. It can be turned on and off using the 'Diagnostics' option on the VRD menu.
Routers=0	Routers=1,2	This is a comma delimited list of the routers that make up the virtual router. The first one is the source router and the last one is the target router. The order of any intermediate routes in the list is important.
VirtualSources=128	VirtualSources=192	This is the total number of virtual sources to be handled by this VRD. The

		maximum number of virtual sources is 999. A [VirtualSource] section is created as a result of this entry which will be described later.
NetworkBuffers=8		Network buffers listening for incoming commands. The number of buffers to allocate depends upon the amount of use to which the VRD is put.
InfoDriver=0	InfoDriver=103	The VRD needs to send you information regarding the availability of routes and resources. Such information will appear to come from an InfoDriver with the ID specified for this parameter. The slot allocation will be explained in a later section

## **Stage 2**

Configure the INI file as shown in the 'configuration' column above restart VRD.EXE to create sections for each of the four routers.

### **[Router 001]**

Default	Configuration	Explanation
ParkSource=0		The park source is normally a deselected state where no source feeds a destination. On some routers, where a deselected state is not recognised, you will need to allocate a real source to be used as the park source.
Destinations=0	Destinations=2,3,4	Each of these numbers is a destination on router 1 that feeds a tieline to another router or back to a source on router 1. The VRD will use this information to create entries for each destination the next time you run it. The target router should only have entries that refer to destinations that feed re-entrant sources.

Re-entrancy=3		You may have several reentrant tielines. i.e. those whose source and destination are on the same router. The VRD can make use of these tielines to pick different signal modifiers on each pass through the router. Setting this parameter limits the number of passes the VRD will allow through a single router. This will prevent a single route potentially using every single tieline from a router.
---------------	--	---

Now do the same for [Router\_002]

### **[Router\_002]**

Default	Configuration	Explanation
ParkSource=0		As above
Destinations=0	Destinations=4,5	As above
Re-entrancy=3		As above

For a VRD with intermediate routers you would configure the ***Destinations*** parameter in all the router sections to reflect the destinations in use for tielines.

### **Stage 3**

Restart the VRD. In each [Router\_xxx] section, VRD will now create Destination\_XXX entries for each destination specified in the respective ***Destinations*** parameter list.

### **[Router\_001]**

Default Configuration	Configuration	Explanation
Destinations=2,3,4		See previous section
Re-entrancy=3		See previous section
Destination_002=NONE	Destination_002=2,2,6	There are three comma delimited numbers. On this router, destination 2 on goes to source 2 on router 2

		and contains resource type 6. Several resources can exist in a tieline by adding further comma delimited numbers.
Destination_003=NONE	Destination_0006=3,2,31	
Destination_004=NONE	Destination_0007=4,2,14	

### **[Router\_002]**

Default Configuration	Configuration	Explanation
Destinations=4,5		See previous section
Re-entrancy=3		See previous section
Destination_004=NONE	Destination_0004=0	See previous section
Destination_005=NONE	Destination_0005=0	See previous section

Configure the [Router\_xxx] sections as shown above and restart VRD.EXE. If the Destination\_xxx entries are valid the VRD will continue to initialise otherwise it will report an error in the INI file. If errors exists correct them and restart. On the target router only destinations that are reentrant should appear in the configuration table.

The INI file should now contain a [VirtualSources] section with a default entry for each source. You can optionally configure this section to reflect how the virtual sources map onto real sources on real routers.

### **[VirtualSources]**

Default	Configuration	Explanation
Source_001=A,B,C,D,E etc	Source_001=1,5,R,S,5	There is an entry for every virtual source. The parameters are of the form <b>A,B,C,D,E... etc</b> where <b>A</b> is the real source index and <b>B</b> is the router on which it exists. The



		default is for <b>A</b> to be the same as the virtual source index and <b>B</b> to be the ID of the foremost router in the chain. <b>C,D,E...</b> etc is the default action to take if a mask is not supplied in the ApplCore 'Route Crosspoint' command.
Source_002=A,B,C,D,E etc	Source_002=2,5,R,P,0	As above
Source_003=A,B,C,D,E etc	Source_003=3,7,R,P,0	As above

## **Database Management**

The VRD depends upon the real router drivers that it is configured to manage. These real router drivers have their own databases. The virtual router sources can be on any router and the destinations are on the target router. The real router driver databases are treated as the 'master copies'. One of the last initialisation routines the VRD performs is to copy the source names used as virtual sources from all the source router INI files and the destination names from the target router INI file and place these in the **[Database]** section of its own INI file. It will also create a **[Router]** section containing the maximum sources and destinations for the virtual router based upon information found in the source and target router INI files. This information provides CSI with the ability to treat the virtual router just like a real one.

After the VRD has been created for the first time you will need to restart CSI so that it can load in the new database for the virtual router.

Due to the fact that the real router drivers hold the master copies of the database it is these drivers that must validate any requests to change their database. If database change commands are sent to the VRD router Id then it redirects the change request to either the source router driver or target router driver as appropriate. These drivers then validate the change request and pass the database change information back to CSI. CSI passes the information onto the VRD, since it is a client of CSI. The VRD will see that the change effects one of its virtual source or target destinations and know that this has already been validated by either of the real router drivers. The VRD will then send change information to CSI to say that the *virtual router database* has changed. As a client of CSI the VRD will get its own message back, but will ignore it.

## **Initialisation**

The VRD will poll all the routers that make up the VRD and store the revertives in its tally tables. If there are no errors then the VRD is ready to accept commands.

During normal operation the VRD will create a section called [Routes] and a section called [Resources]. When a valid virtual route is made the complete path through the routers and the resources allocated are stored for future reference.

## **Commands**

Commands are sent to the VRD in the same way as for an ordinary GRD. The RP command is used to Poll for a tally update and RC is used to make crosspoints. However, because the VRD is not a single physical device there are a few extra considerations.

## **Polling**

You can poll for all the destinations on the target router. The VRD will return the **virtual source** information for all destinations that are within the VRD's control. Destinations not covered by the VRD will have a deselected status. i.e. '---' or -1.

## **Making Crosspoints**

The RC command is used with the VRD number and the virtual source and target destination as you might expect. The RC command has a fourth parameter normally used for controlling breakaways of associated routing with the Packager system. The VRD also makes use of the mask parameter, but uses it for a different purpose.

The mask parameter is a comma delimited string and holds three sets of information. The first is a sub-command either 'INQUIRE' or 'ROUTE'. These can be shortened to 'I' and 'R' respectively. The sub-command tells the VRD whether you want to inquire if a route is possible or actually go ahead and make the route.

The second sub-command indicates how the route is to be used. There are 2 usage attributes indicating either a 'SHARED' or 'PRIVATE' route. Again these can be shortened to just the first letter. Shared routes can be used by more than one target destination. When a route is shared with a new target destination then that destination number is used as a 'user' number and is added to the user list for the route. A destination can only be cleared down or changed by the last user. Private routes can only be used by a single target destination.

Following the second sub-command is a list of resources that need to be picked up within the route. The VRD will attempt to find a valid route using the fewest number of tielines. If no mask is supplied then the VRD will default to making a shared route with no resources.

Where possible tielines that are carrying the same virtual source will be shared. Tielines cannot be shared if a target destination has them marked as private. A sample routing command would look something like this :-

```
RC 200 15 44 'I,P,5'
```

This is an inquiry to VRD 200 about a private route between virtual source 15 and target destination 44 that should include a stereo SIS decoder.

## **Revertive Messages**

Revertive messages from the VRD will be the same as you would expect from a normal GRD. However, when a route connect request has fails you will also receive a revertive of source -1 for the target destination. Only the workstation originating the request will get the message. All other workstations will continue to display the original virtual souce.

In addition the VRD needs to send you information about wether a route is possible and if not why it failed. To receive this information you must register within your panel to receive InfoDriver revertives from the device ID specified as the 'InfoDriver' parameter in the VRD\_xxx.INI file. The VRD sends messages that conform to the InfoDriver format and messages will appear to come from InfoDriver slots. The InfoDriver Id is configurable in the [VRD] section of the INI file. The slot allocation follows the 'External InfoDriver Client' specification and only two slots are used :-

Slot 1	Slot 2	Slot 3
Result code between <b>0 &amp; 5</b>	Route in the form :- <b>'Router Source Dest, '</b>  in comma delimited blocks.	Resources in the form :- <b>'1,5,11,17'</b>
<b>0</b> = Success	Contains the entire route	Contains all the resources in the route
<b>1</b> = No routes available	Contains the route up where the search failed	Contains the resources still to be found.
<b>2</b> = No resources available	Contains the entire route	Contains resources not found. The target router may have been reached without all the required signal modifying resouces being found. This slot contains a comma delimited list of the resources that could not be found.
<b>3</b> = Virtual route violation	Contains the destination that has been violated	Contains ' <b>0</b> '
<b>4</b> = Already routed	Contains the entire route	Contains all the resources in the route
<b>5</b> = Cleared down	The destination that has been cleared down	Contains NULL

If a route is found then the result code will be zero, slot 2 will return the complete route and slot 3 will contain all the resources in the route.

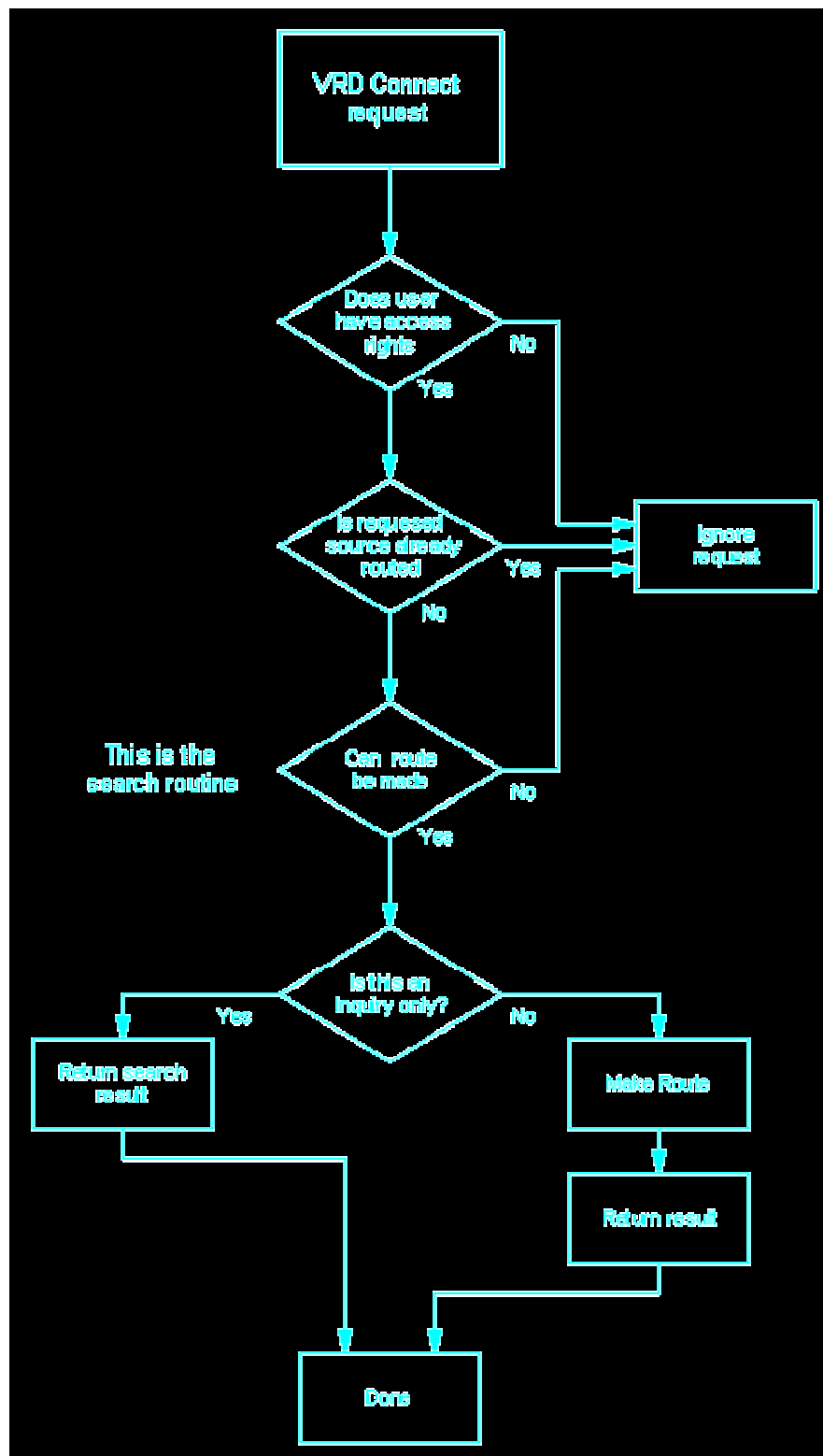
If the route is not found the result code will indicate whether the failure was due to lack of availability of a tie line or one or more resources. Slot 2 will reflect how far the search got and slot 3 will reflect the resources that could **not** be found.

If the result code is a route violation (3) then slot 2 reflects the destination that has been changed outside the VRD.

Messages are sent whenever an 'Inquiry' or 'Route' request is made. The InfoDriver cannot be polled since result codes are only transitory.

## **Route Stability**

Within BNCS the VRD appears as just another router. The fact that a VRD is configured to manage a selection of destinations on several routers does not prevent access to the individual routers that make up the virtual router. This means that you can override the action of a VRD. In such a circumstance the VRD will not reassert any crosspoints, but **will** declare the route to be invalid as a virtual route and zero its tables for the target destination. This is the **only** way of breaking down a private route not owned by you. The following diagram shows the overall mechanism that the VRD uses to handle routing requests.



### VRD operation

The VRD has three distinct operations to perform :-

- Route searching

- Clearing down
- Integrity violation response

## **Route Searching**

The flow diagram below shows the outline procedure for making a virtual route. This differs in the procedure that a normal router driver takes in two ways :-

1. The VRD must search for a valid route from the source router to the target router, picking up all the required signal modifying resources, with the minimum of router re-entrancy.
2. The VRD must enable a route to be found and the results made available without committing the user to actually setting up the route by making crosspoints.

## **Route Search Routine**

The route search routine has been made as simple as possible. It attempts to find the shortest route through two or more routers, picking up any required signal modifying resources on the way. It applies the following rules:-

- The path through the routers is left to right through the 'Routers' list in the INI file. Backward searches are not made.
- Re-entrancy is permitted on a router up to the maximum set by the configuration in the INI file

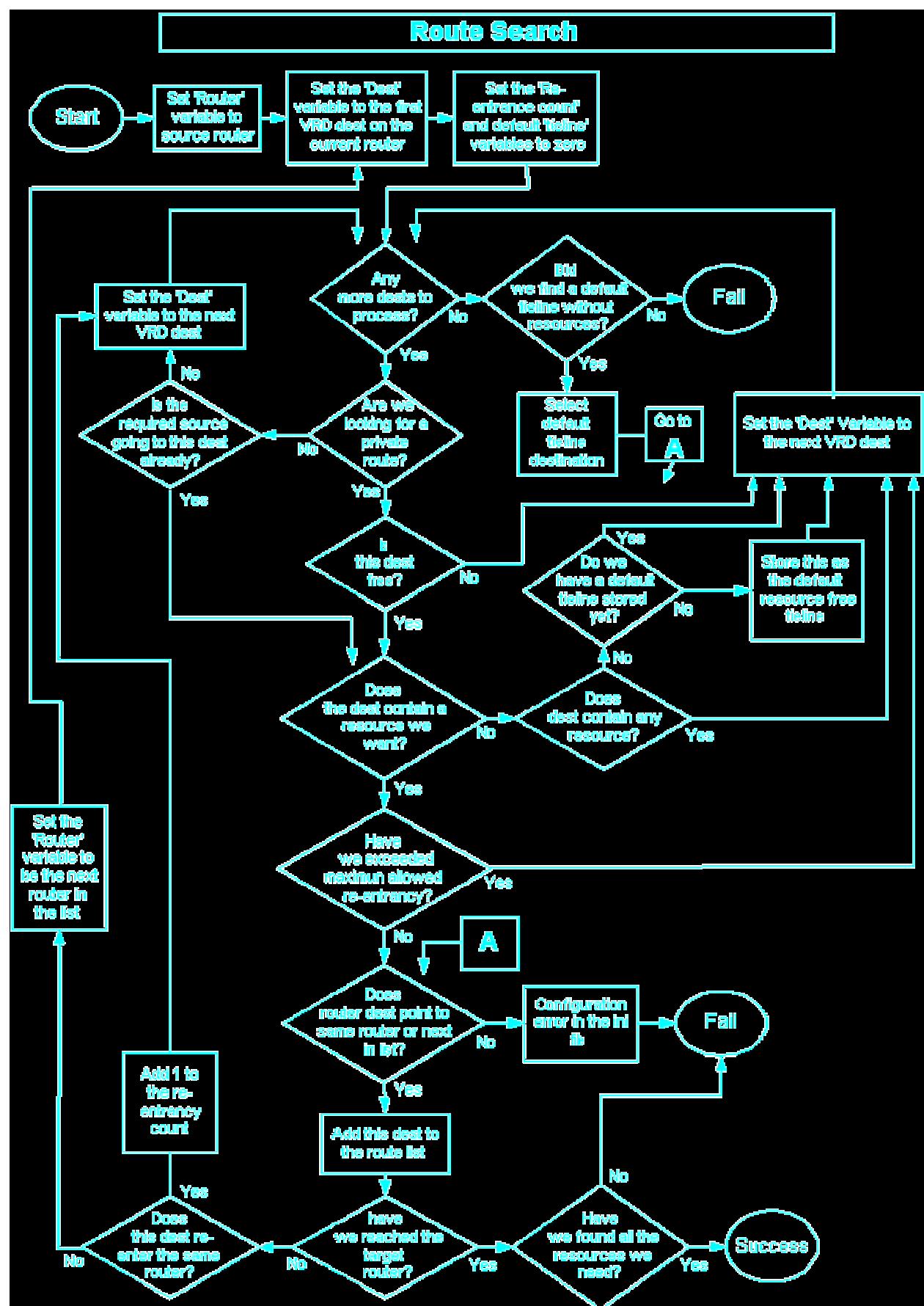
No crosspoints are made or cleared during the search. The result, if the search is successful, is a 'route list' which contains all the routing information necessary to set the appropriate crosspoints.

The flow chart below shows the decision making process that the VRD employs. The search begins on the source router. Each destination on the router that lies within the VRD is checked. If a private route has been requested only parked destinations will be considered, otherwise a shareable destination that is currently carrying the same source is used. The destination is checked to make sure that it does not contain a resource that is not required and which might introduce signal conditioning

inappropriate to the overall route being requested. A check is then made to see if the destination contains a resource that is required. If the destination is not suitable then the next destination within the VRD is checked until the last one is reached. If no tielines have been found that contain a required resource then the first tieline not containing any resource is used. This will take the search onto the next router where hopefully a suitable resource can be found.

The search ends successfully when the target router is reached and will fail if the lack of an available resource prevents the target router being reached. If the 'R' or 'ROUTE' option is used as part of the Router Connect message the VRD will immediately examine the route list and send messages to all the router drivers involved. If the 'I' or 'INQUIRE' option is used then the route list is returned to the user as an InfoDriver message.

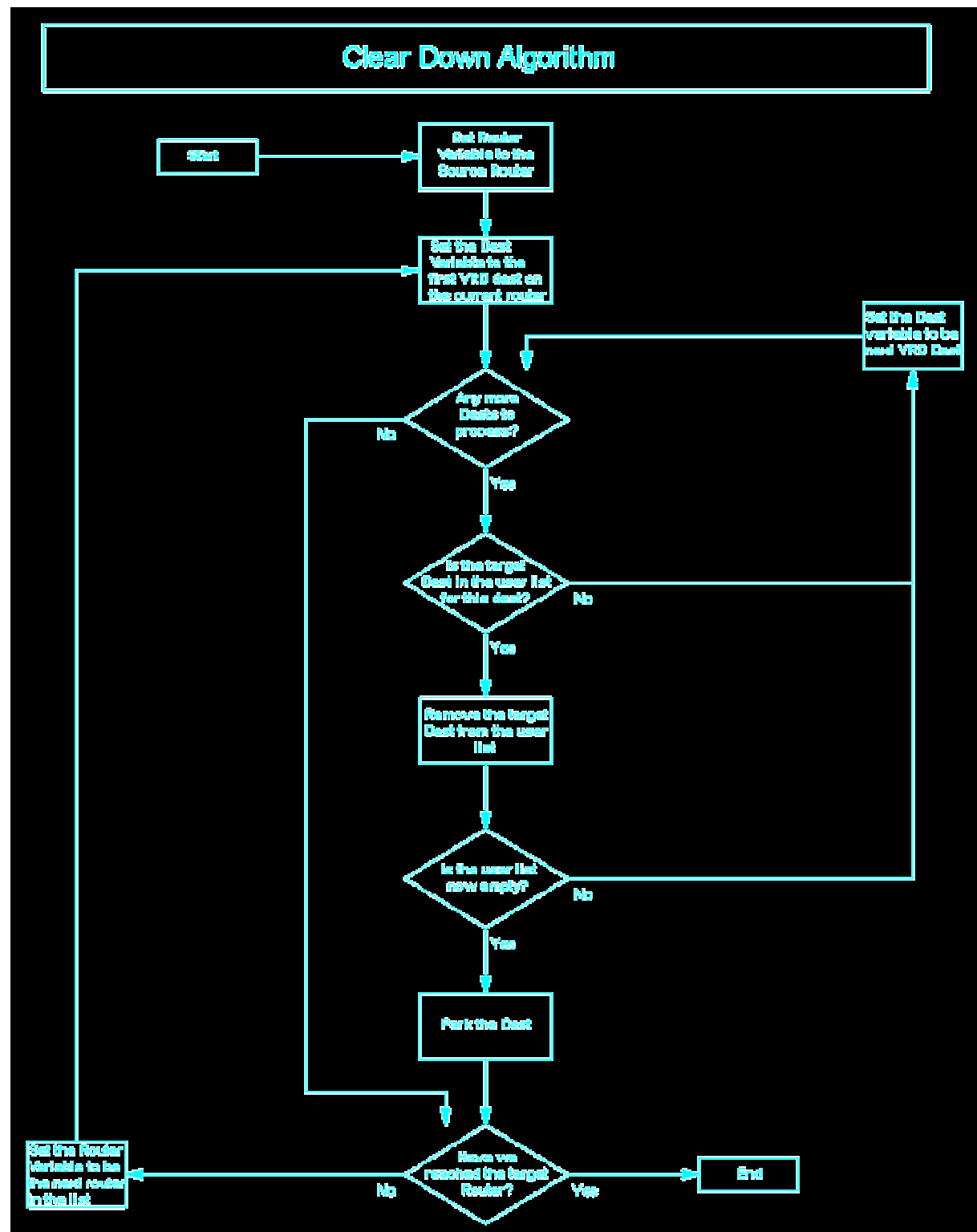
If a route is requested and a previously valid route already exists to the target destination then a 'cleardown' is performed first on that target destination. You will receive 'cleardown' information followed immediately by the new route information.





## Clear Down

The clear down routine scans each destination on each router in the VRD. If the target destination being cleared is found in the user list for a destination then the target destination is removed from the list. If the user list is then found to be empty then the destination is parked.



## **Integrity Violation Response**

If a VRD is configured to manage destinations on the routers that make up the virtual router it should, during normal operation, be allowed to setup and clear down routes without any external intervention. Since BNCS is an open system that leaves the modes of operation entirely to the panel designer and system administrator there is nothing to prevent the destinations within the virtual router being changed without reference to the VRD. There are valid reasons why this may be done, such as substituting a tieline resource due to failure without clearing down a route first. However, when such a change occurs the VRD is no longer able to manage the route.

In such circumstances the VRD will make the following response. If any destination within the VRD is changed other than by the action of the VRD itself then the VRD will put a copy of the user list of target destinations for the affected destination into an '**invalidation list**'. All the target destinations in the invalidation list are affected by the change that has just occurred since they were all sharing the destination. The VRD will then scan each destination on each router within the virtual router. If a target destination in the invalidation list appears in the user list of any destination in the VRD then it is removed from that user list. All references to the virtual route passing through such destinations is cleared. This is done to prevent the VRD from acting upon a clear down command and the likelihood of it disturbing manually setup routes.

No crosspoints are affected. When subsequently searching for routes the VRD will manage a previously invalidated destination based on its new status.

## **Route Inquiry**

To inquire about the route used by a target destination use a Route Crosspoint command with the source set to '0' and 'INQUIRE' as the last parameter :-

### ***RC 3 0 7 'INQUIRE'***

You will receive an InfoDriver reply with slot 1 set to '4' meaning already routed. Slot 2 will contain the route and slot 3 will contain the resources used.

## **Version History**

Version	Details
1.00	<b>First release :-</b>  <ol style="list-style-type: none"><li>1. It had the ability to make and clear virtual routes.</li><li>2. It could detect route violations and declare the virtual route invalid.</li></ol>
1.01	<b>Modifications :-</b>  <ol style="list-style-type: none"><li>1. Now checks to see if required virtual source is already routed to the target destination.</li></ol>

	<ul style="list-style-type: none"> <li>2. Routes and resource usage now stored in the GRD_XXX.INI file.</li> <li>3. Private/Shared mode tracking improved.</li> </ul> <p><b>Additions :-</b></p> <ul style="list-style-type: none"> <li>1. A route command with the source set to 0 and using the INQUIRY option returns the current virtual route and resources used.</li> <li>2. Database management.</li> </ul> <p><b>Bug fixes :-</b></p> <ul style="list-style-type: none"> <li>1. Corrections to information and router revertives.</li> </ul>
1.02	<p><b>Bug fixes :-</b></p> <ul style="list-style-type: none"> <li>1. Changed ToRtr,ToSrce,Resource order to match the manual.</li> </ul>
1.03	<p><b>Modifications :-</b></p> <ul style="list-style-type: none"> <li>1. Added the ability to manage sources on intermediate routers.</li> <li>2. In the absence of a mask on the end of the route crosspoint command the VRD will use the mask supplied in the INI files 'Virtual Source' list. If this does not contain a valid mask the VRD will default to making a shared route with no resources.</li> <li>3. Clearer diagnostic information</li> </ul>
1.04	<p><b>Bug fixes :-</b></p> <ul style="list-style-type: none"> <li>1. Correction to database compilation routine</li> </ul>
1.05	<p><b>Bug fixes :-</b></p> <ul style="list-style-type: none"> <li>1. Correction to allow destination 1 to be first inter-matrix tieline</li> </ul>
1.06	<p><b>Modifications :-</b></p> <ul style="list-style-type: none"> <li>1. When a route connect fails -1 is returned to workstation requesting the route</li> </ul>
1.07	<p><b>Bug fixes :-</b></p> <ul style="list-style-type: none"> <li>1. 50ms inhibit added to prevent rapid VRD requests overloading buffers</li> </ul>
1.08	<p><b>Bug fixes :-</b></p> <ul style="list-style-type: none"> <li>1. Tx NetBIOS buffer allocation now being freed correctly</li> </ul>
1.09	<p><b>Modifications :-</b></p> <ul style="list-style-type: none"> <li>1. Replaced inhibit mechanism with more efficient message memory</li> </ul>
1.10	<p><b>Modifications :-</b></p>

	<ul style="list-style-type: none"> <li>1. Reduced scrolling text buffer to 100 lines.</li> <li>2. Combined Default Resource memory into one global block of 32k</li> <li>3. Inhibited Message Memory for testing purposes</li> </ul>
1.11	<b>Modifications :-</b> <ul style="list-style-type: none"> <li>1. Reinstated Message Memory</li> </ul>
1.12	<b>Modifications</b> <ul style="list-style-type: none"> <li>1. Increased message queue size</li> </ul> <b>Additions :-</b> <ul style="list-style-type: none"> <li>1. Added Tally save and reload functions</li> </ul>
1.13	<b>Modifications :-</b> <ul style="list-style-type: none"> <li>1. Returns -2 to WS requesting route when route cannot be made</li> </ul> <b>Bug fixes :-</b> <ul style="list-style-type: none"> <li>1. Bug fix to database modification routine</li> </ul>
1.14	<b>Additions :-</b> <ul style="list-style-type: none"> <li>1. Profile switch to prevent route table being cleared if required</li> </ul> <b>Bug fixes :-</b> <ul style="list-style-type: none"> <li>1. Bug fix to NetSendDG string length</li> </ul>
1.15	<b>Additions :-</b> <ul style="list-style-type: none"> <li>1. Added Tally save and reload functions</li> </ul>

---