Operation and Reference Manual

Logical Router Control

Imagine Communications Routing Switchers

Version 1.5

Delivering the Moment

imaginecommunications.com

Publication Information

Trademarks

Contact Information

Imagine Communications has office locations around the world. For domestic and international location and contact information, visit our Contact page (http://www.imaginecommunications.com/company/contact-us.aspx).

Support Contact Information

For domestic and international support contact information see:

- Support Contacts (http://www.imaginecommunications.com/services/technical-support.aspx)
- eCustomer Portal (http://support.imaginecommunications.com)

Contents

Manual Information

Audience

This manual is written for engineers, technicians, and operators responsible for installation, setup, maintenance, and/or operation.

Writing Conventions

This manual adheres to the following writing conventions.

Term or Convention	Description
Bold	Indicates dialog box, property sheet, field, button, check box, list box, combo box, menu, submenu, window, list, and selection names
Italics	Indicates email addresses, names of books and publications, and first instances of new terms and specialized words that need emphasis
CAPS	Indicates a specific key on the keyboard, such as ENTER, TAB, CTRL, ALT, DELETE
Code	Indicates variables or command-line entries, such as a DOS entry or something you type into a field.
>	Indicates the direction of navigation through a hierarchy of menus and windows
hyperlink	Indicates a jump to another location within the electronic document or elsewhere
Internet address	Indicates a jump to a Web site or URL

Obtaining Documents

Product support documents can be viewed or downloaded from our website. Alternatively, contact your Customer Service representative to request a document.

Document Revision History

Revision	Date	Comments
1.1	May 2010	Initial release for LRC 1.1
1.2	Jan 2012	Updated release for LRC 1.1
1.4	Aug 2012	Updated release for LRC 1.2
1.5	April 2014	Rebranded

Known Issues in Version 1.0

The following are issues known to be present with the initial release of LRC.

- **Inconsistent Matrix-level Lock/Protect Status and Override**
  If matrix level-outputs that are part of a logical destination are locked or protected at the matrix level (for example, using X-Y commands) and report inconsistent lock status (for example, some level outputs are locked/protected while others are free or locked/protected by a different user ID, the destination will report the highest level of protection active. The mixed or partial lock/protect status must be removed or overridden by the appropriate X-Y commands.
- **Disconnect**
  LRC Protocol v.1.0 does not support a source disconnect command.
- **Breakaway Status Reporting with Remotely Located Matrices**
  Logical Destinations including level outputs that are physically located on other frames may temporarily report breakaway crosspoint status after a successful take. The duration of the breakaway status reporting is approximately equal to the difference in propagation delay for matrix status reporting between the closest (local) frame and the most remote frame that are participating in the logical destination. Upon arrival of the last (successful) status message, the breakaway status should follow appropriately.

Introduction

This document describes the Logical Router Control Protocol (LRC), which allows control and monitoring of routers and router crosspoints based on the names and/or IDs assigned in the Logical Database.

Communications Requirements

To use the LRC/TCP interface the communicating devices must be attached to an Ethernet network, configured properly for IP communications and must support the LRC Socket interface. For details of whether a device supports the LRC Socket interface and how to enable its operation, please consult the devices' documentation.

The LRC/TCP interface listens for incoming connections on TCP port 52116. With TCP based connections, it is the responsibility of the connecting client to reestablish the connection in the event of network interruption or connection loss. Details of TCP socket programming are beyond the scope of this document.

Interface Performance

Because variability exists in network equipment, connection speeds, topologies, configuration, conditions and system load, there are no specified or guaranteed values for message response times or throughput defined or implied for this interface. Typically, conditions favorable for the best response times and throughput are achieved using a closed network LAN/VLAN exclusively dedicated to router control.

Transmission Format

LRC Messages may be sent over TCP socket connections (LRC over TCP).

The format for messages sent or received via the stream-oriented LRC/TCP interface is as follows:

```
~<LRC Message>\
```

~Opening Flag required on all LRC Messages
IMPORTANT: The message packet must not contain an Opening Flag character.

`<LRC Message>` The message packet field should contain a single Logical Router Control Protocol message.
\Closing Flag required on all LRC Messages
IMPORTANT: The message packet must not contain a Closing Flag character.

Message packets are sent and received in a continuous stream over the Socket interface. More than one LRC Packet may be transmitted per TCP packet, and LRC message packets can span TCP packets (the TCP protocol effectively hides this from the application, however).

LRC Message Format

General Form

The general form of an LRC message is as follows:

```
<lrc-message> ::= <type><op><args>
<type>::= one of the supported LRC message types
<op>::= one of the supported LRC "message operators"
<args>::= <arg>*[";"<arg>]
<arg>::= <arg name><arg-type><values>
<arg-name>::= alphanumeric name of the argument
<arg-type>::= supported LRC argument type
<values>::= "{"*<value>*[,<value>]"}"
```

```
Note: Empty sets are allowed.
```

Value

```
<value>::= sequence of any printable ascii chars required to express
argument value except "~","\", "{", ",",
or "}" (Spaces are valid)
```

Type

`<type>`

Example LRC message types are: (see elsewhere for full list)

- XPOINT (crosspoint control/status)
- LOCK (prevents changing a crosspoint’s status)
- PROTECT (prevents others from changing crosspoint status)

Operators

`<op>`

Supported LRC message operators include:

:(change request) – request to change status
!(change notification) – report status change
?(query) – query current status

%(query response) – query response

Arguments

```
<arg-type>
```

Supported LRC message argument types include:

$ (string)– argument list is interpreted as a string
# (numeric)– argument list is interpreted as a number
& (UTF-8) – argument list is interpreted as a UTF-8 encoded string

Logical Names

Logical names (e.g. sources and destinations) use the following syntax:

```
<logical name> ::= <name>*[.<channel>]
```

Note that parameter fields may appear in any order in the message (e.g. "Source" field may appear before "Destination" field or vice versa).

Unicode Support

To facilitate use and display of system databases created using different languages, the Logical Router Control Protocol supports Unicode characters via UTF-8 encoded arguments for certain fields that carry user-defined name information such as source, destination, channel and user names.

UTF-8 encodes ASCII characters 32-127 in one byte, while still supporting Unicode characters in multi-byte sequences. In order to maintain readability and compatibility with devices with no native Unicode support, the actual UTF-8 bytes (code units) are transmitted in LRC messages as ASCII-hex encoded strings.

For example in the word "Français" the character U+00E7 ("ç", or "Latin small letter c with cedilla") is encoded in UTF-8 as the code units C3 A7. When transmitted as an LRC value, it would be represented in the UTF-8 value string using ASCII-hex string value "C3A7." The fully UTF-8 encoded example is show below:

```
UTF-8 encoded Source named "Français" S&{4672616EC3A7616973}
```

In another example, the character U+00F1 ("ñ", or "Latin small letter n tilde") in the word "Español" is encoded in UTF-8 as the code units C3 B1 and is encoded in the LRC value string value as "C3B1":

```
UTF-8 encoded Destination named "Español" D&{45737061C3B16F6C}
```

Network (big-endian) byte ordering is used for the hex strings. Aside from the encoded string values, the LRC message syntax is unchanged. (Note that if UTF-8 encoded values are used, it is permissible for strings to contain characters such as backslash, comma, or tilde that would otherwise be disallowed.) Specific message examples using UTF-8 encoded fields are provided in the "Examples" section.

Unicode reporting mode may be enabled via the "CONNECTION" command

Fields that support Unicode reporting:

- Logical Source Name
- Logical Destination Name
- Channel Name
- User Name (outbound only)

Supported Commands

The following table shows the list of supported LRC commands, a brief description of the command's function, which version of the protocol introduced the command, the list of which forms (change, notification, query and response) are supported by which LRC command:

Command	**LRC Ver**	**Description**	**Change (:)**	**Notify (!)**	**Query (?)**	**Resp (%)**
XPOINT	1.1	Crosspoint control/ source status query command	●	●	●	●
LOCK	1.1	Destination Lock/ lock status query command	●	●	●	●
PROTECT	1.1	Destination Protect/ protect status query command	●	●	●	●
CHANNELS	1.1	Logical channel (level) query / response			●	●
DEST	1.1	Logical destination query / response			●	●
SRC	1.1	Logical source query / response			●	●
DBCHANGE	1.1	Database Change notification		●		
CONNECTION	1.1	Connection (session) control	●			
PROTOCOL	1.1	Protocol information query			●	●
XBUFFER	1.1	Crosspoint buffer control command	●			
XDISCONNECT	1.1	Disconnect logical crosspoint command	●			
XPRESET	1.1	Crosspoint preset command	●			
XSALVO	1.1	Crosspoint salvo query or execution command	●	●	●	●

The following sections provide a detailed explanation of each LRC command's syntax, purpose and use.

XPOINT

The Crosspoint command (XPOINT) is used to issue take requests, monitor status of and send change notifications for logical crosspoints. It supports the following fields:

Field	Supported value type(s)			Used in operation				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
D	●	●*	●	●	●	●	●	Logical Destination Name(s)	1.1
S	●	●*	●	●	●		●	Logical Source Name(s)	1.1
U			●	●				User requesting crosspoint change	1.1

- **D**: Specifies one or more logical destinations to control, status or query. In Query (?) or Change (:) operations, an empty or missing destination list resolves to all logical destinations. Supports UTF-8 encoding.
- **S**: Specifies one or more logical sources. If more than one source is listed, the number of sources specified must equal the number of destinations listed. Ignored in 'query' operations. Supports UTF-8 encoding.
- An empty list in status (!) or reply (%) messages indicates logical source breakaway status.
- **U**: User ID requesting the crosspoint change. Used to control access if destination is locked or protected.

*NOTE: version 1.1 supports UTF-8 UMD/Multiviewer status reporting for XPOINT, LOCK, and PROTECT status messages (i.e. '!' and '%'). Use of UTF-8 for crosspoint control/query messages (':' and '?') will be enabled in a subsequent version

XPOINT Examples

```
~XPOINT:S${SAT 1};D${MON 6}\
```

Request crosspoint take for source SAT1 (by name) to destination MON6

Source = "SAT 1" and Dest = "MON 6". Note that the order of the arguments is not important.

```
~XPOINT!D${MON6};S${SAT1}\
~XPOINT!D#{6};S#{1}\
```

Change notification for destination MON6 switching to source SAT1. Note that the XPOINT command notifies by both name and number. This is helpful for applications that are migrating from a numerically based routing control system to a name based routing control system. Having statuses that include both name and number will help to facilitate the transition to a name based control system.

```
~XPOINT?D${MON6}\
```

Query source switched to destination MON6.

```
~XPOINT%D${MON6};S${SAT1}\
~XPOINT%D#{6};S#{1}\
```

Report source SAT1 (1) switched to destination MON6 (6).

```
~XPOINT:S${SAT1.SD};D${MON6.SD}\
```

Request (breakaway) crosspoint take for source SAT1 to destination MON6 on "SD" channel (level)

```
~XPOINT!S${SAT1.SD};D${MON6.SD}\
~XPOINT!S#{1.1};D#{6.1}\
```

Reports crosspoint status change for source SAT1 to destination MON6 on "SD" channel (level #1). Note both numeric and name responses are returned.

```
~XPOINT!S${};D${MON6}\
~XPOINT!S#{};D#{6}\
```

Additionally, a logical breakaway crosspoint status will be returned for destination MON6 (empty list) if the above "level take" command placed the crosspoint in breakaway status.

```
~XPOINT:S${Src 2.Level 1};D${Dst 6.Level 1}\
```

Assume this command will place Dst 6 in breakaway.

```
~XPOINT!D#{6};S#{}\
~XPOINT!D${Dst6};S${}\
~XPOINT!D#{6.1};S#{1.1}\
~XPOINT!D${Dst 6.Level 0};S${Src 1.Level0}\
~XPOINT!D#{6.2};S#{2.2}\
~XPOINT!D${Dst6.Level 1};S${Src 2.Level 1}\
```

This is the full set of messages returned for Dst 6 which is in breakaway status on 2 channels. Again note that both names and numbers are returned.

```
~XPOINT?D${MON6.SD}\
```

Query "SD" channel (level) crosspoint for destination MON6

```
~XPOINT:S${SAT 1};D${MON 6,MON 7,MON 8,MON 9}\
```

Request crosspoint take for source SAT1 (by name) to destinations MON 6, MON 7, MON 8 and MON 9. Note that leading spaces in source and destination names will be assumed to be part of the name. Do not include a space after the comma.

```
~XPOINT!S${SAT1};D${MON6}\
~XPOINT!S#{1};D#{6}\
```

Notification for destination MON6 switching to source SAT1. The above would be the only response for the previous command if MON 9, MON 8, and MON 7 were already routed to SAT 1.

```
~XPOINT?\
```

Query logical crosspoint status for all destinations

```
~XPOINT%D#{1};S#{}\~XPOINT%D${Dst 1};S${}\
~XPOINT%D#{2};S#{}\~XPOINT%D${Dst 2};S${}\
~XPOINT%D#{3};S#{}\~XPOINT%D${Dst 3};S${}\
~XPOINT%D#{4};S#{}\~XPOINT%D${Dst 4};S${}\
~XPOINT%D#{5};S#{}\~XPOINT%D${Dst 5};S${}\
~XPOINT%D#{6};S#{}\~XPOINT%D${Dst 6};S${}\
~XPOINT%D#{7};S#{}\~XPOINT%D${Dst 7};S${}\
~XPOINT%D#{8};S#{}\~XPOINT%D${Dst 8};S${}\
~XPOINT%D#{9};S#{}\~XPOINT%D${Dst 9};S${}\
~XPOINT%D#{10};S#{}\~XPOINT%D${Dst 10};S${}\
~XPOINT%D#{11};S#{}\~XPOINT%D${Dst 11};S${}\
~XPOINT%D#{12};S#{}\~XPOINT%D${Dst 12};S${}\
~XPOINT%D#{13};S#{}\~XPOINT%D${Dst 13};S${}\
~XPOINT%D#{14};S#{}\~XPOINT%D${Dst 14};S${}\
~XPOINT%D#{15};S#{}\~XPOINT%D${Dst 15};S${}\
~XPOINT%D#{16};S#{}\~XPOINT%D${Dst 16};S${}\
```

Example responses for a query of all destinations where each destination is in breakaway status.

Unicode (UTF-8) XPOINT Examples

```
~XPOINT!D&{4473742031};S&{534154031.4672616EC3A7616973}\
```

Crosspoint Status with UTF-8 encoded values for
Destination ="Dst 1" and Source = "SAT 1.Français"

```
~XPOINT!D&{4473742032};S&{534154031.45737061C3B16F6C}\
```

Crosspoint Status for Destination ="Dst 2" and Source = "SAT 1.Español"

```
~XPOINT!D&{4473742032};S&{534154031.CE95CEBBCEBBCEB7CEBDCEB9CEBACEAC}\
```

Crosspoint Status for destination "Dst 2" and Source = “SAT 1.Ελληνικά” (Greek)

```
~XPOINT!D&{E8BE93E587BAE7ABAF32};S&{E8A19BE6989F31.E7AE80E4BD93E4B8ADE
69687E78988}\
```

Take destination "输出端2" ("Output port 2") to source "衛星1.繁體中文版" ("SAT 1.Chinese")

LOCK

The Destination Lock command (LOCK) is used to secure a destination from further Crosspoint status changes by any User ID including the lock owner. It supports the following fields:

Field	Supported value type(s)			Used in Form(s)				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
D	●	●	●	●	●	●	●	Logical Destination Name(s)	1.1
V	●			●	●		●	Status Value ("ON" or "OFF")	1.1
U			●	●	●		●	User (owner or requestor)	1.1
O	●	●	●	●	●		●	Override by User (owner or requestor)	1.1

- **D**: Specifies one or more logical destinations to control, status or query. In Query (?) or Change (:) operations, empty or missing destination list resolves to all logical destinations. Supports UTF-8 encoding.
- **V**: Specifies value for lock status. Valid values are "ON" and "OFF"
- Empty list in status (!) or reply (%) messages indicates logical source breakaway status.
- **U**: User ID requesting the lock status change.
- **O**: Override lock status (force lock status change)

NOTE: version 1.1 supports UTF-8 UMD/Multiviewer status reporting for XPOINT, LOCK, and PROTECT status messages (i.e. '!' and '%'). Use of UTF-8 for crosspoint control/query messages (':' and '?') will be enabled in a subsequent version.

Lock Examples

```
~LOCK:D${MON6};V${ON};U#{20}\
```

User "20" requests lock for destination MON6

```
~LOCK!D${MON6};V${ON};U#{20}\
```

System reports MON6 locked by User ID "20"
User is "20" and Value is "ON"

```
~LOCK?D${MON6}\
```

Requests lock status for destination MON6

```
~LOCK%D${MON6};V${ON};U#{20}\
```

System reports MON6 Locked by User ID "20"

```
~LOCK:D${MON6};U#{20};V${OFF}\
```

User ID "20" requests unlock for destination MON6

```
~LOCK!D${MON6};V${OFF};U#{20}\
```

System reports MON6 unlocked (Value is "OFF"))

PROTECT

The Destination Protect command (PROTECT) is used to secure a destination from further changes requested by anyone except the lock owner (user). It supports the following fields:

Field	**Supported value type(s)**			**Used in Form(s)**				**Description**	**Min Ver**
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
D	●	●	●	●	●	●	●	Logical Destination Name(s)	1.1
V	●			●	●		●	Status Value ("ON" or "OFF")	1.1
U			●	●	●		●	User Name (owner or requestor)	1.1

- **D**: Specifies one or more logical destinations to control, status or query. In Query (?) or Change (:) operations, empty or missing destination list resolves to all logical destinations. Supports UTF-8 encoding.
- **V**: Specifies value for protect status. Valid values are "ON" and "OFF"
- Empty list in status (!) or reply (%) messages indicates logical source breakaway status.
- **U**: User requesting the protect status change.
- **O**: Override protect status (force status change)

Protect Examples

```
~PROTECT:D${MON6};U#{20};V${ON}\
```

User ID "20" requests protect for destination MON6

```
~PROTECT!D${MON6};U#{20};V${ON}\
```

System reports MON6 protected by User "20"

```
~PROTECT?D${MON6}\
```

Requests protect status for destination MON6

```
~PROTECT%D${MON6};V${ON};U#{20}\
```

System reports MON6 protected by User "20"

CHANNELS

The Channels command is used to query the system for the list of valid system channels (levels):

Field	Supported value type(s)			Used in Form(s)				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
I			●				●	Channel (Level) ID list	1.1
NAME	●	●					●	Channel (Level) name list	1.1

- **I**: Reports a list of the channel (level) Numeric IDs.
- **NAME**: Reports a list the names that correspond to the IDs. One name will be included for each reported ID.

Examples

```
~CHANNELS?\
```

Requests system channel information

```
~CHANNELS%I#{1,2};NAME${HD,SD}\
```

Reply with system channel information: 2 channels (level 1=HD, level 2=SD)

DEST

The Destination command (DEST) is used query the system for the list of valid logical Destinations. Physical port mapping can also be determined:

Field	Supported value type(s)			Used in Form(s)				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
Q	●					●	●*	Query (COUNT, NAME, CHANNELS, PHYSICAL). Response signals end of response data.	1.1 * 1.2
ID			●			●	●	Destination ID	1.1
CHANNELS	●	●					●	Destination valid channels	1.1
COUNT			●				●	Destination count	1.1
NAME	●	●				●	●	Destination name	1.1
LOCATION	●					●	●	Location of a slot	1.2
PHYSICAL	●					●	●	Physical location of a channel	1.2

- **Q**: used in query request to specify what information is to be queried. Valid queries are: COUNT, NAME, CHANNELS, PHYSICAL. Used in response to indicate end of response data.
- **COUNT**: count of logical destinations is the system
- **NAME**: name of destination
- **CHANNELS**: list of valid channels (optionally for a specified logical destination)
- **PHYSICAL**: Physical location of a channel in the form (frame_signature#[hex].slot#.port#.channel#.type.zone). type = VIDEO, AUDIO. zone = INPUT, OUTPUT, XPOINT, SYNC, TDM. frame_signature#[hex] = unique identifier for the frame as 12 hexadecimal characters.
- **LOCATION**: Physical location of a slot in the form (frame_signature#[hex].slot#). frame_signature#[hex] = unique identifier for the frame as 6 hexadecimal characters.
- **ID**: Numeric ID for channel or destination Examples:

DEST Examples

```
~DEST?Q${COUNT}\
```

Queries total number of logical destinations in system

```
~DEST%COUNT#{16}\
~DEST%Q${COUNT}\
```

Reply with count of destinations in system

```
~DEST?Q${NAME}\
```

Queries names for all destinations in system

```
~DEST%I#{1};NAME${Dst 1}\
~DEST%I#{2};NAME${Dst 2}\
~DEST%I#{3};NAME${Dst 3}\
~DEST%I#{4};NAME${Dst 4}\
~DEST%I#{5};NAME${Dst 5}\
~DEST%I#{6};NAME${Dst 6}\
~DEST%I#{7};NAME${Dst 7}\
~DEST%I#{8};NAME${Dst 8}\
~DEST%I#{9};NAME${Dst 9}\
~DEST%I#{10};NAME${Dst 10}\
~DEST%I#{11};NAME${Dst 11}\
~DEST%I#{12};NAME${Dst 12}\
~DEST%I#{13};NAME${Dst 13}\
~DEST%I#{14};NAME${Dst 14}\
~DEST%I#{15};NAME${Dst 15}\
~DEST%I#{16};NAME${Dst 16}\
~DEST%Q${NAME}\
```

Responses

```
~DEST?Q${NAME};I#{1,3,5}\
```

Queries names for destinations with numeric ID 1,3 and 5

```
~DEST%I#{1};NAME${Dst 1}\
~DEST%I#{3};NAME${Dst 3}\
~DEST%I#{5};NAME${Dst 5}\
~DEST%Q${NAME}\
```

Responses

```
~DEST?Q${CHANNELS}\
```

Query channels for all destinations

```
~DEST%CHANNELS${Level 0,Level 1,Level 2,Level 3};I${Dst 1}\
~DEST%CHANNELS${Level 0,Level 1,Level 2,Level 3};I${Dst 2}\
~DEST%CHANNELS${Level 0,Level 1,Level 2,Level 3};I${Dst 3}\
~DEST%CHANNELS${Level 0,Level 1,Level 2,Level 3};I${Dst 4}\
…
~DEST%Q${CHANNELS}\
```

Responses

```
~DEST?Q${CHANNELS};I#{2}\
```

To query channels for a single destination by ID#:

```
~DEST%CHANNELS#{1,2,3,4};I#{3}\
~DEST%Q${CHANNELS}\
```

Response

```
~DEST?Q${CHANNELS};I${Dst 1}\
```

To query a single destination by Name (in this case dest name is Dst 1)

```
~DEST%CHANNELS${Level 1,Level 2,Level 3};I${Dst 1}\
~DEST%Q${CHANNELS}\
```

Response

```
~DEST?Q${PHYSICAL};LOCATION${A5A5A5A5A5A5.15}\
```

Queries for the physical to logical mapping for the specified device location.
All destinations for the requested slot are returned.

```
~DEST%PHYSICAL${A5A5A5A5A5A5.15.1.1.VIDEO.OUTPUT};I#{120.1};NAME${Dst
120.Video}\
~DEST%PHYSICAL${A5A5A5A5A5A5.15.1.1.AUDIO.OUTPUT};I#{120.2};NAME${Dst
120.AudioLR}\
~DEST%PHYSICAL${A5A5A5A5A5A5.15.1.2.AUDIO.OUTPUT};I#{120.2};NAME${Dst
120.AudioLR}\
~DEST%PHYSICAL${A5A5A5A5A5A5.15.2.1.VIDEO.OUTPUT};I#{121.1};NAME${Dst
121.Video}\
~DEST%PHYSICAL${A5A5A5A5A5A5.15.2.1.AUDIO.OUTPUT};I#{121.2};NAME${Dst
121.AudioLR}\
~DEST%PHYSICAL${A5A5A5A5A5A5.15.2.2.AUDIO.OUTPUT};I#{121.2};NAME${Dst
121.AudioLR}\
…
~DEST%Q${PHYSICAL}\
```

Responses

```
~DEST?Q${PHYSICAL};NAME${Dst 123}\
```

Queries for the physical to logical mapping for the specified destination.

```
~DEST%PHYSICAL${A5A5A5A5A5A5.15.4.1.VIDEO.OUTPUT};I#{123.1};NAME${Dst
123.Video}\
~DEST%PHYSICAL${A5A5A5A5A5A5.15.4.1.AUDIO.OUTPUT};I#{123.2};NAME${Dst
123.AudioLR}\
~DEST%PHYSICAL${A5A5A5A5A5A5.15.4.2.AUDIO.OUTPUT};I#{123.2};NAME${Dst
123.AudioLR}\
~DEST%Q${PHYSICAL}\
```

Normal Response

```
~DEST%PHYSICAL${};I#{};NAME${Dst 123}\
~DEST%Q${PHYSICAL}\
```

Response where no mapping exists

```
~DEST?Q${NAME};PHYSICAL${A5A5A5A5A5A5.15.4.1.VIDEO.OUTPUT}\
```

Queries names for the physical destination requested

```
~DEST%I#{123.1};NAME${Dst
123.Video};PHYSICAL${A5A5A5A5A5A5.15.4.1.VIDEO.OUTPUT}\
~DEST%Q${NAME}\
```

Normal Response

```
~DEST%I#{};NAME${};PHYSICAL${A5A5A5A5A5A5.15.4.1.VIDEO.OUTPUT}\
~DEST%Q${NAME}\
```

Response where no mapping exists

SRC

The Source command (SRC) is used to query the system for information about valid logical Sources:

	Supported value type(s)			Used in Form(s)					
Field	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%	**Description**	**Min Ver**
Q	●					●	●*	Query Item (COUNT, NAME, CHANNELS, PHYSICAL). Response signals end of response data.	1.1 * 1.2
ID			●			●	●	Source ID	1.1
CHANNELS	●	●					●	Source valid channels	1.1
COUNT			●				●	Source Count	1.1

NAME	●	●				●	●	Source name	1.1
LOCATION	●					●	●	Location of a slot	1.2
PHYSICAL	●					●	●	Physical location of a channel	1.2

- **Q**: Used in query request to specify what information is to be queried. Valid queries are: COUNT, NAME, CHANNELS, PHYSICAL. Used in response to indicate end of response data.
- **COUNT**: count of logical sources is the system
- **NAME**: name of source
- **CHANNELS**: list of valid channels (optionally for a specified logical source)
- **PHYSICAL**: Physical location of a channel in the form (frame_signature#[hex].slot#.port#.channel#.type.zone). type = VIDEO, AUDIO. zone = INPUT, OUTPUT, XPOINT, SYNC, TDM. frame_signature#[hex] = unique identifier for the frame as 12 hexadecimal characters.
- **LOCATION**: Physical location of a slot in the form (frame_signature#[hex].slot#). frame_signature#[hex] = unique identifier for the frame as 6 hexadecimal characters.
- **ID**: numeric ID for channel or source

SRC Examples

```
~SRC?Q${COUNT}\
```

Queries total number of logical sources in system

```
~SRC%COUNT#{16}\
~SRC%Q${COUNT}\
```

Responses

```
~SRC?Q${NAME}\
```

Queries names of all sources is system

```
~SRC%I#{1};NAME${SAT 1}\
~SRC%I#{2};NAME${SAT 2}\
~SRC%I#{3};NAME${SAT 3}\
~SRC%I#{4};NAME${SAT 4}\
~SRC%I#{5};NAME${SAT 5}\
~SRC%I#{6};NAME${SAT 6}\
~SRC%I#{7};NAME${SAT 7}\
~SRC%I#{8};NAME${SAT 8}\
~SRC%I#{9};NAME${SAT 9}\
~SRC%I#{10};NAME${SAT 10}\
~SRC%I#{11};NAME${SAT 11}\
~SRC%I#{12};NAME${SAT 12}\
~SRC%I#{13};NAME${In 13}\
~SRC%I#{14};NAME${In 14}\
~SRC%I#{15};NAME${In 15}\
~SRC%I#{16};NAME${In 16}\
~SRC%Q${NAME}\
```

Responses

```
~SRC?I#{2,4,6};Q${NAME}\
```

Queries names for sources with numeric ID 2, 4 and 6
(Note that parameter order is not important)

```
~SRC%I#{2};NAME${SAT 2}\
~SRC%I#{4};NAME${SAT 4}\
~SRC%I#{6};NAME${SAT 6}\
~SRC%Q${NAME}\
```

Responses

```
~SRC?I#{2,4,6};Q${CHANNELS}\
```

Queries valid channels for sources with (numeric) IDs 2, 4, and 6

```
~SRC%CHANNELS#{1,2,3,4};I#{2}\
```

Src with ID# 2 is valid on channels with ID 1, 2, 3 and 4 (levels 0,1,2,3) (Channels 1-16 map to levels 0-15)

```
~SRC%CHANNELS#{1,2,3,4};I#{4}\
~SRC%CHANNELS#{1,2,3,4};I#{6}\
~SRC%Q${CHANNELS}\
```

Responses

```
~SRC?I${SAT 1,SAT 2,SAT 3};Q${CHANNELS}\
```

Queries valid channels for sources with names "SAT 1", "SAT 2", and "SAT 3"

```
~SRC%CHANNELS${HD,SD,Video,AES-TDM};I${SAT 2}\
```

Src with ID (name) "SAT 2" is valid on the channels with names "HD", "SD", "Video" and "AES-TDM" (currently maps to levels with same names)

```
~SRC%CHANNELS${HD,SD,Video,AES-TDM};I${SAT 3}\
~SRC%CHANNELS${HD,SD,Video,AES-TDM};I${SAT 4}\
~SRC%Q${CHANNELS}\
```

Responses

```
~SRC?Q${CHANNELS};I${Dst 2}\
```

Query with name followed by response

```
~SRC%CHANNELS${Level 1,Level 2,Level 3};I${Dst 2}\
~SRC%Q${CHANNELS}\
```

Response

```
~SRC?Q${PHYSICAL};LOCATION$(A5A5A5A5A5A5.15}\
```

Queries for the physical to logical mapping for the specified device and slot.
All sources for the requested slot are returned.

```
~SRC%PHYSICAL${A5A5A5A5A5A5.15.1.1.VIDEO.INPUT};I#{20.1};NAME${Sat
20.Video}\
~SRC%PHYSICAL${A5A5A5A5A5A5.15.1.1.AUDIO.INPUT};I#{20.2};NAME${Sat
20.AudioLR}\
~SRC%PHYSICAL${A5A5A5A5A5A5.15.1.2.AUDIO.INPUT};I#{20.2};NAME${Sat
20.AudioLR}\
~SRC%PHYSICAL${A5A5A5A5A5A5.15.2.1.VIDEO.INPUT};I#{21.1};NAME${Sat
21.Video}\
~SRC%PHYSICAL${A5A5A5A5A5A5.15.2.1.AUDIO.INPUT};I#{21.2};NAME${Sat
21.AudioLR}\
~SRC%PHYSICAL${A5A5A5A5A5A5.15.2.2.AUDIO.INPUT};I#{21.2};NAME${Sat
21.AudioLR}\
…
~SRC%Q${PHYSICAL}\
```

Responses

```
~SRC?Q${PHYSICAL};NAME${Sat 23, Sat 24}\
```

Queries for the physical to logical mapping for the specified source(s).

```
~SRC%PHYSICAL${A5A5A5A5A5A5.15.4.1.VIDEO.INPUT};I#{23.1};NAME${Src
23.Video}\
~SRC%PHYSICAL${A5A5A5A5A5A5.15.4.1.AUDIO.INPUT};I#{23.2};NAME${Src
23.AudioLR}\
~SRC%PHYSICAL${A5A5A5A5A5A5.15.4.2.AUDIO.INPUT};I#{23.2};NAME${Src
23.AudioLR}\
~SRC%PHYSICAL${A5A5A5A5A5A5.15.5.1.VIDEO.INPUT};I#{24.1};NAME${Src
24.Video}\
~SRC%PHYSICAL${A5A5A5A5A5A5.15.5.1.AUDIO.INPUT};I#{24.2};NAME${Src
24.AudioLR}\
~SRC%PHYSICAL${A5A5A5A5A5A5.15.5.2.AUDIO.INPUT};I#{24.2};NAME${Src
24.AudioLR}\
~SRC%Q${PHYSICAL}\
```

Responses

```
~SRC?Q${NAME};PHYSICAL${A5A5A5A5A5A5.15.4.1.VIDEO.INPUT}\
```

Queries names for the physical source requested

```
~SRC%NAME${Src
23.Video};I#{23.1};PHYSICAL${A5A5A5A5A5A5.15.4.1.VIDEO.INPUT}\
~SRC%Q${NAME}\
```

Response

DBCHANGE

The Database Change notification message (DBCHANGE) is sent asynchronously by the LRC server to alert clients when there's been an update to the logical database.

Field	Supported value type(s)			Used in Form(s)				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
DATA	●				●			Changed Data (currently only one response 'ALL')	1.1

- **DATA**: type of change (currently "ALL" is the only support value)

The LRC protocol server will send the following notification to attached clients when there has been a change in the logical database:

```
~DBCHANGE!DATA${ALL}\
```

This message may be used by the client application to trigger a request for the updated source, destination, channel names and/or logical crosspoint status.

CONNECTION

Field	Supported value type(s)			Used in Form(s)				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
MODE	●			●				Set connection notification mode	1.1

The CONNECTION message may be sent by a client to adjust per-session settings in the LRC connection.

- **MODE**: The "MODE" parameter selects the user-data reporting mode (ASCII or UTF-8) for all asynchronous messages sent to the client.
  Valid values are:
  - $: Report user data as ASCII strings
  - &: Report user data Unicode (UTF-8)
- User Data includes:
  - Channel (level) names
  - Source names
  - Destination names
  - User Names

Connection Examples

```
~CONNECTION:MODE${&}\
```

Sets to Unicode mode

```
~CONNECTION:MODE${$}\
```

Sets to ASCII mode

PROTOCOL

The PROTOCOL command allows query of protocol version information.

Field	Supported value type(s)			Used in Form(s)				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
Q	●					●		Query Item (NAME, VERSION)	1.1
NAME	●						●	Protocol Name	1.1
VERSION	●						●	Protocol version	1.1

- **Q**: used in query request to specify what information is to be queried. Valid queries are:
  - **VERSION**: request LRC protocol version (e.g. 1.1)
  - **NAME**: request protocol name
- **VERSION**: reports LRC protocol version in form "Major.Minor" (e.g. 1.1)
- **NAME**: name of protocol ("Logical Router Control Protocol")

Protocol Examples

```
~PROTOCOL?Q${NAME}\
~PROTOCOL%NAME${Logical Router Control Protocol}\
```

Query name with response

```
~PROTOCOL?Q${VERSION}
~PROTOCOL%VERSION${1.1}\
```

Query version with response

XBUFFER

This command (XBUFFER) allows control of crosspoint buffers:

Field	Supported value type(s)			Used in Form(s)				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
U			●	●				User	1.1
F	●			●				Operation flag	1.1

- **U**: User requesting the operation.
- **F**: Flag specifying the operation to be performed. Valid operations are:
  - **EXECUTE**: execute stored crosspoint commands
  - **CLEAR**: Clear command crosspoint buffer

XBUFFER Examples

~XBUFFER:F${EXECUTE};U#{1}\

Execute buffered crosspoint commands. This will be followed by crosspoint status changes (i.e. XPOINT!)

XDISCONNECT

The Crosspoint Disconnect command (XDISCONNECT) is used to disconnect logical crosspoints. It supports the following fields:

Field	Supported value type(s)			Used in operation				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
D	●		●	●				Logical Destination Name(s)	1.1
U			●	●				User requesting crosspoint change	1.1

- **D**: Specifies one or more logical destinations to disconnect, Empty or missing destination list resolves to all logical destinations. (NOTE: Underlying hardware must support disconnect for this to actually disconnect sources).
- **U**: User ID requesting the crosspoint disconnect. Used to control access if destination is locked or protected.

XDisconnect Examples

```
~XDISCONNECT:D${ALLD 1}\
```

Disconnect destination

```
~XPOINT!D${ALLD 1};S${}\
~XPOINT!D${ALLD 1};S${}\
~XPOINT!D${ALLD 1};S${}\
~XPOINT!D${ALLD 1};S${}\
~XPOINT!D${ALLD 1};S${}\
```

Responses

XPRESET

The command (XPRESET) is used to issue take requests, monitor status of and send change notifications for logical crosspoints. It supports the following fields:

Field	Supported value type(s)			Used in operation				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
D	●		●	●				Logical Destination(s)	1.1
S	●		●	●				Logical Source(s)	1.1
U			●	●				User requesting crosspoint change	1.1

- **D**: Specifies one or more logical destinations to control, status or query. In Query (?) or Change (:) operations, empty or missing destination list resolves to all logical destinations.
- **S**: Specifies one or more logical sources. If more than one source is listed, the number of sources specified must equal the number of destinations listed.
- Empty list in status (!) or reply (%) messages indicates logical source breakaway status.
- **U**: User ID requesting the crosspoint preset. Used to control access if destination is locked or protected.

XPreset Examples

```
~XPRESET:D#{2};S#{1};U#{1}\
```

Preset a crosspoint command for destination #2. Note, there is no response to an XPRESET command.

XSALVO

This command (XSALVO) allows query or execution of crosspoint salvos.

Field	Supported value type(s)			Used in Form(s)				Description	Min Ver
	ASCII ($)	UTF-8 (&)	Numeric (#)	:	!	?	%		
ID	●		●	●		●		Salvo Identifier	1.1
U			●	●				User	1.1
F	●			●				Execution Flags	1.1

- **ID**: Salvo Identifier
- **U**: User requesting the lock/protect status change.
- **F**: Flags. (For use with ':') XSALVO Currently Supports The following flags:
  - **TAKE**: Takes salvo(s) (default behavior if no flags specified)
  - **PRESET**: causes specified salvo(s) to be stored in the preset buffer
  - **LOCK**: locks salvo crosspoints after execution
  - **PROTECT**: protects salvo crosspoints after execution
  - **UNLOCK**: unlocks salvo crosspoints before execution
  - **NOTAKE**: inhibits crosspoint take; (permits locking/protecting salvo destinations)

Using the LRC/TCP Interface

Establishing a Connection

The LRC/TCP interface listens for incoming TCP connections on port 52116. Once the client application establishes (opens) a socket connection, messages may be sent to the interface. The client will receive messages as the LRC control system transmits them. No handshaking or polling is required. The underlying connection monitoring is provided by TCP, which, depending on the client platform, may report socket status to the application layer. Clients should monitor the socket connection for dropped connections and react to the interruption as appropriate. (Note: Techniques for checking whether a socket is still connected vary by platform. For details please refer to your platform's documentation).

Terminating a Connection

No specific termination handshaking is required to close the connection. The client may close its TCP socket at any time to end the LRC/TCP session.

Interface Behavior

The LRC/TCP interface is designed to be asynchronous and event-driven. Devices using the interface should monitor the received messages for relevant status updates rather than 'poll' the system for status periodically. Continuously polling the system will cause extra updates to be transmitted resulting in increased message traffic that can slow system response by delaying receipt of switch requests and status updates.

Important:

A Client using the LRC Socket can expect to receive and should ignore LRC message types or fields within messages that it does not process or that it is not aware of. This allows for forward compatibility and robustness across different versions of clients and devices. Transmission on the LRC System operates in a "negative acknowledgment" manner in that a transmitting device does not presume the message will be received.

Once the connection is established, transmission and receipt of messages may begin. All bytes received by the socket interface are ignored until the Opening Flag is received (the '~' character). The interface should accumulate all data sent until the closing flag (the '\' character) is received. After the closing flag has been received, the message may be processed. Any data sent between the closing flag and the next transmitted opening flag is ignored.

Reporting Mode

The LRC/TCP interface supports per-connection selection of either ASCII or UTF-8 for string data fields containing logical database information (e.g. source, destination channel and user names). Asynchronous updates (change notifications with '!' operator) and status query responses ('%') will use the requested mode for reporting fields that support the indicated data types.

For example:

```
~CONNECTION:MODE${$}\
```

Selects ASCII string ($) reporting mode

```
~CONNECTION:MODE${&}\
```

Selects UTF-8 string (&) reporting mode

The default reporting mode is ASCII string ($).

NOTE: version 1.1 supports UTF-8 UMD/Multiviewer status reporting for XPOINT, LOCK, and PROTECT status messages (i.e. '!' and '%'). Use of UTF-8 for crosspoint control/query messages (':' and '?') will be enabled in a subsequent version.

Notification Control

Beginning with LRC Protocol version 1.1, some asynchronous notification messages may be enabled or disabled by the client connection. The following messages may be enabled or disabled:

- DBCHANGE
- LOCK
- PROTECT
- XPOINT
- XSALVO

For example, to enable salvo change notifications, use the following syntax:

```
~CONNECTION:ENABLE${XSALVO}\
```

To disable salvo change notifications, use the following syntax:

```
~CONNECTION:DISABLE${XSALVO}\
```

Status Behavior

Status is reported in response to a change or query at the highest signal grouping possible (typically top-level logical in follow situations).

Follow Status

- Status reports "complete" logical source only.
- If source and destination do not have complete overlap (e.g. if a video-only source is switched to a destination defined with video and audio), "extra" destination statuses will be reported.

Breakaway Status

- Logical Destination reports ‘breakaway’ using no source name (empty brackets) for top level status upon initial breakaway status change:
  ```
  ~XPOINT!D${Dst 1};S${}\
  ```
- Subsequent breakaway status reports per channel (level) as appropriate at the highest level possible. (e.g. if a source is consistent at the logical level, then report status there otherwise drill down as required):
  ```
  ~XPOINT!D${Dst 1.HD};S${SAT 1.HD}\
  ~XPOINT!D${Dst 1.SD};S${SAT 2.SD}\
  ~XPOINT!D${Dst 1.AES};S${SAT 2.AES}\
  ~XPOINT!D${Dst 1.Video};S${SAT 2.Video}\
  ```
- If logical a Destination switches to a ‘complete’ logical source, reporting reverts to top level ("follow") status:

  ```
  ~XPOINT!D${Dst 1};S${SAT 2}\
  ```

Status Messages

The table below provides examples of expected status behavior for LRC messages, contrasted with XY messages occurring simultaneously in the system. The example assumes a 4-level (channel) database configuration with Level 0 = "HD", Level 1="SD", Level 2= "AES" and Level 3 = "Video".

Table 1: Status Example comparing XY and LRC messages for breakaway/follow status conditions

XY Messages	LRC Messages
Switched Logical Destination 1 to Logical Source 1 (Name is "SAT 1")	

`S:00,0`
`S:10,0`
`S:20,0`
`S:30,0`
	`~XPOINT!D${Dst 1};S${SAT 1}\`

Switched Logical Destination 1 to Logical Source 2 (Name is "SAT 2")	

`S:00,1`
`S:10,1`
`S:20,1`
`S:30,1`
	`~XPOINT!D${Dst 1};S${SAT 2}\`

Switched Logical Destination 1 HD signal to "SAT 1" (breakaway)	

`S:00,0`	`~XPOINT!D${Dst 1};S${}\`
	`~XPOINT!D${Dst 1.HD};S${SAT 1.HD}\`
	`~XPOINT!D${Dst 1.SD};S${SAT 2.SD}\`
	`~XPOINT!D${Dst 1.AES};S${SAT 2.AES}\`
	`~XPOINT!D${Dst 1.Video};S${SAT 2.Video}\`

Switched Logical Destination 1 HD signal to "SAT 2" (restores follow of "SAT 2")	
`S:00,1`	`~XPOINT!D${Dst 1};S${SAT 2}\`
Switched Logical Destination 1 HD signal to "SAT 1" (breakaway)	

`S:00,0`	`~XPOINT!D${Dst 1};S${}\`
	`~XPOINT!D${Dst 1.HD};S${SAT 1.HD}\`
	`~XPOINT!D${Dst 1.SD};S${SAT 2.SD}\`
	`~XPOINT!D${Dst 1.AES};S${SAT 2.AES}\`
	`~XPOINT!D${Dst 1.Video};S${SAT 2.Video}\`

Switched Logical Destination 1 HD signal to "SAT 3" (breakaway)	
`S:00,2`	`~XPOINT!D${Dst 1.HD};S${SAT 3.HD}\`
Switched Logical Destination 1 to Logical Source 3 ("SAT 3")	

`S:00,2`
`S:10,2`
`S:20,2`
`S:30,2`
	`~XPOINT!D${Dst 1};S${SAT 3}\`

Note: For proper LRC operation and status reporting, the routing system's logical database must be configured properly. For more information on configuring the logical router database, see the CCS Navigator User Manual, Volume 6: Routing Components, or the RouterMapper Configuration Utility Reference Guide.

Configuring Platinum to use LRC

A valid system database configuration is required to use LRC. System databases may be created using Navigator Database Editor or RouterMAPPER. The basic steps require are:

- Configure Router Matrices
- Define the Logical Database Channels (Levels), Destinations and Sources
- Send the Database configuration to the Routing System

For a more detailed description of the configuration steps required to create and send a system database configuration to the target frames using either RouterMapper or the Navigator Logical Database editor, please refer to the documentation for those products.

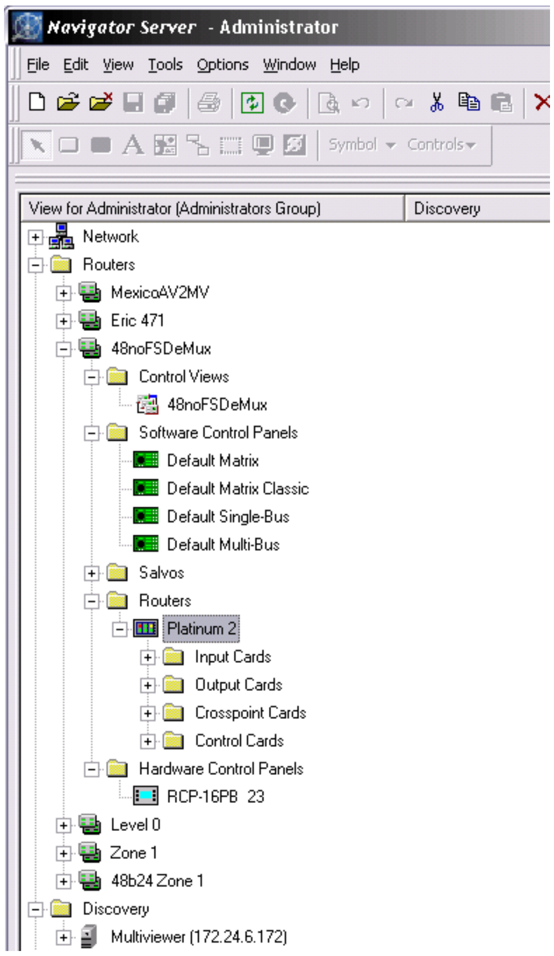
Enabling the Logical Router Protocol

Beginning with Protocol release version 1.1 (included in Platinum PT-RES firmware v 4.2 and later) the LRC Protocol is enabled by default on factory configured Platinum Frames. If you are upgrading from a previous version of Platinum firmware, you will need to enable the Logical Router Protocol server using the Platinum protocol configuration window available in either RouterMAPPER or Navigator.

The following guide illustrates this using Navigator:

Step 1

Open the Platinum frame's configuration window by double clicking the Platinum frame icon in the Navigator navigation pane.

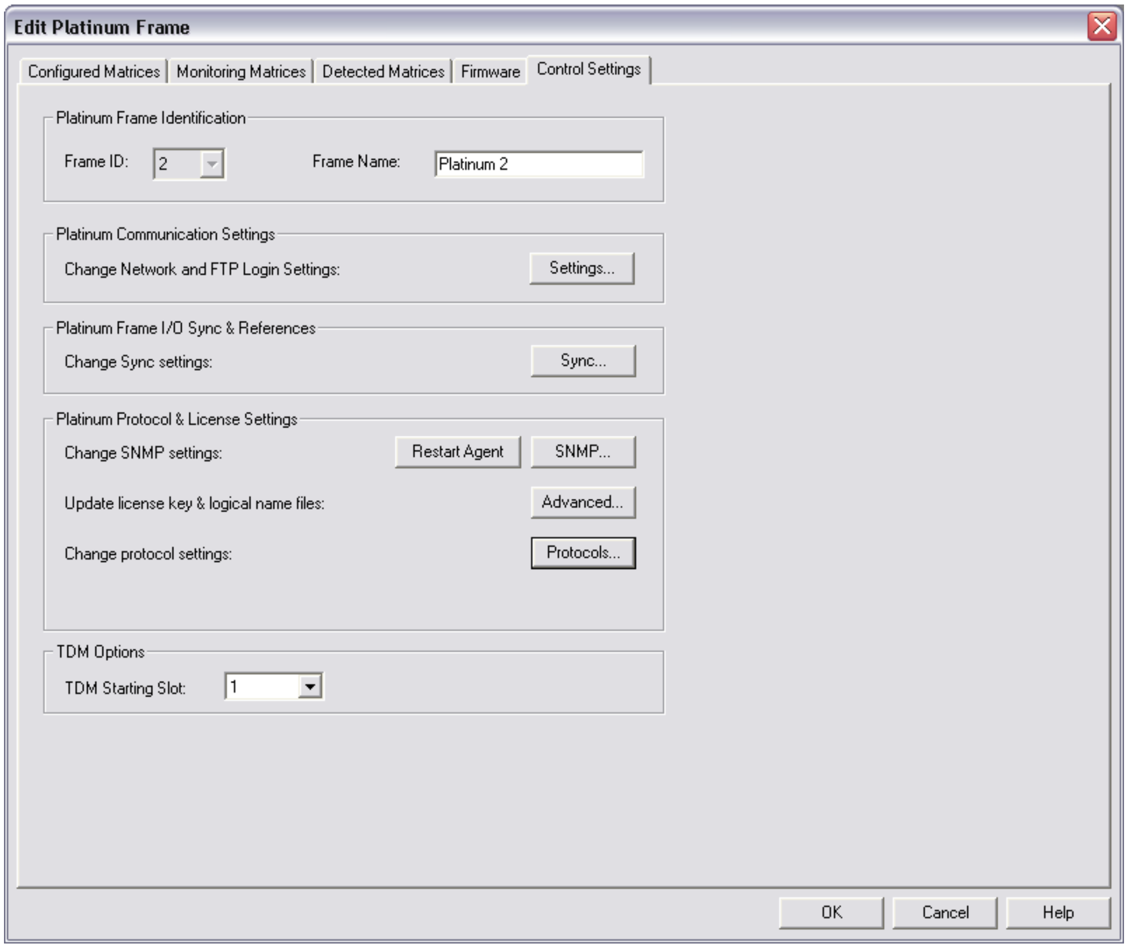


Step 2

The Platinum frame configuration dialog will be displayed.

Select the **Control Settings** tab.

Select the **Protocols** command button.



Platinum Frame Identification

Frame ID: 2   Frame Name: Platinum 2

Platinum Communication Settings

Change Network and FTP Login Settings:

Platinum Frame I/O Sync & References

Change Sync settings:

Platinum Protocol & License Settings

Change SNMP settings:

Update license key & logical name files:

Change protocol settings:

TDM Options

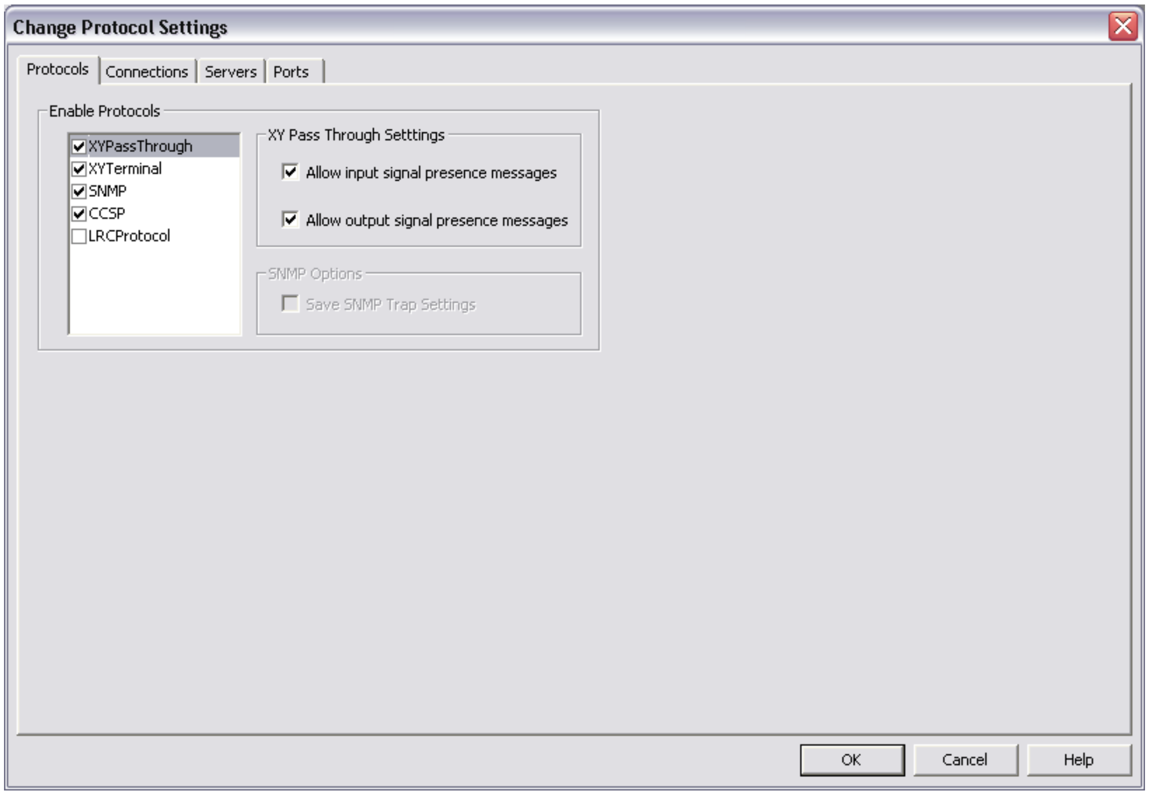
TDM Starting Slot: 1

Figure 1: Selecting the Platinum Control Settings Tab

Step 3

- The Platinum protocol settings configuration dialog will be displayed.
- Select the **Protocols** tab. Ensure the **LRC Protocol** option is selected.

- Then select the **Servers** tab . The Platinum Protocol Server Configuration dialog opens.


Change Protocol Settings

Protocols | Connections | Servers | Ports

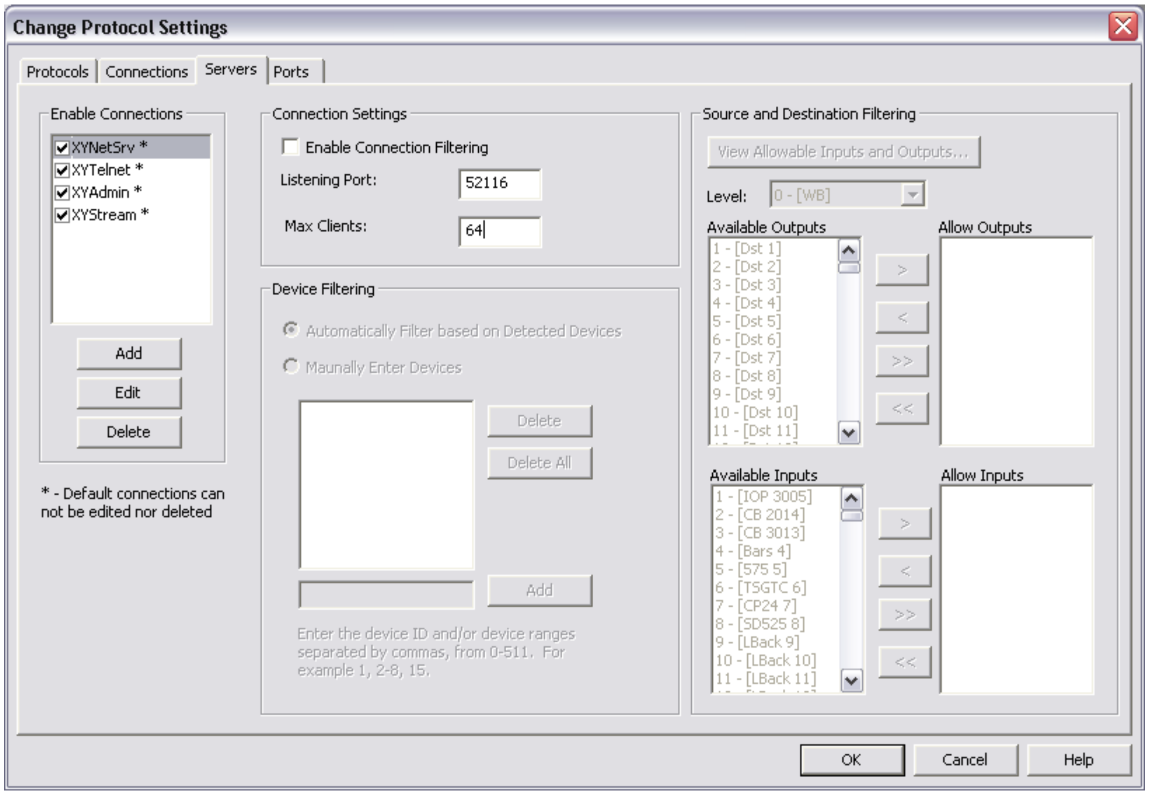
Enable Protocols
- [✓] XYPassThrough
- [✓] XYTerminal
- [✓] SNMP
- [✓] CCSP
- [ ] LRCProtocol

XY Pass Through Setttings
- [✓] Allow input signal presence messages
- [✓] Allow output signal presence messages

SNMP Options
- [ ] Save SNMP Trap Settings

OK | Cancel | Help

Figure 2: Platinum Protocol Settings Dialog


Change Protocol Settings

Protocols | Connections | Servers | Ports

Enable Connections
- [✓] XYNetSrv *
- [✓] XYTelnet *
- [✓] XYAdmin *
- [✓] XYStream *

Add | Edit | Delete

* - Default connections can not be edited nor deleted

Connection Settings
- [ ] Enable Connection Filtering

Listening Port: 52116
Max Clients: 64

Device Filtering
- (●) Automatically Filter based on Detected Devices
- ( ) Maunally Enter Devices

Delete | Delete All | Add

Enter the device ID and/or device ranges separated by commas, from 0-511. For example 1, 2-8, 15.

Source and Destination Filtering

View Allowable Inputs and Outputs...

Level: 0 - [WB]

Available Outputs: 1 - [Dst 1], 2 - [Dst 2], 3 - [Dst 3], 4 - [Dst 4], 5 - [Dst 5], 6 - [Dst 6], 7 - [Dst 7], 8 - [Dst 8], 9 - [Dst 9], 10 - [Dst 10], 11 - [Dst 11]

Allow Outputs

Available Inputs: 1 - [IOP 3005], 2 - [CB 2014], 3 - [CB 3013], 4 - [Bars 4], 5 - [575 5], 6 - [TSGTC 6], 7 - [CP24 7], 8 - [SD525 8], 9 - [LBack 9], 10 - [LBack 10], 11 - [LBack 11]

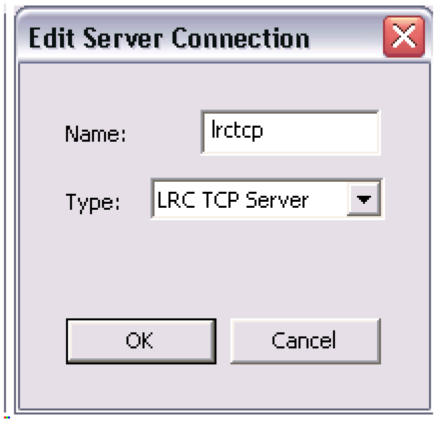
Allow Inputs

OK | Cancel | Help

Figure 3: Platinum Protocol Server Configuration Dialog

Step 4

- If the Server **Enable Connections** window contains an **LRC** connection already, ensure it is checked (enabled) and proceed to Step 6.
- If there is no LRC Connection selected, then click the **Add** button. The **Add New Server Connection** dialog box opens.


Edit Server Connection

Name: lrctcp

Type: LRC TCP Server

Figure 4: Protocol Server 'Add' dialog box

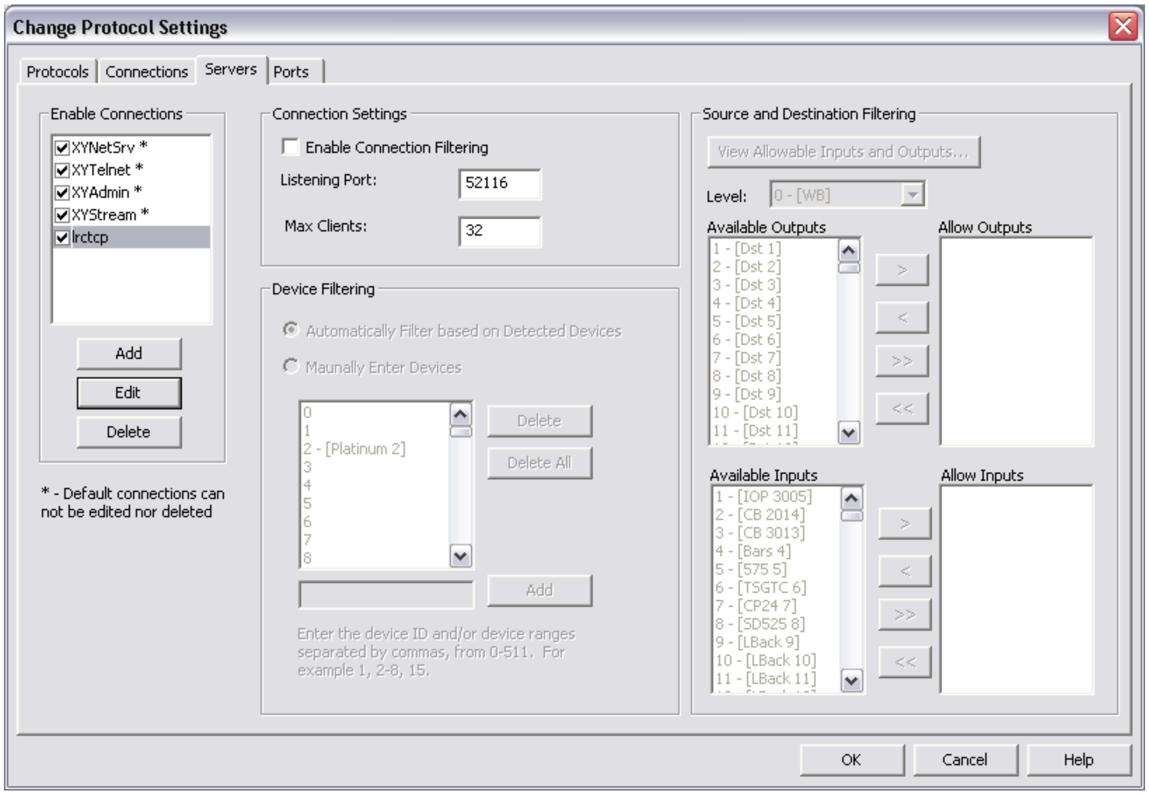
Step 5

1. Using the **Type** drop list control, change the server type to 'LRC TCP Server.
2. Give the Server a name (e.g. 'LRCTCP').
3. Select **OK** to complete the Server addition.

Step 6

- Verify the server is enabled (See 'Enable Connections' field).

- Adjust the maximum number of clients required and the correct LRC port number (32 and 52116 "LRC default port").


Change Protocol Settings

Protocols | Connections | Servers | Ports

Enable Connections
- [✓] XYNetSrv *
- [✓] XYTelnet *
- [✓] XYAdmin *
- [✓] XYStream *
- [✓] lrctcp

* - Default connections can not be edited nor deleted

Connection Settings
- [ ] Enable Connection Filtering
- Listening Port: 52116
- Max Clients: 32

Device Filtering
- (●) Automatically Filter based on Detected Devices
- ( ) Maunally Enter Devices

Enter the device ID and/or device ranges separated by commas, from 0-511. For example 1, 2-8, 15.

Source and Destination Filtering
- Level: 0 - [WB]

Figure 5: Platinum Server Configuration Tab after LRC Server Added

Step 7

1. Select **OK**.
2. When prompted, confirm you wish to send the new configuration to the Platinum frame.

When the send operation completes successfully, the LRC protocol will be active and ready to use.