

# Lynx Data Port Protocol Specification

Copyright © Lynx System Developers, Inc.

## Document History:

- 2019/12/26 – Version 1 initial release.
- 2021/01/14 – Add Image Parameters reverse orientation flag.
- 2021/07/20 – Add Event Status packets.
- 2021/10/10 – Expand Event Status packets.
- 2022/09/28 – Add Image Parameters reset image by time.

The Lynx Data Port (LDP) allows remote applications to access certain information contained in FinishLynx events. FinishLynx, by default, listens for a single incoming TCP connection on port 41601. Each port that FinishLynx listens on can handle only one connection at a time; if an additional connection is made to a port the previous connection is dropped. Additional ports can be created if needed and must use unique TCP port numbers (e.g., 41602, 41603, etc.).

When a FinishLynx event is attached to the LDP module each LDP port is assigned its own camera within the event. For example, if LDP ports 41601 and 41602 exist and an event with two cameras is created then port 41601 will be assigned to the first camera and port 41602 will be assigned to the second camera. If there are more LDP ports than cameras the remaining ports will have no camera assigned, but will still have access to general event information.

## LDP Protocol Fields

The LDP Protocol is binary (not human readable) in order to be as efficient as possible for both FinishLynx and remote applications. Protocol fields consist of various length integers and text strings. Multi-byte integers are always sent “little endian” (least significant byte first). The integers are:

int8 and uint8:	Signed and unsigned 1 byte integers.
int16 and uint16:	Signed and unsigned 2 byte integers.
int32 and uint32:	Signed and unsigned 4 byte integers.
int64 and uint64:	Signed and unsigned 8 byte integers.

Text strings are sent as a uint16 length followed by the specified number of uint16 Unicode characters. An empty string is sent as 2 bytes (a uint16 0 to indicate a zero length string).

## LDP Protocol Format

<packet>	:= <marker><length><type><options><payload>
<marker>	:= uint32: value of 1F9B32F5h (to indicate start of packet)
<length>	:= uint32: packet length, including <marker> (minimum valid value is 12)
<type>	:= uint16: packet type
<options>	:= uint16: packet options (currently unused – always set to 0)
<payload>	:= Zero or more bytes (length and format vary by <type> value)

## LDP Packet Types

### Protocol Version Request – Sent by Remote Application (RA)

<type> := 1  
<payload> := <version><app>  
<version> := uint16: requested protocol version (minimum valid value is 1)  
<app> := string: remote application identifier (optional)

### Protocol Version Reply – Sent by FinishLynx (FL)

<type> := 2  
<payload> := <version><app>  
<version> := uint16: current protocol version (minimum valid value is 1)  
<app> := string: FinishLynx name and version (e.g. "FinishLynx 10.13b01")  
The reply version may be higher or lower than the request version.

### Event Status Request – RA

<type> := 11  
<payload> := none

### Event Status Reply – FL

<type> := 12  
<payload> := <flags><buffer><frame><frames><rate>  
<flags> := uint16: add values to combine options  
1 – event valid                      2 – start valid                      4 – image valid  
8 – reverse orientation    16 – camera offline  
32/64 – external sync status (0=off 1=waiting 2=ready 3=syncd)  
128 – external sync camera  
<buffer> := uint16: camera buffer percentage use  
<frame> := int32: last frame sent (value between -1 and <frames>-1)  
<frames> := int32: number of frames received from the camera  
<rate> := int32: camera frame rate

### Event Info Request – RA

<type> := 3  
<payload> := none

### Event Info Reply – FL

<type> := 4  
<payload> := <filename><enum><rnum><hnum><ename><capture><camera>  
<filename> := string: event filename  
<enum> := string: event number  
<rnum> := string: round number  
<hnum> := string: heat number  
<ename> := string: event name  
<capture> := string: capture location (usually "Finish")  
<camera> := string: camera name

#### Start Info Request – RA

<type> := 5  
<payload> := none

#### Start Info Reply – FL

<type> := 6  
<payload> := <time>  
<time> := int64: start time in microseconds (0 if not available)  
Note that start and image time values will exceed 86,400,000,000 if camera(s) were booted prior to the current day.

#### Image Parameters Request – RA

<type> := 7  
<payload> := <flags><format><pskip><fskip>[<time>]  
<flags> := uint16: add values to combine options (default 2)  
1 – send image as it becomes available  
2 – reset image (send from the beginning)  
4 – reverse orientation (Reply only; ignored in Request packet)  
8 – reset image to <time> below (Request only; not in Reply)  
<format> := uint16: pixel format (default 3)  
1 – pixels are 8 bit grayscale (uint8)  
2 – pixels are 15 bit color (uint16 0rrrrrggggbbbbb)  
3 – pixels are 24 bit color (three uint8 in BGR order)  
4 – pixels are 24 bit color (uint32 00000000rrrrrrrggggggggbbbbb)  
<pskip> := uint16: pixel skip value (default 0 – send all pixels)  
<fskip> := uint16: frame skip value (default 0 – send all frames)  
[<time>] := int64: reset image time value

#### Image Parameters Reply – FL

<type> := 8  
<payload> := <flags><format><pskip><fskip>  
<flags> := uint16: add values to combine options (see above)  
<format> := uint16: pixel format (see above)  
<pskip> := uint16: pixel skip value (see above)  
<fskip> := uint16: frame skip value (see above)

#### Image Frame Request – RA

<type> := 9  
<payload> := none

When FinishLynx receives this packet it sends the next available image frame. If FinishLynx is currently not sending image as it becomes available (Image Parameters <flags> value 1 not set) then this packet can be used to “single step” through the image.

#### Image Frame Reply – FL

<type> := 10  
<payload> := <time><format><pskip><fskip><numpix><pixels>  
<time> := int64: frame time in microseconds (0 if not available)  
<format> := uint16: pixel format (see above)  
<pskip> := uint16: pixel skip value (see above)  
<fskip> := uint16: frame skip value (see above)  
<numpix> := uint16: number of pixels  
<pixels> := image pixels in top to bottom order (format varies, see above)

## LDP Sample Packets

### Protocol Version Request:

F5 32 9B 1F 10 00 00 00 01 00 00 00 01 00 00 00

### Protocol Version Reply:

F5 32 9B 1F 36 00 00 00 02 00 00 00 01 00 13 00 46 00 69 00 6E 00  
69 00 73 00 68 00 4C 00 79 00 6E 00 78 00 20 00 31 00 30 00 2E 00  
31 00 33 00 62 00 30 00 31 00