

LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Document Type: LIGO-T980024-00 - C 01/09/02
Data Acquisition Daemon (DAQD) Client-Server Communication Protocol Version 11
Alex Ivanov

This is an internal working note
of the LIGO Project.

California Institute of Technology
LIGO Project - MS 51-33
Pasadena CA 91125
Phone (818) 395-2129
Fax (818) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project - MS 20B-145
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

WWW: <http://www.ligo.caltech.edu/>

LIGO DRAFT

1 ABSTRACT

This is a description of the communication protocol used by the Data Acquisition Daemon (DAQD) and the client program. Reliable delivery byte stream socket connection (TCP/IP) is used to send commands and deliver the data.

2 KEYWORDS

2.1. Control Connection

Stream socket connection established by the client and used to send client commands and to receive server responses.

2.2. Data Connection

The connection that is used by DAQD to transmit data (either full channels or trend channels) to the client. Could be either separate from or the same as the control connection.

2.3. DAQD

Data acquisition daemon. Multi-threaded UNIX program that does a number of tasks, including channel trend calculation and client data request processing, creating frame files on disk as new data is acquired.

2.4. Full Channel

Data signal channel (no matter what speed, fast or slow), transmitted as it was sampled on a data collection unit.

2.5. Trend Channel

Data channel that represents either of five values (min, max, RMS, mean, n) calculated on the full channel over one second. Minute trend is calculated over one minute. Sampling rate for any trend channel is 1 Hz, minute trend -- 1/60 Hz. Every full channel can have either five trend channels or none. This depends on DAQD configuration and it is specified on the server startup. Whether a trend is being calculated for the particular full channel or not can be determined with 'status channels' command.

3 OVERVIEW

There are two parts to the data communication protocol of DAQD. First part is the protocol of the client requests and the server responses to these requests. This is defined in the following section. Second part is about the definition of the format of the data that is sent by the server responding to client's request. This data is either full channel data (possibly decimated down to the required sampling rate) or trend channel data. Data transmission format is specified in the section five.

Protocol allows requests and data transmissions of the on-line or current data, that is arriving each second, or the off-line data, read from the filesystem storage. Each data block carries timestamp in GPS seconds and nanosecond residual and block length in seconds.

4 CLIENT REQUEST FORMAT

4.1. Sample requests

- Receive all channels. To understand what channels are configured you will use `status channels` command.
start net-writer "127.0.0.1:8090" all;
- Receive one channel, specified by name.
start net-writer "204.152.167.20:8090" { "IFO_LOCK" };
- Receive two channels.
start net-writer "127.0.0.1:8090" { "IFO_LOCK" "PEM_Seis_4" };
- Optional data rate can be given for every channel.
start net-writer "127.0.0.1:8090" { "IFO_LOCK" 32 "PEM_Seis_4" 16 };
- IP address is optional -- DAQD will connect to the client's IP address.
start net-writer "8090" all;
- Port number is optional too -- DAQD will use control connection to send the data.
start net-writer all;
- Receive last 15 seconds of data for all configured channels.
start net-writer "127.0.0.1:8090" 15 all;
- Receive data for all channels for time period given in GPS time (570219219 seconds) and the period (3600 seconds here):
start net-writer "127.0.0.1:8090" 570219219 3600 all;
- Receive trend data on-line for one channel on port 9999.
start trend net-writer "9999" { "IFO_DMRO.min" "IFO_DMRO.max" "IFO_DMRO.rms" };
- Receive minute trend for a certain time period:
start trend 60 net-writer 570219219 3600 {"IFO_DMRO.rms"};
- Receive data at 16 Hz rate (ie. 16 transmissions per second):
start fast-writer {"IFO_DMRO"};
- Receive data *frames* for time period:
start frame-writer 570219219 2 all;
- Receive data *frames* on-line:
start frame-writer all;
start frame-writer {"IFO_DMRO"};

LIGO-DRAFT

- Receive a name of the latest full frame file created:
start name-writer all;
- Stop network writer (this will disconnect data connection from the DAQD side).
kill net-writer 123123;
- Determine communication protocol version. Increased if there are significant changes in the communication protocol.
version;
- Get communication protocol revision. Increased if there are additions to the protocol. Protocols with different revision numbers and the same version number are upward compatible.
revision;
- Get the number of configured channels, channel names, rates and channel group numbers.
status channels;
- Get the number of channel groups, channel group numbers and names.
status channel-groups;
- Close control connection.
quit;

As you can see from the examples, various forms of data request command all start with ‘start *-writer’ keywords. Absolutely any command ends with semicolon. Following optional parameter is the TCP/IP address ((IP address/hostname, TCP port) tuple) given in the format, where IP address is separated from the TCP port by colon. Enclose the address in double quotes, this makes it a token of string type. Semantically this address is the destination where requested data will be sent. Server establishes a connection to that address and sends data. After all data is sent server disconnects. If IP address is omitted, server will determine it from the clients request. If whole address parameter is omitted, server will not be establishing new connection. It will use the connection, on which it received the request (control connection), instead, to send the data. Data will follow server’s positive response to client’s request.

After the optional address parameter there could be up to two time arguments. First is the timestamp, which marks the first second of the requested data (in GPS seconds). Second time argument is the number of seconds (time period) for which the data is requested. The timestamp can be omitted; time period argument then indicates how many seconds of data up to current time is requested. If both time arguments are omitted, then the data is sent on-line, as soon as it arrives, continuously until either client or server disconnects.

The last block of parameters in the net-writer start request is the channel names and sampling rates definition. The simplest form of this is ‘all’, which instructs DAQD to send data for all configured data channels. Client should use ‘status channels’ command to get channel names and rates, so it can parse the data it receives from the server.

Channels could be specified by name, as seen from the examples; optional sampling rate (in Hertz) can be specified after the channel name. Rate can be different for different channels in the configuration set, could also be omitted for some of the channels (or for all channels). Rate must lie between 1 and the channel actual rate and must be a power of two.

Default downsampling method is averaging. Keyword ‘nofilter’ can be used to request decimation. Currently NDS does not support filtering.

4.2. Request to get on-line data.

The ‘start net-writer’ command with no time arguments will send the data on-line or continuously until either you or the DAQD disconnects. DAQD wouldn’t normally disconnect unless something happens (like it is being shutdown or similar).

‘start’ ‘net-writer’ optional_address channel_names;

4.3. Request to get off-line data

If you specify time arguments (one or both of them) you will be getting the data for specific time period. There are two forms of this command. First form, with the only one time argument, sends the data for last number of seconds. Another form has two time arguments that specify the first second to be sent and the length of the transmission in seconds.

‘start’ ‘net-writer’ optional_address period_seconds channel_names;

‘start’ ‘net-writer’ optional_address gps_start_time period_seconds channel_names;

4.4. Server response

Positive response always starts with ‘0000’, i.e. it is literally four ASCII zero digit characters. After that server sends net-writer ID. Net writer ID is unsigned long integer, which is sent in ASCII as eight hexadecimal digits. For instance, ‘0023aef3’. Client must read this ID and can keep it to be able to use in ‘kill net-writer’ command.

Negative response is always four bytes -- any four ASCII hexadecimal digit combination except ‘0000’. It is an error code. Error codes are defined in ‘daqc.h’ header file. See Appendix 3 “error codes” on page 12.

4.5. Status Channels command

‘status channels’ command is used to determine how many full channels there are configured, channel names, rates, whether trend is calculated for a channel or not, channel group numbers and data types. Server will send all that data on the control connection. Format is specified in the Appendix 2 server transmission grammar.

status channels;

4.6. Status Channel Groups command

‘status channel-groups’ command is used to get the list of channel group numbers and channel group names. Channel group number should be matched with the corresponding number in the data received with ‘status channel’ command to get channel group name for certain channel.

status channel-groups;

4.7. Version command

Can be used to check communication protocol version to avoid situations when old client is run against the newer DAQD server and the new server has significant changes in the communication protocol. Server will send version number on the control connection right after the positive response ('0000'). Version number transmission is represented as four ASCII hexadecimal digits.

version;

4.8. Revision command

Minor version number. Transmission format as in 'version' command.

revision;

4.9. Kill net-writer command

This command is used to close data connection. This will not affect control connection, so if the control and data connections are the same stream socket connection, sending 'kill net-writer' command will not disrupt the actual socket connection, but will stop any data transmission from the DAQD for the specified net-writer ID. 'kill net-writer' causes DAQD to disconnect the socket if data connection is separate from the control connection

kill net-writer <ID>;

4.10. Quit command

Used to close control connection from the server side. Server will just close the socket, no response to 'quit' command is sent.

quit;

4.11. Trend data request

Trend data request is syntactically similar to the full channel data request, except that there is 'trend' keyword used to indicate that the trend data is needed:

start trend net-writer all;

start trend net-writer {"IFO_DMRO.min" "IFO_DMRO.max" "IFO_DMRO.rms"};

Trend channel names are constructed by sticking one of the five predefined suffixes to the full channel name. Suffixes are:

- '.min': Minimal sample over the period of one second
- '.max': Maximal sample over the period of one second
- '.rms': Root Mean Square of samples over the period of one second.
- '.mean': Mean value over one second.
- '.n': The number of data points used to calculate above values.

Client should check if the trend is calculated for the channel, looking at the ‘status channels’ command data Trend Flag for each channel. If this flag is not set, then the trend is not calculated for the channel and your ‘start trend net-writer’ request will fail.

You shouldn’t put data rate in ‘start trend net-writer’ command.

4.12. Minute long trend request

start trend 60 net-writer 570219219 3600 {“IFO_DMRO.min” “IFO_DMRO.max” “IFO_DMRO.rms”};

Trend can be requested for a duration of time longer than one second. This was done to give support for requests to get the data trend for significantly long periods of time, days or weeks versus seconds and minutes.

On-line data request is not supported

Presently, as of version 11 of the protocol, only 60 seconds (minute long) trend is supported and minute long trend over the last hour is not available.

4.13. 16Hz data feed

start fast-writer {“IFO_DMRO”};

This starts fast network writer, which sends the data to the client 16 times per second. New data is sent to the client as soon as it becomes available.

Fast network writer is not supported for offline transmissions. Request to get all channels is not supported.

4.14. Request to get filenames

Request to get a name of the last created full frame file.

‘start’ ‘name-writer’ optional_address ‘all’;

Data is sent on-line. New block is sent as soon as new frame file is created by the server. There is one filename per transmission block. Filename is a zero terminated string.

4.15. Request to get frames

Request to get data in frame format, described in LIGO-T970130-05-E, Specification of a Common Data Frame Format for Interferometric Gravitational Wave Detectors.

Request to get frames for the last number of seconds is not supported.

Only ‘all’ channels can be requested in off-line transmission.

Request format supported:

‘start’ ‘frame-writer’ optional_address gps_start_time period_seconds ‘all’;

‘start’ ‘frame-writer’ optional_address channel_names;

‘start’ ‘frame-writer’ optional_address ‘all’;

Server replies with positive ACK and sends net-writer ID on the control connection. Server sends data on the data connection in the format specified on page 7. Following block header, frame data is sent (field **Data** on Figure 1: Data Transmission). Frame data is a frame file read from disk. Frame file has one frame one second long. Timestamp in the block header has only **GPS** field valid, **Nsec** field is not set. Valid timestamp can be retrieved from the frame data. Frame data length equals **Blen** - 16.

4.16. Request to get current GPS time

The following command can be used to determine current time:

‘gps’;

Server replies with positive ACK and then sends one data block header with GPS and Nsec fields set to the current DAQ system time.

4.17. Frame file availability request

This command is used to determine how many seconds of full frame data is available on disk.

‘status’ ‘main’ ‘filesys’;

Server replies with positive ACK, then sends one data comm block which has the number of seconds of data available present in the GPS field.

5 SERVER DATA TRANSMISSION FORMAT

Channel data transmission is sent by DAQD in the response to a ‘start net-writer’ command. Data is sent on the data connection. Control connection is used to send command to DAQD and receive response. Data connection and control connection could be the same network connection. In that case DAQD data transmission will follow positive server response defined by the control connection communication protocol, See 4.4. “Server response” on page 4..

Data is transmitted in blocks. Each data block is preceded with the block header. Generic format of the data transmission is shown in Figure 1: Data Transmission. All data is in binary form. Protocol components are defined as:

- **Offline:** Zero if data is sent on-line, one if off-line.
- **Blen:** Block length. Size of the block in bytes. Does not include size of Blen.
- **Secs:** Data length in seconds. Reflects file size that DAQD is using to save the data. There could be blocks of different size in one transmission. Server is free to choose block size as it sees fit for the optimal processing work load and data transmission efficiency.
- **GPS:** Timestamp of the first data sample in the block, measured in seconds since GPS time origin.
- **Nsec:** Nanosecond residual of **GPS** timestamp.
- **SeqNum:** Block sequence number. This number is increased for transmitted and dropped

- blocks. Can be used to check if any data blocks were dropped. Sequence starts from zero.
- **Data:** channel (full or trend) data, as requested by ‘start net-writer’ command. The order of the channels is the same as in the channel definitions section of ‘start net-writer’ command. So, for each channel defined in the ‘start net-writer’ command there will be Rate * Secs bytes of data. ‘Rate’ equals to the required sampling rate, if specified for the data channel. If rate was not specified, channel sampling rate is used, as configured in DAQD and accessible via ‘status channels’ command. Data samples are grouped by channel.

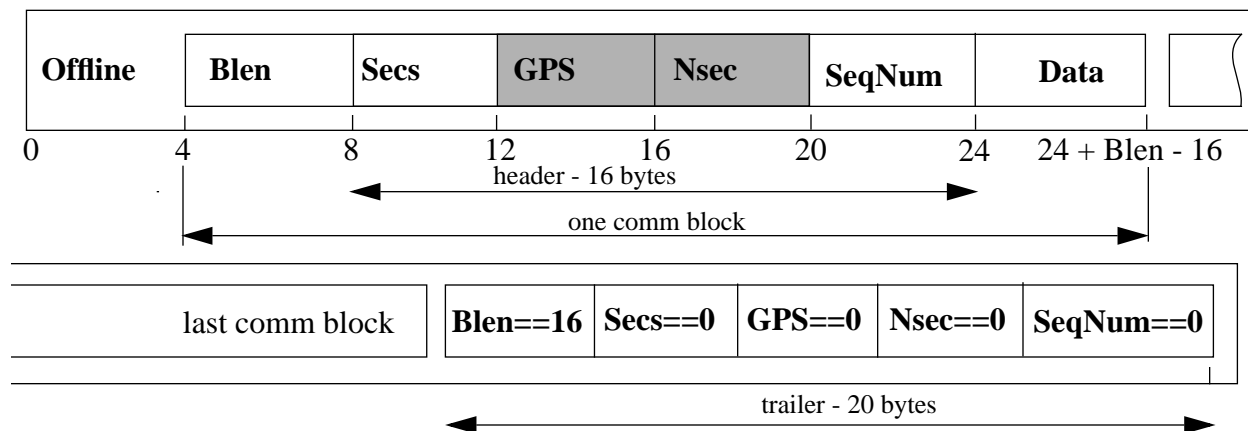


Figure 1: Data Transmission

5.1. Trailer

Transmission is terminated with trailer block, which is zero length data block.

5.2. Full channel data

Full channel data is requested by ‘start net-writer’ command.

There is certain number of samples sent for each requested channel. Channels are requested in the channel_names section of the ‘start net-writer’ command. This could be either ‘all’ or the channel names and, possibly, rates enclosed in curly braces:

start net-writer all;

start net-writer {“IFO_DMRO” “IFO_CMRO” 16};

Figure 2: Two Forms of ‘start net-writer’ Command

For ‘all’ channels, channel order and sample rate of the data is the same as in the response to ‘status channels’ command (or as configure in DAQD). For the second form, where channel names and rates are specified, channel order and sample rate in the data transmission is exactly as specified by the command. For instance, for the second command in Figure 2: Two Forms of ‘start net-writer’ Command, length of ‘Data’ would be 16K + 16 samples and there would be 16K samples sent first for IFO_DMRO channel (assuming that it is configured at 16 KHz) and after that there would be 16 samples of data sent for IFO_CMRO channel.

Now, in the example above IFO_CMRO channels is the fast data channel, meaning that its rate is 16 KHz. In this case DAQD does averaging. For the case above, it will average each 1024 samples.

5.3. Trend channel data

Trend channel data is requested by ‘start trend net-writer’ command.

Again, the principle for the understanding the ‘**Data**’ format is the same as for the full channel data. You shouldn’t really specify sampling rate in the ‘start trend net-writer’ command ever, since it can’t be anything else than one. ‘.min’ and ‘.max’ values are 32bit signed integers, floats or doubles (depending on full channel data type). ‘.rms’ and ‘.mean’ channels are 64bit double precision IEEE 754 floating point value. ‘n’ channels is 32 bit unsigned integer.

APPENDIX 1 CLIENT REQUEST GRAMMAR

Terminals are in " and <>

program: /* Nothing */

| commands

commands: command ';' commands

command: net_writer | name_writer | fast | trend | frame | online_frame | stop | status_commands

status_commands: 'version'

| 'revision'

| 'status' 'channels'

| 'status' 'channel-groups'

net_writer: 'start' 'net-writer' optional_address optional_times channel_names

name_writer: 'start' 'name-writer' optional_address 'all'

fast: 'start' 'fast-writer' optional_address optional_times channel_names

trend: 'start' 'trend' optional_trend_period 'net-writer' optional_address optional_times channel_names

frame: 'start' 'frame-writer' optional_address gps_start_time period_seconds 'all'

online_frame: 'start' 'frame-writer' optional_address channel_names

optional_trend_period: /* Nothing */

| <unsigned long integer>

optional_times: /* Nothing */

| gps_start_time period_seconds

| period_seconds

gps_start_time: <timestamp given in seconds measured from GPS time origin>

period_seconds: <unsigned long integer; the number of seconds>

optional_address: /* Nothing */

| "IP-address:port number" // "127.0.0.1:9090" for instance

| "IP-address" // for example "127.0.0.1"

| "port number" // for example "9090"

LIGO-DRAFT

```

channel_names: 'all'
    | '{ distinct_channel_names }'
distinct_channel_names: /* Nothing */
    | channel_name optional_rate optional_decimate distinct_channel_names
optional_rate: /* Nothing */
    | <unsigned long integer, should be less than or equal to the corresponding channel rate and must be power
of two>
optional_decimate: /* Nothing */
    | "nofilter"
    | "average" // default
channel_name: <string in double quotes, for example "IFO_Lock">
stop: 'kill' 'net-writer' net_writer_id
net_writer_id: <unsigned long decimal integer number sent by the DAQD in response to 'start' command>

```

APPENDIX 2 SERVER TRANSMISSION GRAMMAR

This defines datastream from DAQD to the client program supported by DAQD's net-writer ('start net_writer' and 'start trend net-writer' commands), gps command.

```

=====
'long(0)' means 'long' token that has value '0'
'byte{N}' means 'N' bytes, where 'N' is an integer expression
'long' and 'byte' are terminals (in fact, terminal variables here)
---
```

```

stream: offline blocks trailer
offline: long(0) // on-line data feed
    | long(1) // off-line data transmission
blocks: /* Nothing */
    | block blocks
block: length header data
trailer: long(16) long(0) long(0) long(0) long(0)
length: long
header: data_length_secs timestamp seq_num
timestamp: long long
seq_num: long
data: byte(length - 16)
data_length_secs: long
long: <32 bit quantity in network byte order>

```

Following is the spec on the transmission of the channel data

LIGO-DRAFT

as requested by 'status channels' command.

```

=====
stream: num_of_channels ignore blocks conversion_data
conversion_data: gain slope offset units
gain: eight_hex_digits
slope: eight_hex_digits
offset: eight_hex_digits
units: char{40} // zero terminated or padded string
eight_hex_digits: hex_digit{8} // 32 bit float
blocks: /* Nothing */
    | channel_desc blocks
num_of_channels: four_hex_digits
ignore: four_hex_digits // do not use
channel_desc: channel_name data_rate trend_flag channel_group_num bps data_type
channel_name: char{40} // zero terminated or padded string
data_rate: four_hex_digits
trend_flag: four_hex_digits
channel_group_num: four_hex_digits
bps: four_hex_digits
data_type: four_hex_digits
four_hex_digits: hex_digit{4} // unsigned 16 bit integer
hex_digit: '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'|'a'|'b'|'c'|'d'|'e'|'f'
char: <any one byte, usually printable ASCII character>

```

Following is the spec on the transmission of the channel group data
as requested by 'status channel-groups' command.

```

=====
stream: num_of_groups clock_freq blocks
blocks: /* Nothing */
    | group_desc blocks
clock_freq: four_hex_digits
num_of_groups: four_hex_digits
group_desc: channel_group_name channel_group_num
channel_group_name: char{40} // zero terminated or padded string
channel_group_num: four_hex_digits
four_hex_digits: hex_digit{4}
hex_digit: '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'|'a'|'b'|'c'|'d'|'e'|'f'
char: <any one byte, usually printable ASCII character>

```

LIGO-DRAFT

APPENDIX 3 ERROR CODES

Table 1: Communication Errors

Error Code	Description	In Response To
1	Parse error. This is either syntax or lexical error. Check that your request is in the format described in this document. All keywords are case-sensitive, all should be in lower case. You can't ever omit double quotes and should terminate any command with semicolon.	any command
2	DAQD not configured. Indicates problems with DAQD configuration.	any command
3	Invalid IP address. Check if IP address (the one in the address "IP:port" is all right). IP address is specified in '.' notation -- check inet_ntoa(3N) manual page for further description.	start net-writer start trend net-writer
4	Invalid channel name. Check that the name is all right. Channel name is case sensitive. Use 'status channels' to get channel names.	start net-writer start trend net-writer
5	Couldn't create data connection socket.	start net-writer start trend net-writer
6	Couldn't set socket option for data connection socket	start net-writer start trend net-writer
7	Couldn't establish data connection (connect() failed)	start net-writer start trend net-writer
8	DAQD is busy. This could happen if DAQD is overloaded and wouldn't start new net-writer. There is a limit of number of net-writers running at the same time. It depends on DAQD configuration and couldn't be greater than 32.	start net-writer start trend net-writer
9	Server memory exhausted	any command
12	No such net-writer.	kill net-writer
13	Requested data is not found or not available	start net-writer start trend net-writer (off-line data request only)
14	Couldn't get caller's IP address	start net-writer start trend net-writer

Table 1: Communication Errors

Error Code	Description	In Response To
15	Couldn't do dup() on control connection	start net-writer start trend net-writer
16	Invalid channel data rate. Rate must be a power of 2 and must lie between 1 and channel's actual data rate.	start net-writer start trend net-writer
17	Socket shutdown failed	kill net-writer
18	Trend data is not available, meaning server doesn't calculate or allow access to trend data.	start trend net-writer
19	Full channel data is unavailable.	start net-writer
20	No off-line data is available.	start net-writer start trend net-writer start name-writer
21	Request or request parameter is not supported	start trend net-writer

LIGO-DRAFT