1 Introduction

LDMP (Logger Data Monitoring Protocol) is a simple protocol that allows a client to receive text formatted data records through a socket from the *LoggerNet* server. The original data export server application (ldmp_server.exe) required several command line parameters to tell it what *LoggerNet* server to connect to and what table to export. It was written in such a way that the export server process stopped altogether if anything went wrong or if the client connected and then disconnected from its socket. Since the export server could export only one record per table, several server instances had to be started (and kept alive) on the machine in order to monitor multiple tables.

The new *LDMP* server (ldmp_server2.exe) uses the same protocol as the old export server with the exception that it can export records from multiple tables on the same server. It is written to operate much more robustly than its predecessor and can also support multiple clients at once. The new server's command line interface also allows greater control over the data export process than its predecessor allowed.

This new server also extends the LDMP protocol such that the client can optionally specify the tables the should be exported and the conditions under which those tables should be exported.

Finally, *ldmp_server2* can optionally maintain a log of events relating to the *LDMP* service. It provides options to control over how and whether these events are logged to disc files and can also provide a second optional socket service that streams these events to any client that attaches to the service.

This document serves the following purposes:

- Describes the new server configuration options.
- Describes the Logger Data Monitoring Protocol (*LDMP*)
- Describes the *LDMP* event log and the protocol for monitoring these events.

2 Operating the *ldmp_server2*

The *ldmp_server2* application loads its configuration from its command line. There is no required configuration information. If the application is started without options, it will export data in its default mode. Any command line option that is specified will change that default operation.

The *ldmp_server2* application has the following command line EBNF syntax:

```
command-line := {option whitespace {whitespace} }.
option := "--" option-name ("=" | ":") ["{"] option-value ["}"]
```

This is a formal way of saying that the command line consists of zero or more options separated by whitespace characters (space and tab) where each option is specified with the sequence, "`--`" followed by the option name followed in turn by a "=" or ":" followed by the option value. The option value can be surrounded by quotation marks or by a pair of braces ('{' and '}'). Valid option names are as follows:

- “`server-host-name`” (see section 2.3.1)
- “`server-host-port`” (see section 2.3.2)
- “`logon-name`” (see section 2.3.3)
- “`logon-password`” (see section 2.3.4)
- “`service-port`” (see section 2.3.5)
- “`config-file-name`” (see section 2.3.6)
- “`table-names`” (see section 2.3.7)
- “`order-option`” (see section 2.3.8)
- “`start-option`” (see section 2.3.9)
- “`backfill-seconds`” (see section 2.3.12)
- “`start-time`” (see section 2.3.10)
- “`client-specified`” (see section 2.3.13)
- “`close-after-disconnect`” (see section 2.3.14)
- “`events-to-disc`” (see section 2.3.15)
- “`event-log-path`” (see section 2.3.16)
- “`event-log-file`” (see section 2.3.17)
- “`event-log-size`” (see section 2.3.18)
- “`event-log-count`” (see section 2.3.19)
- “`export-events`” (see section 2.3.20)
- “`event-export-port`” (see section 2.3.21)
- “`output-format`” (see section 2.3.23)
- “`run-as-daemon`” (see section 2.3.24) (Linux Only)

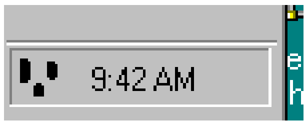


Figure 1: The *ldmp_server2* icon in the system tray

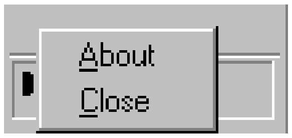
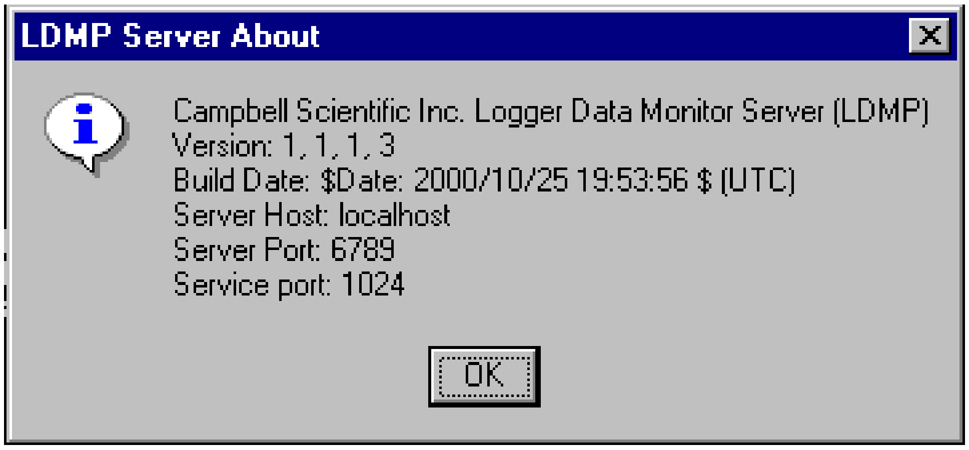


Figure 2: The application menu



Campbell Scientific Inc. Logger Data Monitor Server (LDMP)
Version: 1, 1, 1, 3
Build Date: $Date: 2000/10/25 19:53:56 $ (UTC)
Server Host: localhost
Server Port: 6789
Service port: 1024

Figure 3: *ldmp_server2* About Box

2.1 Controlling *ldmp_server2* Under Windows

If the *ldmp_server2* process is able to initialise properly from its command line arguments, its icon will be placed in the computer system tray as shown in figure 1. This is the only indication that the *ldmp_server2* is running.

A menu for the application can be accessed at any time by clicking on the application icon in the system tray with the right mouse button. Doing so will produce results similar to those shown in figure 2.

Choosing "Close" from this menu will result in shutting *ldmp_server2* down. Choosing "About" from this menu will result in a dialogue box as shown in figure 3.

This about box will describe the server version, release date (specified in greenwich mean time), the *LoggerNet* server host name and TCP port, and the TCP port on which *ldmp_server2* is offering its service.

2.2 Controlling *ldmp_server2* Under Linux

When *ldmp_server2* is run under Linux, there is no special user interface that is presented. If run as a normal application, the program will write little or nothing to the console window. It can be terminated using ctrl-C or using the kill command. If *ldmp_server2* is run as a daemon, it will run in a process detached from any console. Its process ID can be determined by using the following command line:

```
ps -A | grep ldmp_server2 | cut -b -6 -
```

These process identifiers can be used as arguments to the kill command to end the process(es).

2.3 *ldmp_server2* Options

2.3.1 `server-host-name` Option

Specifies the IP interface address of the computer that is hosting the desired *LoggerNet* server.

Syntax

```
option := "--server-host-name=" (ip-address | domain-name).
ip-address := number "." number "."number "." number.
domain-name := name-component {"." name-component}.
name-component := { [a..z] | [A..Z] | [0..9] }
```

Default Value `localhost`

Remarks If the *LoggerNet* service is not available at the specified address, the *ldmp_server2* process will close all incoming client connections immediately after they are opened. The LDMP server will attempt to establish a connection with the *LoggerNet* server each time a client tries to connect.

2.3.2 `server-host-port` Option

Specifies the TCP port address of the *LoggerNet* service on the host machine.

Syntax

```
option := "--server-host-port=" port-number.
port-number := digit {digit}. ; 0 < port-number < 65535
```

Default Value 6789

Remarks This option rarely needs to be specified. This option might need to be specified in order to access the *LoggerNet* service through a port-mapping firewall. It also may need to be specified if the *LoggerNet* service is installed on its host machine at a different address other than its default.

2.3.3 `logon-name` Option

Specifies the logon account name that the *ldmp_server2* application should use to logon to the *LoggerNet* service.

Syntax

```
option = "--logon-name=" logon-name.
```

Default Value `ldmp_server2`

Remarks If security is enabled on the *LoggerNet* server, the option value must identify a valid account on the *LoggerNet* server. If the logon process fails, the client socket will be shut down.

If the account name contains white space characters (space, tab, etc.) it will be neccesary to enclose the option value in braces ('{' and '}') or double quotation marks ('"').

2.3.4 `logon-password` Option

Specifies the password that will be used in conjunction with the account name to log on to the *LoggerNet* service.

Syntax

```
option := "--logon-password=" password.
```

Default Value an empty string

Remarks If security is enabled for the *LoggerNet* service, the option value must match the password for the account named by the `logon-name` option. The client socket will be closed immediately if the logon protocol fails.

If the option value contains whitespace characters, it will be necessary to enclose the option value in braces ('{' and '}') or double quotation marks ('"').

2.3.5 `service-port` Option

Specifies the TCP port number that will be used to connect to the *ldmp_server2* service.

Syntax

```
option := "--service-port=" service-port.
service-port := digit {digit}. ; 0 < service-port <= 65535
```

Default Value 1024

Remarks This option should be specified if there is already or should be) a socket on the host computer that is using the default address. This can be determined by running the `netstat -a` command at the command prompt on the host computer.

2.3.6 `config-file-name` Option

Specifies that more options are to be read from the file named by the option value.

Syntax

```
option = "--config-file-name=" file-name.
```

Default Value No configuration file will be read unless this option is specified.

Remarks All of the options that can be specified on the program command line can be declared within the specified file as well.

If the option value contains whitespace characters, it will be necessary to enclose the option value in braces ('{' and '}') or double quotation marks ('"').

2.3.7 `table-names` Option

Specifies the tables that the *ldmp_server2* process should export.

Syntax

```
option = "--table-names={" table-list "}".
table-list = { "{" station-name table-name "}" }.
```

Note that the braces are required around each station-name, table-name pair.

Default Value An empty list (all available tables will be exported).

Remarks This option is used to restrict the list of tables that the *ldmp_server2* process can export. Neither the station name nor the table name have to be defined on the *LoggerNet* service at the time the *ldmp_server2* process is started or when a client connects to the LDMP socket. The LDMP server will use the list of names to evaluate whether to export a table when and if it becomes available.

Example The following example shows how this option can be used to specify that two tables should be exported:

```
ldmp_server2 --table-names={{csi_yard hr1} {csi_yard min5}}
```

2.3.8 `order-option` Option

Specifies the order in which table records will be exported to the client. The options that are available are:

`collected`	Records will be emitted in the order that the *LoggerNet* server collected them from the datalogger.
`logged-with-holes`	This specifies that the records will be exported in the same order as the datalogger logged them. If a record for the table still needs to be collected, no further records will be sent until that record is collected or is deemed uncollectable by the *LoggerNet* server.
`logged-without-holes`	This specifies that records will be exported in the same order as the datalogger logged them. If a record still needs to be collected from the datalogger and there are newer records that have been collected, the missing record will be skipped in favour of the newer record.
`real-time`	This specifies that records will be exported in the order in which they were logged by the datalogger but that all but the most recently logged (and collected) record will be skipped.

Syntax

```
option := "--order-option=" order-specifier.
order-specifier := "collected" |
                   "logged-with-holes" |
                   "logged-without-holes" |
                   "real-time".
```

Default Value `real-time`

Remarks This option specifies the order in which records will be emitted for *all* of the tables on the *LoggerNet* server.

2.3.9 `start-option` Option

Specifies how the first record for each table will be chosen. The following choices are available:

`at-time`	Specifies that the starting record will have a time stamp close to the value specified by the `start-time` option (see 2.3.10)
`at-newest`	Specifies the the starting record will be the newest (in collected order) in the table.

`after-newest`	Specifies that the starting record will be the next record in logger order after the newest record in the table.
`relative-to-newest`	Specifies that the starting record that will be chosen will have a time stamp closest to the newest record less the value specified with the `backfill-seconds` (see 2.3.12) option.
`at-oldest`	Specifies that the starting record will be the oldest in the table.

Syntax

```
    option := "--start-option=" start-option-value.
    start-option-value := "at-time" |
                          "at-newest" |
                          "after-newest" |
                          "relative-to-newest" |
                          "at-oldest".
```

Default Value `at-newest`

2.3.10 `start-time` Option

Specifies the starting time stamp that will be used when the value of `start-option` (see 2.3.9) is specified as “`at-time`”.

Syntax

```
    option := "--start-time={" time-stamp "}".
```

Default Value “`1 jan 1990`”

2.3.11 Input Time Stamps

ldmp_server2 is able to recognise five different types of input time stamp formats. It uses the length and number of tokens as well as computer locale settings to distinguish between these formats. The syntax for an input time stamp is presented here:

```
time-stamp := time-format1 | ;The OS locale settings are
              time-format2 | ;used to distinguish between
              time-format3 | ;time-format1 and time-format2
              time-format4 |
              time-format5.
time-format1 := mm "-" dd "-" year " "
                [hh[":"mn[":"ss["."sf]]]].
time-format2 := dd "-" mm "-" year " "
                [hh[":"mn[":"ss["."sf]]]].
time-format3 := year mm dd
                [hh[":"mn[":"ss["."sf]]]].
```

```
time-format4 := dd " " mon [","] year " "
                [hh[":"mn[":"ss["."sf]]]].
time-format5 := mon " " dd [","] year " "
                [hh[":"mn[":"ss["."sf]]]].
mm := 2{digit}.   ; month 0 < mm <= 12
dd := 2{digit}.   ; month 0 < dd <= 31
year := [century]decade.
century := 2{digit}.
decade  := 2{digit}.
hh := 2{digit}.  ; hours into day 0 <= hh < 24
mn := 2{digit}.  ; minutes into hour 0 <= mn < 60
ss := 2{digit}.  ; seconds into minute 0 <= ss < 60
sf := 9{digit}.  ; fractional seconds
mon := "jan" | "feb" | "mar" | "apr" | "may" | "jun" |
       "jul" | "aug" | "sep" | "oct" | "nov" | "dec".
```

Note that only english abbreviations for month names are supported at this time. Also, if the year is specified without specifying the century (a two digit year value), the century will be implied as either 1900 or 2000 based upon the value of the year.

When a time stamp is expected as a part of an option value and a format is specified that uses one or more spaces, the time stamp must be enclosed in brackets ('' and '') so that the command line parser will treat the tokens in one value.

2.3.12 `backfill-seconds` Option

Specifies the number of seconds that will be subtracted from the time-stamp of the newest record in a table to select a starting record when the value of `start-option` (see 2.3.9) is specified as "relative-to-newest".

Syntax
```
     option = "--backfill-seconds=" seconds.
     seconds = digit {digit}.
```

Default Value 60 seconds

2.3.13 `client-specified` Option

Specifies whether the client should be allowed to specify the set of tables that should be exported.

Syntax
```
     option := "--client-specified=" client-specified-value.
     client-specified-value := ("true" | "1") | ("false" | "0").
```

Default Value *false*

Remarks If the value for this option is specified as *true*, the server will wait for the client to send a line of text that describes the tables and starting options for those tables before any records are sent. This case is covered in the protocol description section 3.2.

2.3.14 `close-after-disconnect` Option

Specifies that the *ldmp_server2* application should close after the last client has disconnected.

Syntax

```
option := "--close-after-disconnect=" close-after-disconnect-value.
close-after-disconnect-value := ("true" | "1") | ("false" | "0").
```

Default Value *false*

Remarks This option is provided ao that the application can close automatically after a client discontinues the service. It will also cause the connection to close if an error occurs with the server.

2.3.15 `events-to-disc` Option

Specifies that *LDMP* events should be written to disc as they are generated by the *ldmp_server2* application.

Syntax

```
option := "--events-to-disc=" events-to-disc-value.
events-to-disc-value := ("true" | "1") | ("false" | "0").
```

Default Value *true*

Remarks If this option is specified as true, the *ldmp_server2* application will attempt to create a log file containing *LDMP* related events using the parameters specified by the "`event-log-path`" (see section 2.3.16), "`event-log-file`" (see section 2.3.17), "`event-log-size`" (see section 2.3.18), and "`event-log-count`" (see section 2.3.19) program options.

2.3.16 `event-log-path` Option

Specifies the path where the event log file will be created if the "`events-to-disc`" (see section 2.3.15) option is enabled.

Syntax

```
option := "--event-log-path=" path-name.
path-name := string.
```

Default Value The `logs` directory underneath the LoggerNet working directory.

Remarks If the specified path does not exist, the server will attempt to create it.

2.3.17 `event-log-file` Option

Specifies the base name of the log file that should be created (or appended to) if the "`events-to-disc`" (see section 2.3.15) option is specified as true.

Syntax
```
option := "--event-log-file=" file-name.
file-name := string.
```

Default Value "ldmp$.log"

Remarks This option specifies the name of the active log file as well as the base pattern for baled log files. These files will be created in the directory specified by the value of the "`event-log-path`" (see section 2.3.16) option. Because this parameter specifies a parameter, the file name should contain a dollar sign character which will identify the portion of the file name that will be replaced when the log file is baled.

2.3.18 `event-log-size` Option

Specifies the size in bytes that the event log file will be allowed to grow before it is baled. This parameter will only be effective if the "`events-to-disc`" (see section 2.3.15) parameter is set to true.

Syntax
```
option := "--event-log-size=" event-log-size-value.
event-log-size-value := uint4.
```

Default Value 1 MB (1,048,576 bytes)

Remarks This parameter controls the size to which the active event log file. Once the file size has exceeded this parameter value, the active file will be closed and then renamed with a unique number replacing the dollar sign character in the file name. This action is referred to as baling the log file.

2.3.19 `event-log-count` Option

Specifies the number of baled log files that will be allowed to accumulate before the oldest are deleted. This parameter will be effective only if the "`events-to-disc`" (see section 2.3.15) option is set to *true*.

Syntax
```
option := "--event-log-count=" event-log-count-value.
event-log-count-value := uint4.
```

Default Value 5

Remarks This parameter controls the total number of baled event files that will be allowed in the event log directory specified by "`event-log-path`" (see section 2.3.16). The files are counted each time that the active event log file is baled. If the specified count is exceeded, the files will be deleted starting with the oldest until the number of files is under the specified count.

2.3.20 `export-events` Option

Specifies whether the *ldmp_server2* application should offer a separate socket service that streams *LDMP* events to connected clients.

Syntax

```
option := "--export-events=" export-events-value.
export-events-value := ("true" | "1") | ("false" | "0").
```

Default Value *false*

Remarks If this option is specified as *true*, the *ldmp_server2* application will offer a second socket service that, if attached to, will stream the events as they are created. This service will be offered on the value specified by the "`event-export-port`" (see section 2.3.21) option.

2.3.21 `event-export-port` Option

Specifies the TCP port that the *ldmp_server2* application will make available if the value of the "`export-events`" (see section 2.3.20) option is set to *true*.

Syntax

```
option := "--event-export-port=" event-export-port-value.
event-export-port-value := uint2.
```

Default Value 1025

Remarks The value specified by this option must be unique on the host machine.

2.3.22 `exclude-stats` Option

Specifies whether the LDMP server will export records from tables in the `__statistics__` data broker. If set to true, these tables will not be considered for export. If this option is set to false (the default value), the statistics table will be considered for export. Whether these tables actually are exported still depends upon the tables filter specified either on the command line or in client speciifed options.

Syntax

```
option := "--exclude-stats=" exclude-stats-value.
exclude-stats-value := bool.
```

Default Value false

Remarks This option is supported in *ldmp_server2* version 1.1.4.3 and newer.

2.3.23 `output-format` Option

Specifies whether the records will be formatted using the standard comma separated values format (the default) or the records will be formatted as XML objects. If the latter option is specified, the server output will be compatible with the expectations the ActionScript XmlSocket component.

Syntax

```
option := "--output-format=" output-format-value.
output-format-value := "csv" | "xml".
```

Default Value "csv"

Remarks This option is supported in *ldmp_server2* version 1.1.9.2 and newer.

2.3.24 `run-as-daemon` Option

Linux only. Specifies whether the program should run detached from any terminal. If set to true, the process will fork when starting up and the daemon process will not be associated with the terminal that started the process.

3 The Logger Data Monitoring Protocol

This section will describe the interactions between a *ldmp_server2* process and one of its clients. This description will be specified in three sections: starting the client session, session operation, and ending the client session.

3.1 Starting the Client Session with no Specification

A client session begins when a connection is made to the *ldmp_server2* service socket and the value of "`client-specified`" (see section 2.3.13) is *false*. A client does so by opening a socket to the computer IP interface address and the TCP service port specified in the "`service-port`" (see section 2.3.5) option. How this is done depends on the language that the client is written in.

When the client connection is made, *ldmp_server2* will attempt to create a connection to the *LoggerNet* service using the values specified with the "`server-host-name`" (see section 2.3.1) and "`server-host-port`" (see section 2.3.2) options. If that connection succeeds, *ldmp_server2* will attempt to log onto the *LoggerNet* service. If any of these operations fail, *ldmp_server2* will shut down the client session.

Once *ldmp_server2* is logged onto the *LoggerNet* service, it will begin enumeration transactions with the *LoggerNet* server in order to obtain a list of station and table names. Whenever *ldmp_server2* receives notification from the *LoggerNet* server about a station and table and that table is enabled through the value of the "`table-names`" (see section 2.3.7) option, it will start a data advise transaction on that table using the values specified in the "`order-option`" (see section 2.3.8), "`start-option`" (see section 2.3.9), "`backfill-seconds`" (see section 2.3.12), and "`start-time`" (see section 2.3.10) options.

3.2 Starting the Client Session with Specification

A client session begins when a connection is made to the *ldmp_server2* service socket and the value of "`client-specified`" (see section 2.3.13) is *true*. A client does this by opening a socket to the computer IP interface address and the TCP service port specified by the "`service-port`" (see section 2.3.5) option.

Once the socket has been opened, the server will be in a state where it is waiting for the client options specification. The first semi-colon (ascii decimal code 59) sent by the client will mark the end of this specification. If *ldmp_server2* cannot parse the specification or the specification has not been received within two minutes of the time the socket was first opened, the server will close the socket.

3.2.1 Client Specification Syntax

The syntax for this specification follows:

```
client-spec := [ output-format-spec ] { table-spec } ";".
output-format-spec := "--output-format=" ( "csv" | "xml" ).
table-spec := "{" station-name table-name { table-option } "}".
station-name := string.
table-name := string.
table-option := order-option |
                start-option.
order-option := "--order-option=collected" |
                "--order-option=logged-with-holes" |
                "--order-option=logged-without-holes" |
                "--order-option=real-time".
start-option := "--start-option=" at-time-spec |
                                  at-newest-spec |
                                  after-newest-spec |
                                  relative-to-newest-spec |
                                  at-oldest-spec.
at-time-spec   := "{at-time {" time-stamp "}}".
at-newest-spec := "at-newest".
after-newest-spec := "after-newest".
relative-to-newest-spec := "{relative-to-newest "
                           backfill-seconds "}".
at-oldest-spec := "at-oldest".
```

If the client does not specify the output format, the format used for the client session will be determined by the global output format (see "`output-format`" (see section 2.3.23)).

If starting or order options are not specified with a `table-spec`, the specified or default values of "`start-option`" (see section 2.3.9) and "`order-option`" (see section 2.3.8) will be used. If the specification contains no table specifications, the tables specified by the "`table-names`" (see section 2.3.7) option will be used.

If a client requests a table that is not specified in the "`table-names`" (see section 2.3.7) option (unless `table-names` specifies an empty list which is the default), that table will not be exported by *ldmp_server2*. By the same token, *ldmp_server2* will not export tables that are specified in the `table-names` option that are not specified in the client specification.

3.2.2 Order Option Details

The order option controls the order in which records will be exported for the table. These options correspond exactly with the values that can be associated with the "`order-option`" (see section 2.3.8). The following values are supported:

`collected`	This specifies that records will be exported in the same order that the *LoggerNet* server collected them.
`logged-with-holes`	This specifies that the records will be exported in the same order as the datalogger logged them. If a record for the table still needs to be collected, no further records will be sent until that record is collected or is deemed uncollectable by the *LoggerNet* server.
`logged-without-holes`	This specifies that records will be exported in the same order as the datalogger logged them. If a record still needs to be collected from the datalogger and there are newer records that have been collected, the missing record will be skipped in favour of the newer record.
`real-time`	This specifies that records will be exported in the order in which they were logged by the datalogger but that all but the most recently logged (and collected) record will be skipped.

3.2.3 Start Option Details

The start option controls the first record in the table that will be selected for export by the *LoggerNet* server. The following value identifiers are defined:

`at-time`	This option specifies that the first record selected will have a time-stamp equal to or greater than the time stamp specified as an argument. If a record does not exist in the table that has a time stamp that is an exact match, the record that has the closest, newest timestamp in the table will be chosen. If there is no record that has a newer time stamp, this option will be the same as the after-newest option. The time stamp argument must conform to the input time stamp specification described in section 2.3.11.

`at-newest`	This option specifies that the first record selected will be the newest record in the table in logged order. If the table is empty, this option will have the same effect as `after-newest`.
`after-newest`	This option specifies that no record will be exported until after a record has been collected that is newer (in logged order) than the newest record in the table.
`relative-to-newest`	This option specifies that the first record selected will have a time stamp greater than or equal to the time stamp of the newest record less the specified interval (in seconds).

3.2.4 An Example Specification

Here is an example of a client specification:

```
--output-format=csv
{csi_yard sec_1 --order-option=real-time
                --start-option=at-newest}
{csi_yard min_5 --order-option=logged-without-holes
                --start-option={relative-to-newest 3600}}
{csi_yard hr_1 --order-option=logged-with-holes
               --start-option=at-oldest};
```

Once *ldmp_server2* has received and successfully parsed the client specification, it will attempt to create a connection with the *LoggerNet* service specified by the "`server-host-name`" (see section 2.3.1) and "`server-host-port`" (see section 2.3.2) options. It will also attempt to log onto the service using the values of the "`logon-name`" (see section 2.3.3) and "`logon-password`" (see section 2.3.4) options.

3.3 Client Session Operation

Once the client session has been started, *ldmp_server2* will enter into a state where it is waiting for records from any of the allowed tables. Once it receives a record from any of these tables, it will output the record to the client's socket using the format specified for this session.

3.3.1 CSV Record Format

```
record-format := "\"" station-name "\","
                 "\"" table-name "\","
                 "\"" record-time-stamp "\","
                 "\"" record-number "\","
                 column {"," column} "\r\n".
station-name := string.
table-name   := string.
```

```
record-time-stamp := year "-" month "-" day " "
                     hour ":" minute ":" second
                     "." milli-second.
record-number := digit{digit}.
column := "\"" column-name "\","
          "\"" column-type "\","
          "\"" column-value "\"".
```

Each scalar value defined in the table will have its own triple pair of `column-name`, `column-type`, and `column-value`. Columns that are defined as arrays will be broken down into individual scalars with the column name mangled to reflect the scalar's array address.

The following example shows a record from one of Campbell Scientific's test stations. Note that there is normally only one line break in the output. The extra line breaks in the example have been added in order to make the record appear on the printed page.

```
"csi_yard","hr_1","2000-10-30 11:00:00.000","7765","air_c_AVG",
"FLOAT","7.489","air_c_MAX","FLOAT","8.1","air_c_MIN","FLOAT",
"6.872","pres_mbar","FLOAT","1013","rh","FLOAT","84.7",
"sol_wpm2_AVG","FLOAT","298.6","ws_mps_S_WVT","FLOAT","3.003",
"wd_deg_D1_WVT","FLOAT","0.483","wd_deg_SD1_WVT","FLOAT",
"14.31","ws_mps_MAX","FLOAT","6.37","wd_deg_SNM","FLOAT",
"351.1","prec_mm_TOT","FLOAT","0"
```

In this example, record number 7765 has been received for table "hr_1" defined on station "csi_yard". This table has twelve columns (all of them are scalars).

3.3.2 XML Record Format

When the "`output-format`" (see section 2.3.23) option is specified as "xml", the record will be formatted as an XML data structure. The root element of this structure will be named "record" and will have the following attributes:

station Specifies the name of the data broker that produced this record

table Specifies the name of the table that produced this record.

record-no Specifies the record number for this record

time Specifies the time stamp for this record. All time stamps (including those embedded in the data) will be formatted according to the following syntax:

```
time-format := year "-" month "-" day "T"
               hour ":" minute ":" second [ "." frac-second ].
```

The child elements of the "record" element are the fields associated with the record. These elements will be given the name, "field". The field values will be formatted within these elements. Each field element will have the following attributes:

name Specifies the name and, optionally, the array address of the field.

type Specifies the XSD data type of the field

process Specifies the process string used to create this field

units Specifies the units for this field.

After the record is output on the client socket, a nul (0) character will also be written to mark the end of the record.

3.4 Acknowledging Records

Once *ldmp_server2* has written the record to the socket, it will enter into a state where it is waiting for the client's acknowledgement. This acknowledgement will come from the client in the form of a single carraige return (ASCII decimal code 13). Once that acknowledgement has been received, *ldmp_server2* will switch back to state where it is waiting for the next record. Any characters that *ldmp_server2* receives from the client while not in a waiting for acknowledgement state will be ignored by the server. Any characters received by *ldmp_server2* that are not carraige returns will also be ignored.

The *LoggerNet* data advise transaction will not be advanced for a table until a record on that table has been acknowledged by the client. If the value of "`order-option`" (see section 2.3.8) is `real-time`, any records other than the newest record logged by the *LoggerNet* server will be skipped while *ldmp_server2* is waiting for the acknowledgement.

3.5 Ending the Client Session

The client session can be ended by any of the following circumstances:

- The client disconnects its end of the socket or an error occurs on the client session socket.
- *ldmp_server2*'s main connection to the *LoggerNet* server fails (This can be caused by bad network connections, server crashes, power outages, etc).
- The log-on process fails with the *LoggerNet* server (invalid user name or password).

4 *LDMP* Event Logging and Monitoring

The *ldmp_server2* application has facilities that can be enabled to log events to disc and/or to TCP sockets. The types of events logged include:

- Client connect and disconnect events
- Error messages (such as an invalid client specification)
- Record export events (such as records received)
- Operational status messages.

4.1 Event Record Format

Each event that is logged will have a regular format that conforms to comma separated value syntax. Each type of event will have a unique numeric identifier as well as a name. The identifier and name will be printed as the first two field in the event record. Specific types of events will defined fields in addition to the name and identifier. Each event record will be separated by a carraige-return line-feed sequence.

4.2 Event Log Files

If the "`events-to-disc`" (see section 2.3.15) option is enabled, the *ldmp_server2* application will log event records to disc as they are generated. The following options will control the process of baling log files:

- "`event-log-path`" (see section 2.3.16)
- "`event-log-file`" (see section 2.3.17)
- "`event-log-size`" (see section 2.3.18)
- "`event-log-count`" (see section 2.3.19)

A baling process is used to minimise the risk of filling the host computer's hard disc with event log files. The same algorithm is used within the *LoggerNet* server to control the log files that it creates as well.

If "`events-to-disc`" (see section 2.3.15) is enabled but the log files cannot be created or maintained, the *ldmp_server2* application will be aborted.

4.3 Event Export (Streaming)

If the "`export-events`" (see section 2.3.20) option is enabled, the *ldmp_server2* application will offer a socket service at the TCP port specified by the "`event-export-port`" (see section 2.3.21) option. Any number of clients can attach to this port and will be able to receive the stream of formatted events through the resultant socket as the events are generated. The format of events in this stream will be the same as the format of the events as they are recorded to disc.

The *ldmp_server2* application will ignore any data sent by the client on a log monitor socket. No acknowledgement to log records will be required by the server.

4.4 Example of Setting Up Event Logging

The following configuration file extract shows an example of how the *ldmp_server2* application can be set up to produce event logs and to allow clients to monitor the event log:

```
--events-to-disc=true
  --event-log-path=c:\campbellsci\data\ldmp-events

--export-events=true
  --event-export-port=1026
```

In this example, both event log files and event export via socket are enabled. Options are also specified that change the default event path as well as the default log export port address.

4.5 Event Types

This section describes all of the possible types of events that can be logged by the *ldmp_server2* application.

4.5.1 `LDMP server started` (Id = 1)

This message is posted when the *ldmp_server2* application is first started.

Parameters server version

Remarks Because this message is posted only when the server starts, it is likely to only show up only in the log file.

Example
```
"2001-02-21 09:00:00.000","1","LDMP server started","1.1.2.1"
```

4.5.2 `LDMP server stopped` (Id = 2)

This message type is posted when the server is being shut down normally.

Parameters none

Example
```
"2001-02-21 09:00:00.000","2","LDMP server stopped"
```

4.5.3 `LDMP server aborted` (Id = 3)

This message type is posted when the *ldmp_server2* application has had to abort for some reason.

Parameters reason for aborting

Example
```
"2001-02-21 09:00:00.000","3","LDMP server aborted",
"resource failure"
```

4.5.4 `client attached` (Id = 4)

This message is posted when a client connects to the *LDMP* service.

Parameters The client address and ephemeral port

Example
```
"2001-02-21 09:00:00.000","4","client attached","192.168.2.34",
"55067"
```

4.5.5 `client detached` (Id = 5)

This message is posted when a client connection has been shut down normally.

Parameters The client address and ephemeral port

Example
```
"2001-02-21 09:00:00.000","5","client detached","192.168.2.34",
"55067"
```

4.5.6 `table export started` (Id = 6)

This message is posted when export process has begun for a table to a client connection.

Parameters The client address, ephemeral port, station name, table name, start option, and ordering option.

Example
```
"2001-02-21 09:00:00.000","6","table export started",
"192.168.2.34","55067","csi_yard","hr_24","relative",
"logged-with-holes"
```

4.5.7 `client socket error` (Id = 7)

This message is posted when an export process has been interrupted by a client socket error (the socket connection to the client failed).

Parameters The client address and ephemeral port.

Example
```
"2001-02-21 09:00:00.000","7","client socket errror",
"192.168.2.34","55067"
```

4.5.8 `data export failure` (Id = 8)

This message is posted when a client export process has to be stopped because of an error that occurred between the *ldmp_server2* application and the *LoggerNet* server.

Parameters The client address and ephemeral port.

Example
```
"2001-02-21 09:00:00.000","8","data export failure","192.168.2.34",
"55067"
```

4.5.9 `record received` (Id = 9)

This message is posted when a record has been received from the server for a client.

Parameters The client address, ephemeral port, station name, table name, file mark number, and record number

Example
```
"2001-02-21 09:00:00.000","9","record received",
"192.168.2.34","55067","csi_yard","hr_24","0","0"
```

4.5.10 `table export ended` (Id = 10)

Parameters The client address, ephemeral port, station name table name, and reason.

4.5.11 `record sent` (Id = 11)

Parameters The client address, ephemeral port, station name, table name, file mark number, and record number.

Example
```
"2001-02-21 09:00:00.000","11","record sent",
"192.168.2.34","55067","csi_yard","hr_24","0","0"
```

4.5.12 `record acknowledged` (Id = 12)

Parameters The client address, ephemeral port, station name, table name, file mark number, and record number.

Example
```
"2001-02-21 09:00:00.000","12","record acknowledged",
"192.168.2.34","55067","csi_yard","hr_24","0","0"
```