1 Introduction

1.1 The LDEP Protocol

LDEP (Logger Data Export Protocol) is a simple protocol for exporting datalogger records formatted as text over TCP sockets. The original protocol was developed to work with Campbell Scientific's RTMC product[1] using sockets or named pipes. The protocol has since been adapted to work with data from Campbell Scientific's LoggerNet product. One significant difference between the LoggerNet servers and the original server in RTMC is that the support for named pipes has been dropped in favour of exclusive use of TCP. One other significant difference is that the LoggerNet version of the protocol allows the records to be formatted as comma separated values rather than the orignal record format defined in RTMC. The original format is still supported in LoggerNet but is much more difficult to parse.

1.2 Program Overview

ldep_server2 is designed to take the place of an earlier program, ldep_server, that was written to work with LoggerNet versions 1.1 and 1.2. The original program allowed one client connection and exported all tables that were enabled for scheduled collection. ldep_server2 performs the same way but adds the following enhancements:

- The program options are configured entirely from the command line rather than from the registry.

- The program installs an icon in the system tray and is controlled through the context menu associated with that icon. This makes it more difficult for the program to be inadvertently shut down.

- ldep_server2 is able to work with LoggerNet 2.0 and 2.1 servers as well as 1.1 and 1.2 servers.

- If the programs connection to the LoggerNet server fails, it will automatically retry to establish that connection periodically until the program is either shut down or succeeds at remaking the connection.

- Socket connections with ldep_server2 clients will be maintained even if the program has no current connection to the LoggerNet server.

- The program now keeps its state information in a text file rather than the binary file that was used by the previous version.

1.3 Description of this Document

This document serves the following purposes:

- Describes how to operate and configure ldep_server2.

- Describes the Logger Data Export Protocol (LDEP)

---

[1]RTMC was a real time monitoring system designed to work with table mode dataloggers and operating under OS/2

2   Operating the Program

2.1   Configuring the Program

All of the configuration options for `ldep_server2` are specified on the program's command line. These options can be written to a file that is then specified as the configuration file on the program's command line. The program command line has the following 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 2.1.1)

- "`server-host-port`" (see section 2.1.2)

- "`logon-name`" (see section 2.1.3)

- "`logon-password`" (see section 2.1.4)

- "`service-port`" (see section 2.1.5)

- "`config-file-name`" (see section 2.1.6)

2.1.1   `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 `ldep_server2` process will periodically attempt to connect to the server.

2.1.2 `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.

2.1.3 `logon-name` **Option**

Specifies the logon account name that the `ldep_server2` application should use to logon to the *LoggerNet* service.

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

Default Value `ldep_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 ('"').

2.1.4 `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. If the logon attempt fails, `ldep_server2` will periodically attempt to reconnect using the same account name and password.

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

2.1.5 `service-port` **Option**

Specifies the TCP port number that will be used to connect to the `ldep_server2` service.

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

Default Value 1034

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.

2.1.6 `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 ('"').

2.1.7 `format` **Option**

Syntax
```
option := "--format=" format-spec.
format-spec := comma-delimited-spec | rtms-compatible-spec.
comma-delimited-spec := "comma-delimited" | "1".
rtms-compatible-spec := "rtms-compatible" | "2".
```

Default Value `comma-delimited`

Remarks

2.1.8 `state-file-name` **Option**

Syntax
```
option := "--state-file-name=" file-name.
```

Default Value `c:\Campbellsci\ldep_server2\ldep_server2.state`

Remarks The program will attempt to read this file once when it starts up and will set its initial state based upon its content. As the program state changes, the `ldep_server2` will periodically rewrite this file with the new state.

The file contains a list of table names with identifiers (a combination of file mark and record number) for the last record sent from that table.

2.2   Controlling the Program

The program can be started either from the command prompt, from a desktop icon, or from a configured start menu item. Once the program is started, its icon will appear in the windows desktop system tray as shown in figure 1.

Figure 1: The `ldep_server2` icon in the system tray

The application menu can be accessing by clicking the right mouse button while the pointer is over the icon. Doing so will produce results similar to those shown in figure 2.
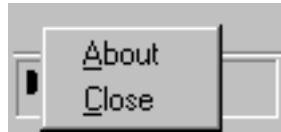

Figure 2: The application menu

Choosing "Close" from this menu will result in shutting down `ldep_server2`. Choosing "About" from the menu will result in a dialogue box similar to that shown in figure 3.
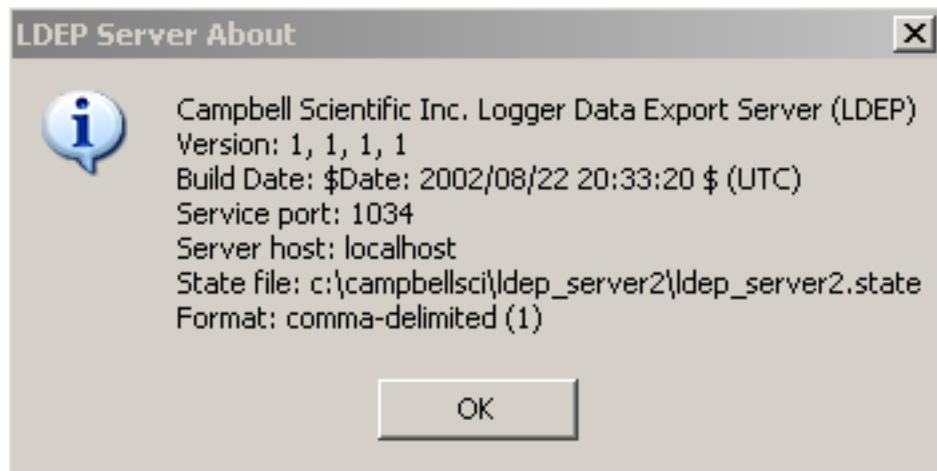

Figure 3: The about box

3 Logger Data Export Protocol

3.1 Protocol Overview

The logger data export protocol is best viewed as a stream of records sent by the export server over the socket. Each time the export server sends a record, it will wait for an acknowledgement from the client for that record. Once the acknowledgement is received, the export server will send the next record as soon as it becomes available. If the acknowledgement is not received by the export server within one minute from the time when it sent the record, the export server will re-transmit the same record. The export server will also re-transmit the record if the client sends an acknowledgment that the export server considers to be invalid. The export server is responsible for keeping track of the next record to send and will keep this state after the TCP connection between the export server and its client is lost.

A single instance of the export server can serve only one client at any given time. Once it has accepted a TCP connection from a client, it will refuse any others until that connection is closed or lost. The reason for this restriction due to the state that the server keeps for that client.

`ldep_server2` will automatically choose the tables that will be exported based upon whether those tables are a part of scheduled collection. When the export server encounters a table that it hasn't sent to the client before, it will start with the earliest record in that table. The export server will send table records in the order in which they were logged by the datalogger. If a collectable hole[2] is encountered in a table, the export server will wait on that table until the hole is filled or until the hole is no longer judged by the `LoggerNet` server to be collectable

3.2 Record Formats

3.2.1 Comma-Delimited Format

The comma-delimited format was developed to make it easier to write applications that interpreted the data. Each record that is presented in this format identifies the station and table from which it was collected as well as the time that it was logged. The name and SQL data type of each value in the record is provided as well. The following is an EBNF description of a record.

```
record := "\"" 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
```

---

[2]A hole is defined as a set of records that were skipped over in the collection process. A collectable hole is a set of records that have not yet been overwritten in the datalogger's memory.

```
                        "." milli-second.
record-number := digit{digit}.
column := "\"" column-name "\","
          "\"" column-type "\","
          "\"" column-value "\"".
column-name := string ["(" subscript {"," subscript} ")"].
column-type := sql-data-type.
column-value := string.
```

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 data type string associated with each column is a standard SQL expression that describes the column data type.

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).

3.2.2 RTMS Compatible Format

The `RTMS` compatible format was first used with the `LDEP` protocol and was made so that the string could be easily be manipulated into a SQL `INSERT` statement. Indeed, `RTMS` included a `REXX` script thatread the data from the named pipe and inserted the data into `DB2`.

This format is not particularly easy to format in a non-SQL environment but is provided in order to maintain compatibility with applications written originally to work with `RTMS`.

The following is an EBNF description of the format:

```
record := station-name "," table-name
          " (" field-spec { field-spec } ") "
          "VALUES"
          "(" field-value { field-value } ")"\r\n"
field-spec := field-name " " sql-data-type.
field-value := ( time-stamp | record-no | number | string ).
time-stamp := "'" year "-" month "-" day " " hour ":" minute ":" second "'".
record-no = 10{ digit }
number := ["+"|"-"] { digit } ["." { digit }].
```

A typical record in this format might look as follows (line breaks were added so the record could fit on the printed page):

```
Lgr,Sec15 (TMSTAMP TIMESTAMP,RECNBR DECIMAL(10,0),Battery_V FLOAT,Temp
FLOAT) VALUES ('1993-12-08 15:02:00',123456,13.5,72.123)
```

3.2.3   Record Acknowledgement Format

The client is expected to acknowledge each record that it receives by writing an acknowledgement record to the TCP connection. The acknowledgement carries the station name, table name, and record number for the record that is being acknowledged. The format of the acknowledgement is the same regardless of the record format that was chosen. This format is decribed in EBNF as follows:

```
record-ack := station-name "," table-name "," record-no "\r\n".
```

Because of the way that ldep_server2 parses the acknowledgement, the fields in the acknowledgement can be optionally surrounded by quotes. This is not a general feature of the protocol, however.