A Sierra Monitor Company

Driver Manual
(Supplement to the FieldServer Instruction Manual)

FS-8700-94 Lutron Digital microWATT Plus

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after May 1, 2001

Driver Version:	**1.05**
Document Revision:	**10**

TABLE OF CONTENTS

1. Lutron Digital microWATT Plus™ Driver Description [1]

The Digital microWATT Plus™ (DMWPL) driver allows the FieldServer to transfer data to and from Digital microwatt Plus™ -LZCs (DMWPL-LZC's) over either RS-232 or RS-485 using Digital microWATT Plus™ protocol. The FieldServer can emulate a Client (Standard operation or Application Specific operation) or a Server (Standard operation) although Server functionality is limited. The following is a summary of the Digital microWATT Plus™ driver's abilities.

Standard Operation: - Operating as a standard FieldServer Driver with customizable read/write capability. Customer has control of what data is read/written and where data is stored. Using the configuration file the customer can map data between the DMWPL-LZC and other vendor's remote devices connected using a different protocol. This mode of operation is described in sections 1.1 and 1.3

Application Specific Operation (ASO): The driver requires minimal configuration. The customer specifies connections and nodes and the driver configures itself. Data mapping between the ASO driver and another protocol is done automatically. This mode of operation is described in section 1.2

1.1. Operation as a Standard Client

The driver can request "Status" and "Power" information from a remote DMWPL-LZC, as well as send commands to the DMWPL-LZC, including control mode commands and setpoints.

Under the "flat" data option, the driver records "Status" or "Power" data from the DMWPL-LZC byte-for-byte into FieldServer Data Arrays.

Under the "collated" data option, the driver interprets the "Status" and "Power" messages returned by the DMWPL LZC's and records the information into a Present Value array and an Out of Service array. The Present Value array contains summary data about the status of the controller and the Out of Service array contains information about the controller and Server function availability. The collated data is presented in the FieldServer's Data Arrays. The location in the Data Arrays is fixed and is determined by the unit address and the data being stored. Refer to Table 2 for the array location of each data value collated by the FieldServer.

The driver can poll, on each serial port for any number of DMWPL-LZC's connected to any number of DMWPL Router Panels (DMWPL-RP's) provided that the unit and DMWPL Router Panel ID's are legal in terms of the addressing rules of the DMWPL.

The ability of the FieldServer to scale the incoming data before it is stored in the Data Arrays is not available for this driver. An object oriented BACnet driver enhancement may include scaling.

When a node is offline the driver will set the Out of Service state to true for all elements of the Out of Service Data Array.

[1] Digital MicroWATT/Digital microWATT Plus/Lutron are all registered trademarks of Lutron Electronics co inc

1.2. Application Specific Operation (ASO) as a Client

Application Specific Operation (ASO) of this driver maps data read by this driver acting as a Client to a series of complex data objects which are presented as BACnet (IP or Ethernet) Objects using the FieldServer's Server capabilities.

Thus, a BACnet (IP or Ethernet) Client can browse the FieldServer for BACnet Objects which contain the data read from the Lutron DMWPL. In addition, the BACnet Client can set objects Out of Service, can write set points and can activate modes on the Lutron DMWPL-LZC's by writing to these BACnet objects.

Refer to Appendix A for more information.

1.3. Operation as a Standard Server

The driver can emulate any number of legally addressed units.

The emulated DMWPL units can be connected to a specific port on the FieldServer or can be generalized across any port.

Commands are used to change operating states; changed set points have the effect of changing data in the FieldServer's Data Arrays but have no operational/physical effect.

No validation is performed on the values/set points sent by remote -LZCs.

1.4. Digital microWATT Plus™ Client/Server Communications

The communications between Lutron DMWPL-LZC's takes the form of messaging and message responses. Depending on the type of messages being sent to a DMWPL LZC, there can be a number of different types of responses. The responses may depend on Controller status, the type of commands received, and the integrity of the message itself.

The FieldServer can act as both a Digital microWATT Plus™ Client and Server (with limitations).

The following table outlines the range of messages that the Digital microWATT Plus™ driver will produce, and the expected responses.

Table 1: Digital microWATT Plus™ message Interaction

Message Type	Comments
Write Command	Client side message directed at a DMWPL-LZC to change settings or set points. The Server should return an ACK if the message was properly received, a NAK if the message was corrupted or could not be understood. If the driver (acting as a Server) receives a message for a unit that it is not configured for and the DMWPL-RP address is 0, then the driver will not return a message. A bad address statistic will be recorded. If the DMWPL-RP adress is non-zero then the driver will return a MIA message. If a DMWPL-RP receives a message for a unit that does not exist, it will return a message consisting of a string of X's in place of data.
Read Command	Client side message directed at a DMWPL-LZC to request status or power information. The Server should return the requested data if the command message was properly received and a NAK if the message were corrupted or could not be understood. If a DMWPL-RP receives a message for a unit that does not exist, then it will return a message consisting of a string of X's If a message is directed to a unit that not exist (and the DMWPL Router Panel address is 0), then no response is expected, and a timeout statistic will be reported in the Client side of the DMWPL. If there is a DMWPL-RP address then the driver Server side will return MIA

Communications functions - Supported functions at a glance:

Command Description	Function	Client can send	Server Response	Server can Parse
Afterhours mode Activate	Write	Yes		Yes
Afterhours mode Cancel	Write	Yes		Yes
Emergency mode Activate	Write	Yes		Yes
Emergency mode Cancel	Write	Yes		Yes
Burn-In mode Activate	Write	Yes		Yes
Burn-In mode Cancel	Write	Yes		Yes
Normal mode Activate	Write	Yes		Yes
Set Max Light Limit	Write	Yes		Yes
Cancel Max Light Limit	Write	Yes		Yes
Set Light Level	Write	Yes		Yes
Set Light Level to Off	Write	Yes		Yes
Set Load Shed	Write	Yes		Yes
Retrieve STATUS from unit	Read	Yes		Yes
Retrieve POWER from unit	Read	Yes		Yes
Wall Control Enable	Write	Yes		Yes
Wall Control Disable	Write	Yes		Yes
Occupant Sensor Enable	Write	Yes		Yes
Occupant Sensor Disable	Write	Yes		Yes
Photo Sensor Enable	Write	Yes		Yes
Photo Sensor Disable	Write	Yes		Yes
Reset Energy Counter	Write	Yes		Yes
Reset Lamp Hours Counter	Write	Yes		Yes
Ack / Nak	Response		Yes	N/A
Status	Response		Yes	N/A
Power	Response		Yes	N/A

Max Nodes Supported

FieldServer Mode	Nodes	Comments
Client	1	Only 1 Client node allowed on multidrop systems
Server	63	Limit is per port.

2. Driver Scope of Supply

2.1. Supplied by FieldServer Technologies for this driver

FieldServer Technologies PART #	Description
FS-8915-16	RJ-45 Pigtail for RS-485 connection
FS-8700-94	Driver Manual.

3. Hardware Connections

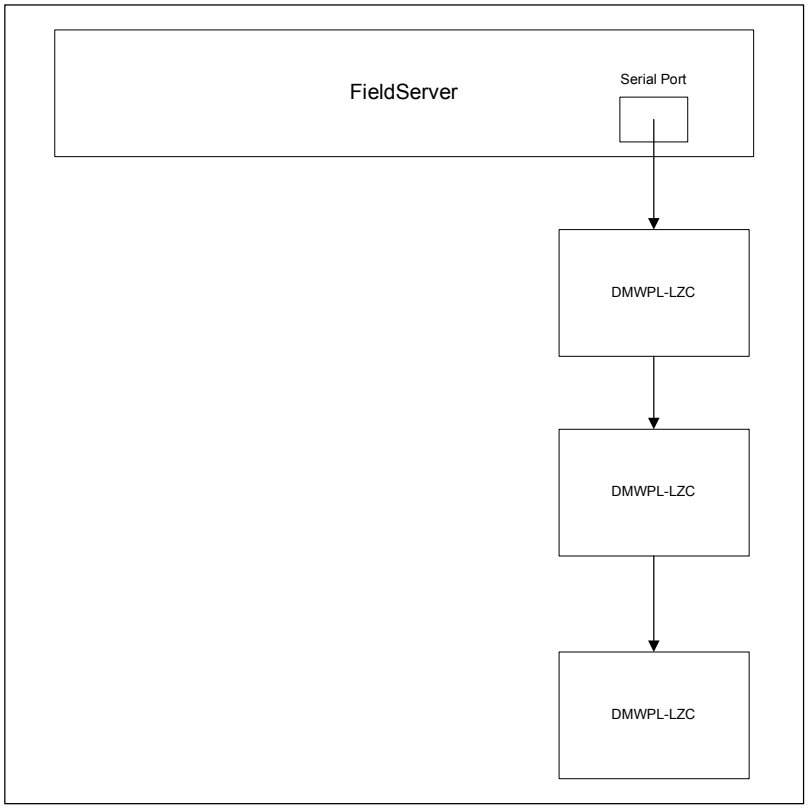
The FieldServer is connected to the Lutron DMWPL-LZC as shown below.

Configure the Lutron DMWPL-LZC according to manufacturer's instructions

3.1. Connection configurations

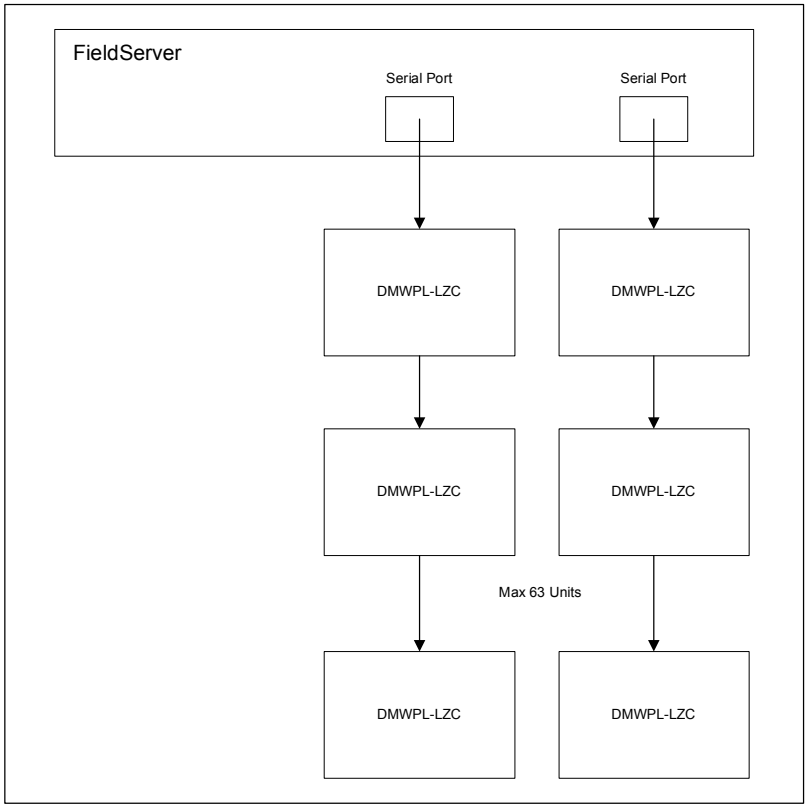
3.1.1. Connection Style 1

Use a single serial port on the FieldServer to poll a max of 63 DMWPL-LZC's connected in series. Uses DMWPL-RP address zero.



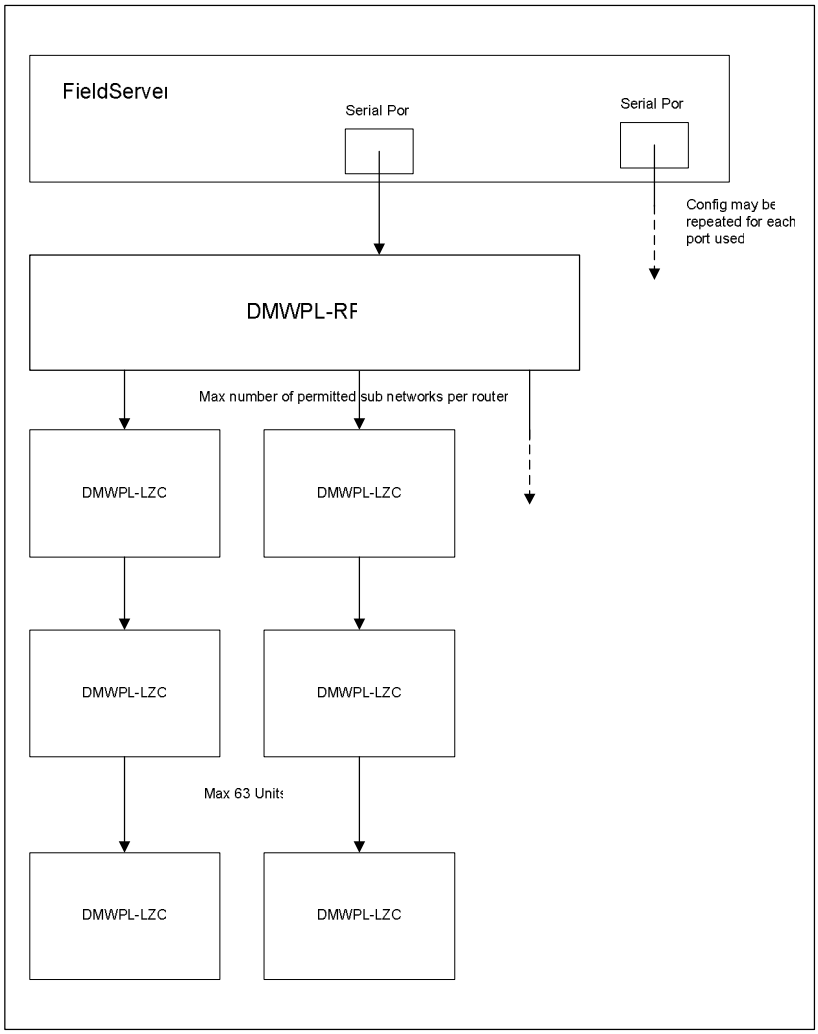
3.1.2. Connection Style 2

Use multiple serial ports on the FieldServer to poll a max of 63 DMWPL-LZC's connected in series on each port. Uses DMWPL-RP address zero.



3.1.3. Connection Style 3

Use (multiple) serial port(s) on the FieldServer to connect to DMWPL-RP's. Each DMWPL-RP may be connected to a maximum of 3 sub networks with a max of 63 DMWPL-LZC's.



3.2. Connection Notes

Cable, distance and baud rate choices and settings to comply with manufacturers requirements for connecting DMWPL- LZC's and DMWPL-RP's.

When configuring the baud rate for Lutron DMWPL connections, select 9600 if connected directly to a DMWPL-LZC and 38400 if connected via a Lutron DMWPL-RP.

Determining Router Number:
Router Number = ((Router DIP switch address) * 4) + Router Link Number
Thus: a router with DIP switch address 1 and link number 2 would be = (1*4) +2 = 6

If no router is being used (the DMW controllers are connected directly to the FieldServer), the Router number should be set to 0.

4. Configuring the FieldServer as a DMWPL Client (Standard Operation)

For a detailed discussion on FieldServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" sample files provided with the FS).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with a DMWPL Server.

When the driver is configured as a Client, it can request and store data in different array locations that are fixed in size. Currently the storage options are "flat", and "collated".

Flat storage copies incoming STATUS and POWER messages from the DMWPL-LZC byte-for-byte including the message terminators

The collated storage method is semi-object oriented. It interprets the STATUS and POWER messages and records specific information within two arrays: the present value (PV) array, and the missing-in-action array (MIA) or Out of Service. There are 20 PV's that this method records, and a corresponding number of MIA status members. If a read request from the FieldServer results in an MIA response, then all 20 members of the MIA array will be set to node offline or missing. Other logic specifying the 20 PV elements may also affect specific MIA states.

As a Client, the driver can also change DMWPL-LZC settings and status through write Map Descriptors which issue write commands.

A separate Map Descriptor is required for each DMWPL-RP (including DMWPL-RP zero). This Map Descriptor will define the base storage location to be used to store the data for unit 1 on that DMWPL-RP. All other units will be stored relative to this base offset.

For collated data storage, each LZC polled will have the following data stored. The offsets in the following table are relative to the base offset for the LZC. The base offset is based on the unit address.

Table 2 Collated Data Arrays

Data Array[2]		1	2
Data Array Elements Used		1	1
Obj Id.	Description	Data Array Offset	
		Present_Value	Out_of_Service (Missing in Action)
1	Relay State	0	0
2	Light Control	1	1
3	Operating State	2	2
4	Load Shed	3	3
5	Max Light Limit	4	4
6	Wall Control	5	5
7	Wall Control En/Disable	6	6
8	Occupancy	7	7
9	Occupant Sensor En/Disable	8	8
10	Photosensor Cutback	9	9
11	Photosensor Cutback En/Disable	10	10
12	Power	11	11
13	Energy	12	12
14	Burn-in Hours	13	13
15	Lamp Hours	14	14
16	Timing State	15	15
17	Time Remaining	16	16
18	Current Light Level	17	17
19	Timing State (Analog)	18	18
20	Operating State (Analog)	19	19

Notes:
The timing state of a DMWPL unit is determined from the "event" byte from a status message, and is also stored as offset 15 and 18 in the present value array shown above. Each timing state is exclusive. I.e. multiple states cannot exist at the same time.

Table 3 - Valid Timing State Values

Present Value	Corresponding Timing State
1	Time to Flash
2	Time to Off
3	Occupied Timeout

The operating state of a DMWPL unit is determined from the "mode" bytes from a status message, and is also stored as offset 3 in the present value array shown above. The 6 possible modes are recorded in the following table. The modes occur in a hierarchy. According to Lutron protocol, if a mode higher than the current mode is set, then the driver will remember the lower mode's state. If a mode lower than the current mode is set, then all modes above this mode will be cleared. Therefore, operating states do not have to be exclusive; there can be multiple operating states at the same time; however, there are a set of rules defining the how the states

[2] Data is stored in two Data Arrays

are turned on and off. For the Client side driver, the highest operating state in the hierarchy is stored in the present value array. For the Server side driver, the modes are recorded in the STATE array as well as the flat arrays according to valid write commands. Note: there are no Maintenance mode active/cancel commands; therefore, maintenance modes are not shown in the Server side STATE array.

Table 4 - Valid Operating State Values

Present Value	Corresponding Mode
1	Emergency
2	Burn In
3	Maintenance (2 modes here)
4	After Hours
5	Normal

Both maintenance modes are represented by the same present value.

4.1. Data Arrays/Descriptors

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for DMWPL communications, the driver independent FieldServer buffers need to be declared in the “Data Arrays” section, the DMWPL-LZC addresses need to be declared in the “Client Side Nodes” section, and the data required from the Servers needs to be mapped in the “Client Side Map Descriptors” section. Details on how to do this can be found below.

Currently this driver can send/receive DMWPL messages and store the received information into two formats on the Client side, and only one format on the Server side. On the Client side, the formats are: "flat", and "collated".

The "flat" Data Arrays simply provide a location where incoming STATUS and POWER messages from a DMWPL-LZC are copied byte-for-byte into a FieldServer Data Array including terminating characters.

The two "collated" Data Arrays store values interpreted from the STATUS and POWER data messages. Currently these arrays can store up to 20 predefined values. The MIA Data Array stores unit online/offline states: a "1" indicates offline, and a "0" means online or Ok. The present value array stores values represented as floating decimal points. The collated Data Array table above provides a complete listing of the stored values and their offset in the FieldServer's array. For more complete information on the meaning of these values please consult Lutron DMWPL documentation.

On the Server side, all messages are stored as "flat" arrays. An additional array is used to summarize STATE information that is manipulated by incoming write commands. For the driver's Server side, there is no difference between collated or flat storage. All incoming command messages to the driver's Server side will only be stored into flat arrays and the STATE array. The following table identifies the parameters stored in the Server side STATE array. The flat arrays store data into the POWER and STATUS arrays in the same structure as response messages. See Lutron DMWPL's communication protocol for details on this message structure.

4.1.1. Server Side "State" Array

Function	Accepted values	Storage offset	Comments
Afterhours Mode	0/1	0	
Emergency Mode	0/1	1	
Burn-in Mode	0/1	2	
Normal Mode	0/1	3	
Set High End	0-99	4	function disabled
Set Low End	0-99	5	function disabled
Max Light Limit	0-99	6	
Set Light Level	0-99	7	
Set Load Shed	0-99	8	
Wall Control	0/1	9	
Occupant Sensor Enable	0/1	10	
Photo Sensor Enable	0/1	11	

Note that in the tables, * indicates an optional parameter, with the bold legal value being the default.

Section Title
Data_Arrays

Column Title	Function	Legal Values
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_Array_Format	Provide data format. Each Data Array can only take on one format.	Float, Bit, UInt16, SInt16, Packed_Bit, Byte, Packed_Byte, Swapped_Byte
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required by the Map Descriptors for the data being placed in this array.	1-10,000

Example

```
//    Data Arrays

Data_Arrays
Data_Array_Name,    Data_Format,    Data_Array_Length
FLAT_POWER,         BYTE            30
FLAT_STATUS,        BYTE,           68
MAS_COLLATE_PV,     FLOAT,          20
MAS_COLLATE_MIA,    BYTE ,          20
MW_STATE,           UINT16,         20
```

4.2. Client Side Connection Descriptions

Section Title
Connections

Column Title	**Function**	**Legal Values**
Port	Specify which port the LZC is connected to the FieldServer	P1-P8, R1-R2[3]
Protocol	Specify protocol used	microWATT,
Baud*	Specify baud rate	110 – 115200, standard baud rates only, **9600**
Parity*	Specify parity	Even, Odd, **None,** Mark, Space
Data_Bits*	Specify data bits	7, **8**
Stop_Bits*	Specify stop bits	**1**
Handshaking*	Specify hardware handshaking	RTS, RTS/CTS, **None**
Poll _Delay*	Time between internal polls	0-32000 seconds, **1 second**
Application*	See Section Appendix A for more information	**None** Lutron-to-BACnet-Ethernet Lutron-to-BACnet-IP Lutron-to-Virtual-BACnet-Ethernet Lutron-to-Virtual-BACnet-IP

Example

```
//      Client Side Connections

Connections
Port,       Protocol,       Baud,       Parity,     Handshaking,     Poll_Delay
P8,         microWATT,      9600,       None,        None,           0.100s
```

[3] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

4.3. Client Side Node Descriptors

Section Title		
Nodes		
Column Title	**Function**	**Legal Values**
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	DMWPL unit address of physical Server node	1-255
Protocol	Specify protocol used	microWATT
Connection	Specify which port the -LZC is connected to the FieldServer	P1-P8, R1-R2[4]
Route	DMWPL-RP address. This is a calculation based on which DMWPL-RP port a DMWPL unit is attached to, as well as the DMWPL-RP number. There are three possible ports, and 21 possible DMWPL-RP numbers. Refer to section 4.4 for information on how to calculate DMWPL-RP addresses.	1-255

Example

```
//  Client Side Nodes

Nodes
Node_Name,          Node_ID,    Protocol,     Connection,    Route
microWATT_1,        1,          microWATT,    P8,            5
```

4.4. DMWPL -RP Address Calculation

For FieldServer drivers, the destination nodes need to be configured for all commands directed at DMWPL-LZCs. The Node_ID represents the DMWPL "Unit number" and the Route defines the "DMWPL-RP address". DMWPL-RPs are numbered with dip switches according to Lutron supplied job drawings. This numbering ranges from 1 up. However, this is NOT the DMWPL-RP address to be supplied to the FieldServer configuration files. The Lutron DMWPL communication protocol defines the DMWPL-RP address to be a combination of the DMWPL-RP number (set by the dip switches) and DMWPL-RP port number. There are 3 ports per DMWPL-RP, and these are numbered 1, 2, and 3.

Install Table of Valid DMWPL-RP no's

[4] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

4.5. Client Side Map Descriptors

4.5.1. FieldServer Related Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from "Data Array" section above
Data_Array_Offset	Starting location in Data Array	0 to maximum specified in "Data Array" section above
Function	Function of Client Map Descriptor	RDBC, WRBC, WRBX, WRB

4.5.2. Driver Related Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the node names specified in "Client Node Descriptor" above
Command	First command field	See Lutron DMWPL for protocol specifications
Cmd1	Second command field	
Cmd2*	Third command field. This field is not normally required. If Lutron adds a command to the protocol and the command is not supported by this driver then use this field to complete the specification of the command.	
Cmd3*	Fourth command field. This field is not normally required. If Lutron adds a command to the protocol and the command is not supported by this driver then use this field to complete the specification of the command.	Numeric character. See Lutron DMWPL for protocol specifications
Length	Length of Map Descriptor	1-125 (Register,AI) 1-800 (Coil, DI)
Address	Starting address of read block	This field must be set to a dummy value as indicated in the Map Descriptor examples. It is used to avoid conflict over responsible Map Descriptors.

4.5.3. Timing Parameters

Column Title	Function	Legal Values
Scan_Interval	Rate at which data is polled	≥0.001s

4.5.4. Map Descriptor Example 1 - Collated Read

This example shows how to set up a configuration to read collated data from a DMWPL-LZC. The Data_Array name and the Data_Byte name must reference the correct Data Arrays with the appropriate size.

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name,  Scan_Interval,  Function,  Data_Array_Name,  Data_Array_Offset,  DA_Byte_Name,  DA_Byte_Offset,  Node_Name,  Address,  Length,  Store_type
Get_Status,           1.0s,           Rdbc,      MAS_COLLATE_PV,   0,                  MAS_COLLATE_MIA,  0,              node_A,     0,         10,      collated
Get_Power,            1.0s,           Rdbc,      MAS_COLLATE_PV,   10,                 MAS_COLLATE_MIA,  0,              node_A,     10,        10,      collated
```

Data Arrays used to store the PV and MIA status of a unit. They must be declared in the order shown

"collated" defines the type of storage method. Others include "flat".

Node name identifies the DMWPL unit you wish to receive information from. The unit number is specified in Client side node definitions.

4.5.5. Map Descriptor Example 2 - Flat Read

This example shows how to set up a configuration to read flat byte-for-byte messages from a DMWPL-LZC. The Data_Array name and the Data_Byte name must reference the correct Data Arrays with the appropriate size.

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name,  Scan_Interval,  Function,  Data_Array_Name,  Data_Array_Offset,  Node_Name,  Address,  Length,  Store_type
Get_Status,           1.0s,           Rdbc,      FLAT_STATUS,      0,                  node_A,     0,         68,      flat
Get_Power,            1.0s,           Rdbc,      FLAT_POWER,       0,                  node_A,     0,         30,      flat
```

"flat" defines the type of storage method. Others include "collated"

Data Arrays used to store the STATUS and POWER messages byte-for-byte. They must be declared to be at least 68 and 30 bytes respectively.

Node name identifies the DMWPL unit you wish to receive information from. The unit number is specified in Client side node definitions.

4.5.6. Map Descriptor Example - Write Commands

This example shows all the available configurations to write data to a DMWPL-LZC from the FieldServer DMWPL driver.

Map_Descriptor_Name,	Data_Array_Name,	Data_Array_Offset,	Scan_Interval,	Function,	Node_Name,	Address,	Length,	Store_type
Afterhours_active,	DA1,	0,	1.0s ,	Wrbx,	node_A,	0,	1,	mw_write
Afterhours_cancel,	DA2,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Emergency_active,	DA3,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Emergency_cancel,	DA4,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
BurnIn_active ,	DA5,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
BurnIn_cancel ,	DA5,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Normal_active *,	DA7,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Set_max_light_limit* ,	DA8,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Cancel_max_light_limit,	DA9,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Set_light_level,*	DA10,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Set_light_level_off,	DA11,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Set_load_shed, *	DA12,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Wall_control_enable,	DA13,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Wall_control_disable,	DA14,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Occupant_sensor_enable,	DA15,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Occupant_sensor_disable,	DA16,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Photo_sensor_enable,	DA17,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Photo_sensor_disable,	DA18,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Reset_energy_counter,	DA19,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write
Reset_lamp_hour_counter,	DA20,	0,	1.0s ,	wrbx ,	node_A ,	0 ,	1 ,	mw_write

wrbx identifies this as a write rather than a read command

.

* These commands use a value found at in the Data Array Asscoiated with the Mapdesc. The other commands do not reference Data Array values. To envoke a write command not requiring additional data, any value needs to be written to the Data Array and offset referenced. This event will trigger the driver to initiate the associated command message once only. To envoke a write command that requires input data, the setpoint or value should be entered in the Data Array and offset referenced in the Map Descriptor. Again the driver will send the correct command message, but will use the data value recently enter into the Data Array.

5. Configuring the FieldServer as a DMWPL Server

For a detailed discussion on FieldServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" files on the driver diskette).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with a DMWPL Client.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for DMWPL communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the FieldServer virtual node(s) needs to be declared in the "Server Side Nodes" section, and the data to be provided to the Clients needs to be mapped in the "Server Side Map Descriptors" section. Details on how to do this can be found below.

Note that in the tables, * indicates an optional parameter, with the bold legal value being the default.

The microWATT Plus Server can record commands and set points in FieldServer Data Arrays; however, they will have no other physical affects other than changing stored data. The Server can also generate STATUS and POWER response messages based upon the "flat" data stored in its arrays.

Three Data Arrays are relevant when configuring the DMWPL as a Server: the STATE, STATUS, and POWER arrays. The STATE array records setpoint and status information sent by Client write commands. The STATUS and POWER arrays are "flat" byte-for-byte copies of response messages of the same name. These arrays include space for message terminators.

If the driver's Server side receives a write command message, then the Server will update all three of its Data Arrays with the relevant information. If the driver's Server side receives a read command message, then it will return either the POWER or STATUS message stored in the Data Arrays byte-for-byte.

5.1. Server Side Connection Descriptors

Section Title		
Connections		
Column Title	**Function**	**Legal Values**
Port	Specify which port the -LZC is connected to the FieldServer	P1-P8, R1-R2[5]
Protocol	Specify protocol used	microWATT
Baud*	Specify baud rate	110 – 115200, standard baud rates only, **9600**
Parity*	Specify parity	Even, Odd, **None,** Mark, Space
Data_Bits*	Specify data bits	7, **8**
Stop_Bits*	Specify stop bits	**1**
Handshaking*	Specify hardware handshaking	RTS, RTS/CTS, **None**
Server_Hold_Timeout*	Specifies time FieldServer will reserve Server side connection while waiting for the Client side to update data in Data_Array (if necessary)	>1.0s

Example

```
//      Server Side Connections

Connections
Port,           Protocol,       Baud,       Parity,     Handshaking
P8,             microWATT,      9600,       None,       None
```

5.2. Server Side Node Descriptors

Section Title		
Nodes		
Column Title	**Function**	**Legal Values**
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	DMWPL unit address of physical Server node	1-255
Protocol	Specify protocol used	microWATT
Server_Hold_Timeout*	Specifies time FieldServer will reserve Server side connection while waiting for the Client side to update data in Data_Array (if necessary)	>1.0s

[5] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

Example

```
//  Server Side Nodes

Nodes
Node_Name,          Node_ID,        Protocol        *
microWATT_2,        1,              microWATT
```

5.3. Server Side Map Descriptors

5.3.1. FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from "Data Array" section above
Data_Array_Offset	Starting location in Data Array	0 to maximum specified in "Data Array" section above
Function	Function of Server Map Descriptor	Server
Server_Hold_Timeout*	Specifies time FieldServer will reserve Server side connection while waiting for the Client side to update data in Data_Array (if necessary)	>1.0s

5.3.2. Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of target node acting as a Server	One of the node names specified in "Server Node Descriptor" above
Command	First command field	See Lutron DMWPL for protocol specifications
Cmd1	Second command field	
Cmd2	Third command field. Not normally required.	
Cmd3	Fourth command field. Not normally required.	
Length	Length of Map Descriptor	
Address	Starting address of read block	

* Note No information is needed on Server side.

5.3.3. Map Descriptor Example - Server side STATUS and POWER

This example shows how to configure the DMWPL driver to act as Server to respond to STATUS and POWER requests. The data on the Server side is stored as response messages already, so a response consists of a byte-for-byte copy of FieldServer Data Arrays to message.

```
//  Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name,    Function,   Data_Array_Name,    Data_Array_Offset,  Node_Name,  Store_type
Retrieve_STATUS,        Passive,    STATUS_FLAT,        0,                  node_A,     flat
Retrieve_POWER,         Passive,    POWER_FLAT,         0,                  node_A,     flat
```

These Map Descriptors are passive and respond to incoming requests

Data Array names identify what Data Array the Server side will respond with.

Offset should always be zero.

Node name identifies the Server side DMWPL unit number

All Server side Map Descriptors are stored as flat

5.3.4. Map Descriptor Example - response to enable/disable and set point commands

This example shows how to configure the DMWPL driver to act as Server to respond to STATUS and POWER requests. The data on the Server side is stored as response messages already, so a response consists of a byte-for-byte copy of FieldServer Data Arrays to message.

```
// Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name,  Function,  Data_Array_Name,  Data_Array_Offset ,  DA_Byte_Name,  DA_Byte_Offset ,  DA_Float_Name,  DA_Float_Offset,  Node_Name,  Store_type
Afterhours_active  ,  Passive ,  MW_STATE,        0                  ,  POWER_FLAT,    0               ,  STATUS_FLAT,    0               ,  node_A   ,  flat
Afterhours_cancel ,   Passive ,  MW_STATE ,       0                  ,  POWER_FLAT,    0               ,  STATUS_FLAT,    0               ,  node_A   ,  flat
Emergency_active ,    Passive ,  MW_STATE ,       1                  ,  POWER_FLAT,    0               ,  STATUS_FLAT,    0               ,  node_A   ,  flat
Emergency_cancel,     Passive ,  MW_STATE ,       1                  ,  POWER_FLAT,    0               ,  STATUS_FLAT,    0               ,  node_A   ,  flat
BurnIn_active      ,  Passive ,  MW_STATE ,       2                  ,  POWER_FLAT,    0               ,  STATUS_FLAT,    0               ,  node_A   ,  flat
BurnIn_cancel      ,  Passive ,  MW_STATE ,       2                  ,  POWER_FLAT,    0               ,  STATUS_FLAT,    0               ,  node_A   ,  flat
Normal_active      ,  Passive ,  MW_STATE ,       3                  ,  POWER_FLAT,    0               ,  STATUS_FLAT,    0               ,  node_A   ,  flat
```

Data Array names identify the Data Array the where the Server side will store information. For Client side write commands, the Server may store to any of the STATE, STATUS, or POWER arrays so all are given. These arrays must appear in the correct field shown.

Data-array offset defines the storage location within the STATE array only. Use the values provided

The POWER and STATUS arrays are always zero referenced.

Node name identifies the Server side DMWPL unit number

All Server side Map Descriptors are stored as flat arrays.

Appendix A.1. Application Specific Operation as a Client

Application Specific Operation (ASO) links the Lutron DMWPL driver as a Client to a BACnet Ethernet or BACnet IP Server on the FieldServer.

Normally the relationship between the data read by a Client and data made available by a different Server is established by building a configuration file which maps the data from the one to the other. This is termed Standard Operation.

A minimal configuration is required in the configuration CSV file - Client side connections and nodes and the Server side connection and node need to be defined.

Appendix A.2. The CSV file 'Application' parameter

You can control the application specific operation by using different keywords when specifying the 'Application' parameter for the connection. This is done in the configuration CSV file.

The following are valid entries

1. None
2. Lutron-to-BACnet-Ethernet
3. Lutron-to-BACnet-IP
4. Lutron-to-Virtual-BACnet-Ethernet
5. Lutron-to-Virtual-BACnet-IP

When you use choices 2 & 3 one BACNet node will be created and this node. This node will have 20 objects per Lutron Node.

1. Lutron Node 1
2. Lutron Node 2 One single BACnet Node with x sets of 20 objects
3. Lutron Node x

When you use choices 4 & 5 then for each Lutron DMWPL node a BACnet node will be created.

1. Lutron Node 1 BACnet virtual node 1
2. Lutron Node 2 BACnet virtual node 2
3. Lutron Node x BACnet virtual node x

The node number of the BACNet nodes is based on the DMW Node and Route parameters. The following formula can be used to determine the node number that the driver will generate

If (route >= 4 and <= 6) BACnet Node_ID = (route - 4) * 256 + DMW Node_ID ;
Otherwise BACnet Node_ID = (route + 10) * 256 + DMW Node_ID ;

When the route number is not in the expected range of 4-6 then the node number generated will be quite different allowing a user to note the exception quite easily.

Appendix A.3. ASO Configuration CSV File

Example 1 - No connection or nodes are defined for the Server side.

The DMWPL driver will create a default connection on adapter N1 and a default Server node.

```
//  Client Side Connections

Connections
Port ,          Baud ,      Protocol  ,         Application
P1   ,          9600 ,      microWATT,          Lutron-to-BACnet-Ethernet
```

Make one connection for each DMW connection to the FieldServer.

Set the baud rate, parity etc. as required.

Additional information on these parameters is provided in section 4.1

Make this driver perform Application Specific Operation by specifying the *Application* parameter.

Valid entries are:
Lutron-to-BACnet-Ethernet
Lutron-to-BACnet-IP
Lutron-to-Virtual-BACnet-Ethernet
Lutron-to-Virtual-BACnet-IP

```
//  Client Side Nodes
//
Nodes
Node_Name ,   Port,   Protocol,     Node_ID,   Route,   Probation_Delay,   Retries
node_A    ,   P1,     microWATT,    1      ,   5     ,   1.0s           ,   0
```

Create one node for each DMW device you wish to poll for data. Specify the Node_ID and route as required.

Each node must have a unique name.

The Node_ID and Route will be used to build the Id of each object. Additional information on these parameters is provided in section 4.2

Follow these recommendations for best performance.

Example 2: A Server node and connection have been specified.

The DMWPL driver will look for a connection and node before creating the default. If a Server node is found (with protocol BACnet_IP or BACnet_Ethernet) then that node will be used as the Server node.

```
//  Client Side Connections

Connections
Port ,          Baud ,      Protocol  ,         Application
P1   ,          9600 ,      microWATT,          Lutron-to-BACnet-Ethernet
```

```
//  Client Side Nodes
```

```
Nodes
Node_Name ,   Port,   Protocol,     Node_ID,   Route,   Probation_Delay,   Retries
node_A    ,   P1,     microWATT,    1      ,   5    ,   1.0s           ,   0
node_B    ,   P1 ,    microWATT,    2      ,   5    ,   1.0s           ,   0
```

```
//  Server Side Connections
//
Connections
Adapter,                    Protocol
N1     ,                    BACnet_Eth
```

```
//  Server Side Nodes
//
Nodes
Node_Name ,          Protocol   ,          Node_ID
bnetA     ,          BACnet_Eth ,          1
```

The DMW driver searches the CSV file for a BACnet_Ethernet node. If it finds one, it uses that node to create the BACnet objects so that remote BACnet Clients can browse the FieldServer. If it finds more than one then an error is printed.

Appendix A.4. The 20 BACnet Data Objects created per Node.

For each DMWPL node, 20 Data BACnet Objects are created. They are tabulated below. The values of some properties are hard coded.

Relay State		**Light Control**	
Binary Output		Analog Output	
Object_Identifier	1	Object_Identifier	2
Object_Name	"RelayState"	Object_Name	"Light Control"
Object_Type	Binary Output	Object_Type	Analog Output
Present_Value	1,0	Present_Value	0-99
Active_Text	"ON"	Units	Percent
Inactive_Text	"OFF"	Relinquish_Default	99
Out_Of_Service	true when MIA	User-Input calls "Set Light Level" and uses input from user as parameter	
Relinquish_Default	0		
User-Input: "ON" calls "Set Light Level" and sets light level to 99;."OFF" calls "Set Light Level to Off"			
Operating State		**Load Shed**	
Multistate Output		Analog Output	
Object_Identifier	3	Object_Identifier	4
Object_Name	"OperatingState"	Object_Name	"Load Shed"
Object_Type	Multi-state Output	Object_Type	Analog Output
Present Value[6][7]	Emergency(0), Afterhours(1), Normal(2), Burn-in(3), Maintenance(4)	Present_Value	0-99
Number_Of_States	5	Units	Percent

[6] Maintenance mode is reported when mode is not Emergency, Afterhours, Normal or Burn-In.
[7] No command must be sent if user requests Maintenance mode through BACnet object

Out_Of_Service	True when MIA	Out_Of_Service	true when MIA
Relinquish_Default	"Normal"	Relinquish_Default	0
Each time that the driver writes to the -LZC it sends the 'Activate Normal Mode; command and then if necessary it sends a second command to activate the new state.		User-Input calls "Set Load Shed" and uses input from user as parameter	
User-Input values should be able to Activate and Cancel Afterhours mode, Emergency mode,Burn-in mode, and Activate Normal mode. (using the corresponding command)			
Max Light Limit		**Wall Control**	
Analog Output		Analog Output	
Object_Identifier	5	Object_Identifier	6
Object_Name	"MaxLightLimit"	Object_Name	"WallControl"
Object_Type	Analog Output	Object_Type	Analog Output
Present_Value	0-99	Present_Value	0-99
Units	Percent	Units	Percent
Out_Of_Service	true when MIA	Out_Of_Service	true when MIA
Relinquish_Default	99		
User-Input calls "Set Max Light Limit" and uses input from user as parameter When the user input is equal to 99 then the driver also sends a 'cancel max light limit' command to the Lutron -LZC.			
Wall Control Enable/Disable		**Occupancy**	
Binary Output		Binary Input[8]	
Object_Identifier	7	Object_Identifier	8
Object_Name	"WallControlEnableDisable"	Object_Name	"Occupancy"
Object_Type	Binary Output	Object_Type	Binary Input
Present_Value	0,1	Present_Value	1,0
Active_Text	"Enabled"	Active_Text	"Occupied"
Inactive_Text	"Disabled"	Inactive_Text	"Unoccupied"
Out_Of_Service	true when MIA	Out_Of_Service	true when MIA
User-Input calls "Wall Control Enable" or "Wall Control Disable"			
Occupant Sensor Enable/Disable		**Photosensor Cutback**	
Binary Output		Analog Input[8]	
Object_Identifier	9	Object_Identifier	10
Object_Name	"OccupantSensorEnableDisable"	Object_Name	"PhotosensorCutback"
Object_Type	Binary Output	Object_Type	Analog Input
Present_Value	0,1	Present_Value	0-99
Active_Text	"Enabled"	Units	Percent
Inactive_Text	"Disabled"	Out_Of_Service	true when MIA
Out_Of_Service	true when MIA		
User-Input calls "Occupant Sensor Enable" or "Occupant Sensor Disable"			
Photosensor Cutback Enable/Disable		**Power**	
Binary Output		Analog Input[8]	
Object_Identifier	11	Object_Identifier	12
Object_Name	"PhotosensorControlEnableDisable"	Object_Name	"Power"
Object_Type	Binary Output	Object_Type	Analog Input
Present_Value	0,1	Present_Value	0x0000 to 0xFFFF
Active_Text	"Enabled"	Units	None
Inactive_Text	"Disabled"	Out_Of_Service	true when MIA
Out_Of_Service	true when MIA		
Relinquish Default	1	Value is reported as a unit less number (clicks) where 1 click = 5/32 watt-hour, so clicks*32/5 = watt-hours	
User-Input calls "Photo Sensor Enable" or "Photo Sensor Disable"			
Energy		**Burn-in Hours**	
Analog Output		Analog Input[8]	
Object_Identifier	13	Object_Identifier	14
Object_Name	"Energy"	Object_Name	"BurnInHours"
Object_Type	Analog Output	Object_Type	Analog Input
Present_Value	0x00000000 to 0xFFFFFFFF	Present_Value	000-099
Units	Watt-hours	Units	Hours
Out_Of_Service	true when MIA	Out_Of_Service	true when MIA
Relinquish Default	0		

[8] DMWPL value comes from STATUS response, byte 48

Need to convert "Clicks" (bytes 8-11) to watts: conversion is click = 5/32 watt hours, so clicks*5/32 = watt hours			
User-Input calls "Reset Energy Counter" (no parameters)			
Lamp Hours		**Timing State**	
Analog Output		Multistate Input	
Object_Identifier	15	Object_Identifier	16
Object_Name	"LampHours"	Object_Name	"TimingState"
Object_Type	Analog Output	Object_Type	Multi-state Input
Present Value	000 to 999	Present_Value	None, Warning, OffDelay, OccExtension
Out_Of_Service	true when MIA	Number_Of_States	4
Units	hundreds of hours	Out_Of_Service	true when MIA
Relinquish_Default	0		
User-Input calls "Reset Lamp Hours Counter" (no parameters)			
Time Remaining		**Current Light Level**	
Analog Input		Analog Input	
Object_Identifier	17	Object_Identifier	18
Object_Name	"TimeRemaining"	Object_Name	"CurrentLightLevel"
Object_Type	Analog Input	Object_Type	Analog Input
Units	minutes	Present_Value	0 to 99
Out_Of_Service	true when MIA	Units	percent
		Relinquish_Default	0
		Out_Of_Service	true when MIA
Timing State as an Analog Input[9]		**Operating State as an Analog Output**[10]	
Analog Input		Analog Output	
Object_Identifier	19	Object_Identifier	20
Object_Name	"TimingStateAsAI"	Object_Name	"OperatingStateAsAO"
Object_Type	Analog Input	Object_Type	Analog Output
Present Value	None, Warning, OffDelay, OccExtension	Present_Value	Emergency (0), Afterhours (1), Normal (2), Burn-in (3), Maintenance (4)
Number_Of_States	4	Number_of_States	5
Out_Of_Service	true when MIA	Relinquish Default	Normal
		User-Input values should be able to Activate and Cancel Afterhours mode, Emergency mode and Burn-in mode, and Activate Normal mode. (using the corresponding command)	

Appendix A.5. Notes on Object Properties

Object_Identifier - Every instance of every object has been allocated a unique ID using the object identifier value found in Appendix A.4.

The ID is calculated as follows
ID = Object_Identifier from Appendix A.4 + 100 * Node_ID + 10000 * DMWPL -RP

And thus has the form Rrddnn where:
nn is the Object_Identifier from Appendix A.4
dd is the -LZC address (Node_ID in FieldServer terminology)
rr is the DMWPL -RP address (Route in FieldServer terminology)

Object_Name -Most BACnet Client browsers list the objects they find on a Server by name. To facilitate this process we have created a unique name for each instance of each object by appending the Object_Identifier to the Object_Name.

[9] Duplicate of 'Timing State' except for type change
[10] Duplicate of 'Operating Mode' except for type change

Out of Service - Each object has an Out_of_Service property. The value of this property can be changed by a BACnet Client. This property must be set/unset appropriately to allow the BACnet Client to change the value of the Present_Value of an object.

Inputs : (AI/BI/MI)	Remote BACnet Client cannot write to the present value (Pv) unless the object has been put out of service.
Outputs: (AO/BO/MO/AV/BV/MV)	Remote BACnet Client cannot write to the present value (Pv) unless the object has been put into service.

When the remote BACnet Client queries the value of the Out_Of_Service property the value returned is a calculated value and is not simply the state of the property.

The value returned is calculated as follows:

Node Status	OOS Property value	Value returned to remote Client
Good	0	0 (In Service)
Good	1	1 (out of Service)
Bad	0	1
Bad	1	1

A node's status is bad if the Lutron DMWPL-RP returns a MIA message or if a Lutron DMWPL-LZC does not respond to a poll when the FieldServer is directly connected to it.

Appendix A.6. Driver Limitations & Exclusions

When MIA becomes true, the present value will hold the last valid present value
The driver Server side provides a limited emulation of a DMWPL-LZC. The driver does not emulate the response of a DMWPL-RP.

Appendix B.1. Driver Error Messages

Some messages are only printed once and then suppressed for subsequent occurrences of the same message. When the driver prints these messages they are indicated with a * after the error number.

Error Message	Explanation
mWATT:#1 Err. Illegal Node_ID [%d] - Set to 1˜	The node ID field is not appropriately set
mWATT:#2 Err. Repeater id is not defined˜	The repeater id was not found for CDO store
mWATT:#3 Err. Array storage type not defined˜	The type of storage, (flat/collated) was not found
mWATT:#4 Err. Object id not defined˜	Object id was not found for CDO store
mWATT:#5 Err. Object type not defined˜	Object type was not found for CDO store
mWATT:#6 Err. Object name	Object name was not found for CDO store

˜ Download the CSV file to the FieldServer, make the changes, upload the file and then reset the FieldServer for the changes to take effect.

Error Message	Explanation
not defined~	
mWATT:#7 Err. Relinquish not defined~	Relinquish not found for CDO store
mWATT:#8 Err. Active text not defined~	Active text not found for CDO store
mWATT:#9 Err. Inactive text not defined~	Inactive text not found for CDO store
mWATT:#10 Err. Units not defined~	Units not found for CDO store
mWATT:#11 Err. Command not defined~	Command entry not found
mWATT:#12 Err. Cmd1 not defined~	Cmd1 entry not found
mWATT:#13 Err. Cmd2 not defined~	Cmd2 entry not found
mWATT:#14 Err. Cmd3 not defined~	Cmd3 entry not found
mWATT:#15 Err. Cannot store length %u in da %s ~	CDO store overflows Data Array size.
mWATT:#16 FYI. Possible corrupt message. Client side parse unsuccessful - message Ignored. ∞	A message was found that did not match the driver specifications for a complete message to be parsed.
mWATT:#17 Err. Cannot store length %u in da %s	One of the "flat" arrays is not long enough for the storage requirements, check the csv for correct lengths and offsets; address is not a referenced field
mWATT:#18 Err. Attempt to read past end of Data Array! %s	Server side attempting to respond to a read command and the request is larger than the flat Data Array - check the declared Data Array sizes and offsets. Offsets should be zero in general for this driver.
mWATT:#19 Err. MD command value - check csv ~	Entry not found, check CSV command fields
mWATT:#20 Err. MD cmd1 value - check csv~	Entry not found, check CSV cmd1 fields
mWATT:#21 Err. MD cmd2 value - check csv~	Entry not found, check CSV cmd2 fields
mWATT:#22 Err. MD cmd3 value - check csv~	Entry not found, check CSV cmd3 fields
mWATT:#23 Err. do diagnostic 1∞	A developer diagnostic was invoked.
MWATT:#24 Err. do diagnostic 2∞	
mWATT:#25 Err. do diagnostic 3∞	
mWATT:#26 Err. MD store_typ value - check csv~	Entry not found, check CSV store_typ fields
mWATT:#27 Err. do diagnostic 4∞	A developer diagnostic was invoked.
mWATT:#28 Err. Illegal Map Descriptor length - defaulting to 1	Entry not found, check CSV length field

∞ Upload your configuration file(s), take a log and then call support

Error Message	Explanation
mWATT:#29 Err. Flag not defined	Entry not found, check CSV flag fields
mWATT:#30 Err. Cant open file={%s} for writing.∞	Contact technical support
mWATT:#31 Err. Cant close file={%s} for writing.∞	
mWATT:#32 Err. Lo scale not defined~	The Data Array low scale was not found for scaling values
mWATT:#33 Err. High scale not defined~	The Data Array high scale was not found for scaling values
mWATT:#34 Err. Message Lo scale not defined~	The message low scale was not found for scaling values
mWATT:#35 Err. Message High scale not defined ~	Check CSV - the message high scale was not found for scaling values
mWATT:#36 Err. do diagnostic 5∞	A developer diagnostic was invoked.
mWATT:#37 Err. do diagnostic 6∞	
mWATT:#38 Err. do diagnostic 7∞	
mWATT:#39a* Err. Cmd Code 'S' on Write Through.∞	An internal error occurred.
mWATT:#40 Err. Write Through not recognized.∞	A remote BACnet Client wrote to one of the objects on the FieldServer and the driver attempted to create a message to write the command/set point to the DMWPL-LZC. This operation could not be completed.
mWATT:#41. Err. Write Through. Bad Operating Mode=%d	A remote BACnet Client wrote a new operating mode to the FieldServer and the value of the Multi State variable was out of bounds so the driver could not interpret the action to perform. Correct the configuration or command set of the remote BACnet Client.
mWATT:#42 FYI. Write thru on MD=%s ignored.	In application specific operation, writing to some of the Data Objects produces no write to the DMWPL-LZC because no action has been specified. Refer to Appendix A.4 for further information
mWATT:#43 FYI. No Write when operating mode=3 (Maint).	Although it is permitted for a remote BACnet Client to write to the OperatingMode, no action has been defined when the mode is 'Maintenance'. In this case the no command is sent to the DMWPL-LZC.
mWATT:#50. Err. Cant get Pv. Err=%d∞	Write to the DMWPL-LZC failed because the driver could not obtain the present value from the object.
mWATT:#53 FYI. ASO Operation. Lutron_DMWPL->BACnet_Ethernet♣.	The DMWPL driver is operating in Application Specific Operation and the data read from the DMWPL-LZCs is automatically provided as a number of BACnet objects available to a BACnet Client using the BACnet Ethernet protocol.
mWATT:#54 FYI. ASO Operation. Lutron_DMWPL->BACnet_IP♣.	
mWATT:#55 Err. One BACnet Node required. ~	For application specific operations you may only define one node in the configuration file for the BACnet Server. The driver has found more than one. Check the CSV file.

♣ This message is printed for your information only and can be safely ignored if it reports an operation you expect.

Error Message	Explanation
mWATT:#56 MD Name too long ∞	An internal error has occurred.
mWATT:#57 Err. There are no DMWPL -LZCs.	For Application specific operation at least one DMWPL node must be defined.
mWATT:#59. FYI. Cmd Codes Unknown. =%d %d ~	The driver found a MapDesc whose command code it could not recognize. Check CSV command and cmd1.
mWATT:#60a Err. Heading not equal to keywords∞	An internal error has occurred.
mWatt:#67 FYI. Unexpected Route=%d. Normally 4,5 or 6. =>BACnet node=%d	When the application="Lutron-to-Virtual-BACnet-IP" or "Lutron-to-Virtual-BACnet-Ethernet" one BACNet node gets created for each DMWPL node. The node number is based on the “Route” parameter in the configuration file, normally 4, 5 or 6. A different route has been specified. The driver will work with the setting specified but draws your attention by printing this message. If your setting is correct then take no further action. Otherwise, edit the configuration file. Further information is provided in Appendix A.2

Appendix B.2. Driver Statistics

The following table identifies statistics generated by the Lutron DMWPL serial driver and their meanings.

Driver Statistics Recorded	Explanation
PLC Read Messages sent	Number of messages of STATUS or POWER request messages sent from the Client side
PLC Write Messages sent	Number of write command messages sent from the Client side
PLC Bytes sent	Total number of bytes sent by the Client side driver
PLC Message received	Total number of messages of all types received (ACK, NAK, error response, normal response) on the Client side driver.
PLC Bytes received	Total number of bytes received by all message types on the Client side driver.
Scada Messages sent	Number of response messages sent by the Server side driver
Scada Bytes sent	Total number of bytes sent by the Server side driver
Scada Messages received	Number of messages received on the Server side.
Protocol	Protocol error, the message is recognized but does not take a valid form, or any other messaging error that does not fall into a specific category.
IC_Timeout	Incoming message buffer experiences an excessive delay between incoming bytes
Timeout	Client side messages that were not responded to by a Server. Possibly a communications error or a command was made to a unit that does not exist.
Streaming	Message size was too large for the output buffer. This should never occur with the DMWPL protocol.
NAK	Message was received but could not be interpreted.
Station	A message with incorrect unit identification responded.
Checksum	A message was received but the message checksum did not match the calculated checksum. The likely cause is a corrupted message.
Bad Address	There was no appropriate Map Descriptor available to respond to an incoming message on the driver's Server side

THIS PAGE INTENTIONALLY LEFT BLANK