Lutron Integration Protocol Ethernet Driver Using Telnet as Transport Layer FS-8705-48

Chipkin - Enabling Integration

salesgroup1@chipkin.com

Tel: +1 866 383 1657

Driver Version: 0.0.1
Document Revision: 2

TABLE OF CONTENTS

1 Description

This is an Ethernet Driver that establishes a telnet connection to the Lutron controllers and devices. Using this connection commands to the lighting system can be issued and status and monitoring data can be read from the Lutron lighting system.

This driver does not support the Lutron serial interface for 3rd Party Integration. If you need it talk to our sales department.

The driver is an active client driver. It initiates communication transactions. Data sent to the Lutron system can be connected to data objects of other protocols such as BACnet, Modbus and over 120 more protocols. Data read from Lutron is exposed to remote clients such as BMS or CMS using data onjects of protocols like BACnet, Modbus and over 120 more.

The driver cannot be used to simulate a Lutron controller because only the client side has been implemented.

Max Nodes Supported

FIELDSERVER MODE	NODES	COMMENTS
Active Client	10	*A max of 10 Lutron connections per gateway*

Former Driver Type

Ethernet using Telnet to transport Lutron protocol.
Active Client

Compatibility Matrix

FIELDSERVER MODEL	COMPATIBLE WITH THIS DRIVER
All legacy products (FS2010/4010/3510)	Yes,
All current products as at July 2020	Yes,
EZ Gateways, QuickServer, Quickserver classic, Multiport Gateways	Yes,

Devices Tested

DEVICE	TESTED (FACTORY, SITE)
	2020 July, USA location with Lutron present at the site.

Not Supported – Driver is Future Proof

This driver has been written in such a way that should Lutron add new servicers, the driver will be capable of supporting them. The only changes required will be confirmation ones. The driver is capable of sending almost all asci strings and can load some or all of the string with live data.

If a response contains non numeric data the driver does not store the non-numeric field.

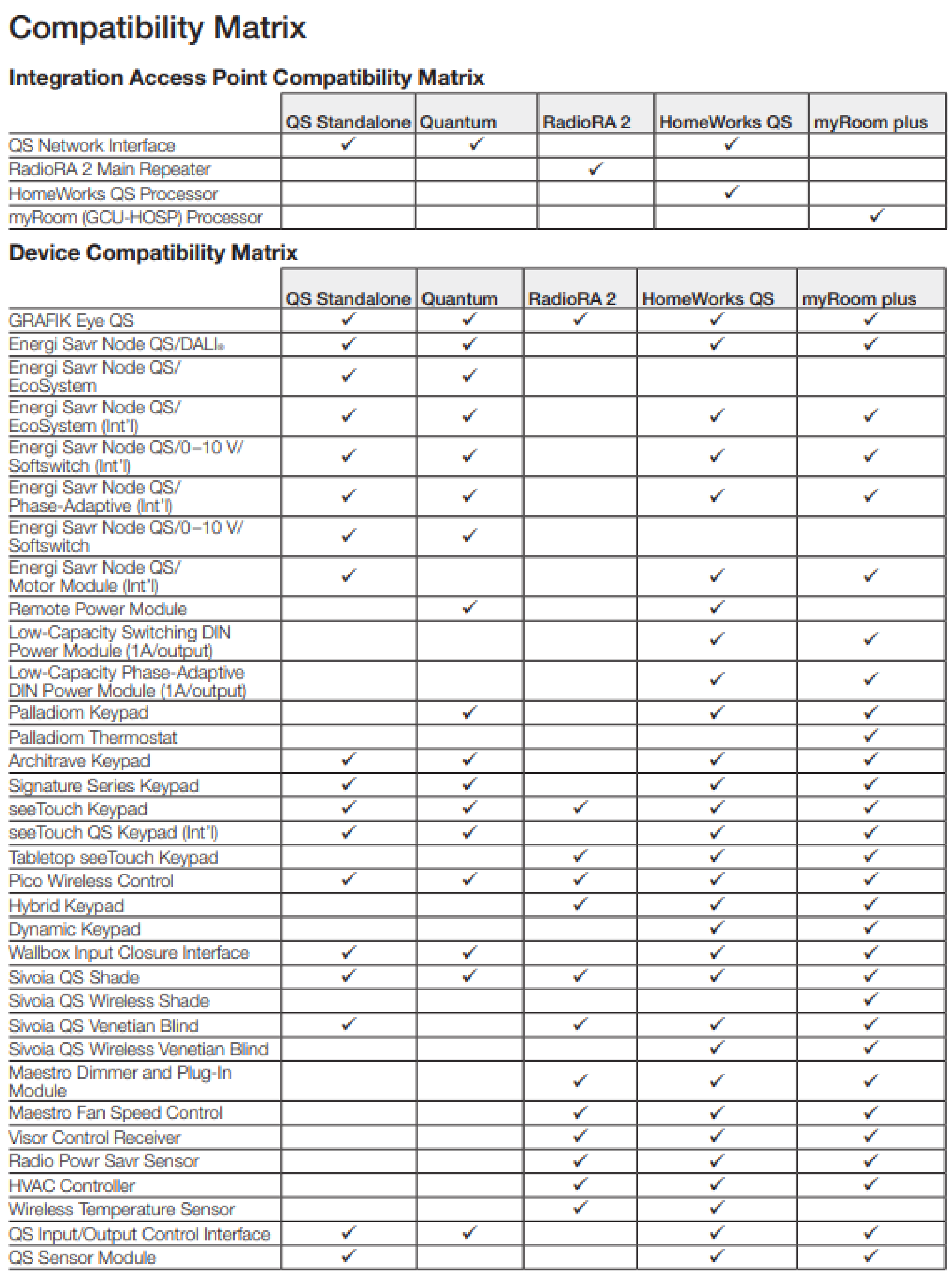
~Help is not supported

Services Supported

All options of Command, Query and Responses of each service supported

SERVICE
device
error
output
system
Integrationid
details
programming
group
timeclock
monitoring
area
Shadegrp
ethernet
sysvar
Hvac
reset

2 Compatibility



Compatibility Matrix

Integration Access Point Compatibility Matrix

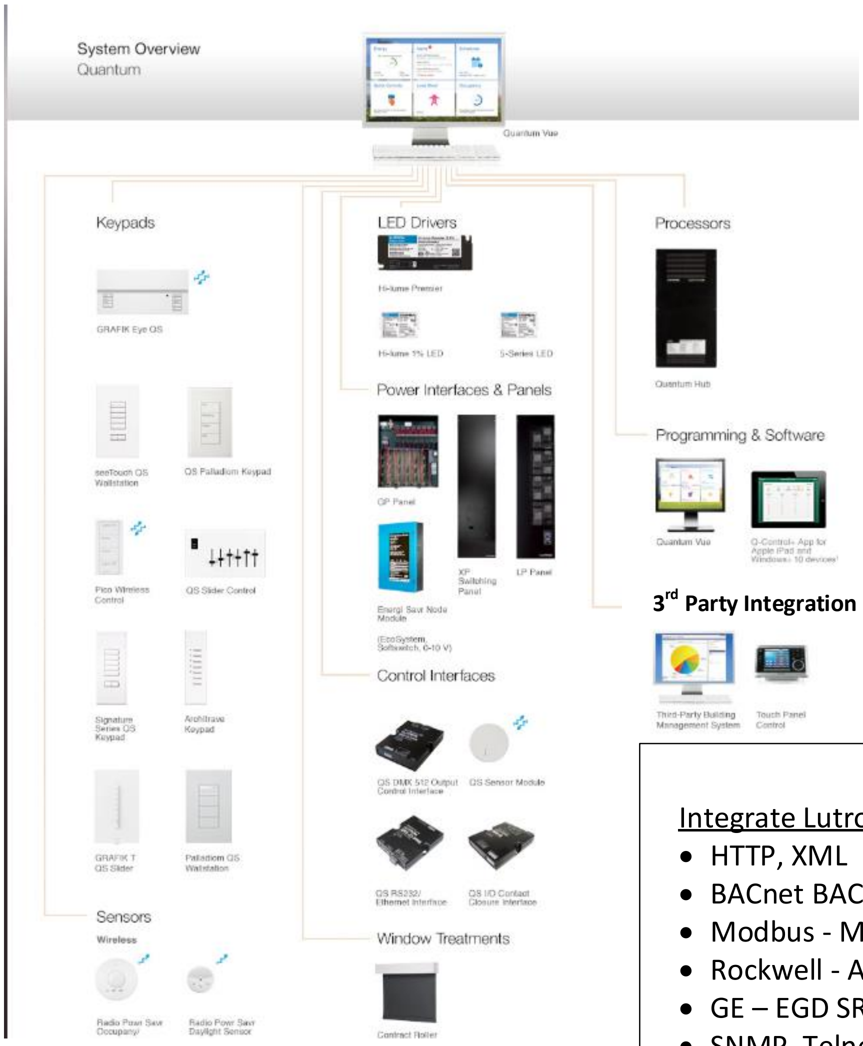
	QS Standalone	Quantum	RadioRA 2	HomeWorks QS	myRoom plus
QS Network Interface	✓	✓		✓	
RadioRA 2 Main Repeater			✓		
HomeWorks QS Processor				✓	
myRoom (GCU-HOSP) Processor					✓

Device Compatibility Matrix

	QS Standalone	Quantum	RadioRA 2	HomeWorks QS	myRoom plus
GRAFIK Eye QS	✓	✓	✓	✓	✓
Energi Savr Node QS/DALI®	✓	✓		✓	✓
Energi Savr Node QS/ EcoSystem	✓	✓			
Energi Savr Node QS/ EcoSystem (Int'l)	✓	✓		✓	✓
Energi Savr Node QS/0–10 V/ Softswitch (Int'l)	✓	✓		✓	✓
Energi Savr Node QS/ Phase-Adaptive (Int'l)	✓	✓		✓	✓
Energi Savr Node QS/0–10 V/ Softswitch	✓	✓			
Energi Savr Node QS/ Motor Module (Int'l)	✓			✓	✓
Remote Power Module		✓		✓	
Low-Capacity Switching DIN Power Module (1A/output)				✓	✓
Low-Capacity Phase-Adaptive DIN Power Module (1A/output)				✓	✓
Palladiom Keypad		✓		✓	✓
Palladiom Thermostat					✓
Architrave Keypad	✓	✓		✓	✓
Signature Series Keypad	✓	✓		✓	✓
seeTouch Keypad	✓	✓	✓	✓	✓
seeTouch QS Keypad (Int'l)	✓	✓		✓	✓
Tabletop seeTouch Keypad			✓	✓	✓
Pico Wireless Control	✓	✓	✓	✓	✓
Hybrid Keypad			✓	✓	✓
Dynamic Keypad				✓	✓
Wallbox Input Closure Interface	✓	✓		✓	✓
Sivoia QS Shade	✓	✓	✓	✓	✓
Sivoia QS Wireless Shade					✓
Sivoia QS Venetian Blind	✓		✓	✓	✓
Sivoia QS Wireless Venetian Blind				✓	✓
Maestro Dimmer and Plug-In Module			✓	✓	✓
Maestro Fan Speed Control			✓	✓	✓
Visor Control Receiver			✓	✓	✓
Radio Powr Savr Sensor			✓	✓	✓
HVAC Controller			✓	✓	✓
Wireless Temperature Sensor			✓	✓	
QS Input/Output Control Interface	✓	✓		✓	✓
QS Sensor Module	✓			✓	✓

3 Connection Configuration

Lutron Quantum System Overview

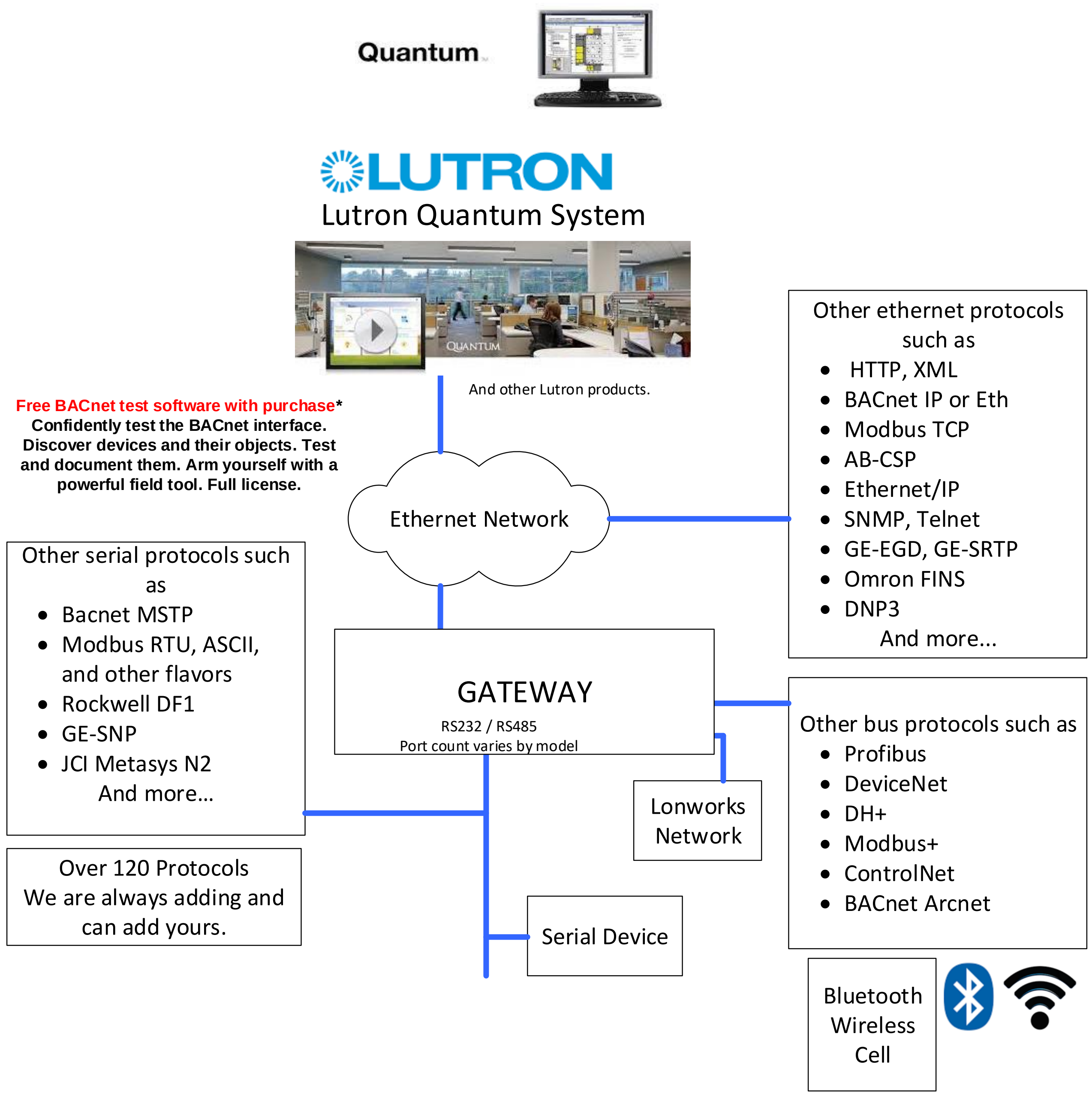


Integrate Lutron with FieldServer

- HTTP, XML
- BACnet BACnet IP MSTP Eth
- Modbus - Modbus TCP / RTU / Jbus
- Rockwell - AB-CSP, Ethernet/IP, DF1
- GE – EGD SRTP SNP SNPx
- SNMP, Telnet
- Omron FINS
- DNP3 – Ethernet and Serial
- Mbus, KNX, Profibus
- **Over 120 Protocols.**

4 Block Diagram

Multiple upstream protocols and connection supported. See list of FieldServer Drivers.


Quantum
LUTRON
Lutron Quantum System
QUANTUM
And other Lutron products.
Free BACnet test software with purchase*
Confidently test the BACnet interface. Discover devices and their objects. Test and document them. Arm yourself with a powerful field tool. Full license.
Ethernet Network
Other ethernet protocols such as
HTTP, XML
BACnet IP or Eth
Modbus TCP
AB-CSP
Ethernet/IP
SNMP, Telnet
GE-EGD, GE-SRTP
Omron FINS
DNP3
And more...
Other serial protocols such as
Bacnet MSTP
Modbus RTU, ASCII, and other flavors
Rockwell DF1
GE-SNP
JCI Metasys N2
And more...
GATEWAY
RS232 / RS485
Port count varies by model
Other bus protocols such as
Profibus
DeviceNet
DH+
Modbus+
ControlNet
BACnet Arcnet
Lonworks Network
Over 120 Protocols
We are always adding and can add yours.
Serial Device
Bluetooth
Wireless
Cell

5 Communication Settings

Default destination port is 23. May be changed by means of configuration.

Driver expects telnet server to be

- Require authentication – user and password
- Not be encrypted
- Must return a prompt after it has accepted a command or after it has sent a response.
- The prompt must be alphanumeric and may contain some human reasable symbols eg. QEST>

6 How the Driver Works

The driver allows you to configure multiple tasks.
Each task may have one service specified.
In specifying the service a place holder for live data can be specified.
The service terminating chars can be specified. Normally they are CRLF

Example – fixed definition

Task is defined as: ?output,100123
Driver sends ?output,100123CRLF
Controller responds: ~output,11,22,33
Driver stores 11 22 and 33 in 3 consecutive location specified in the configuration. These locations are typically mapped onto the 'other' protocol such as a set of BACNet objects or Modbus registers.

Example – use live data

Task is defined as: ?output,<DA_Int_Ids[0]>
Driver replaces <…> with the value it extarcts from the Dasta Array named 'DA_Int_Ids' at location 0. Lets say it find the value 555.

Driver sends ?output,**555**CRLF
Controller responds: ~output,11,22,33
Driver stores 11 22 and 33 in 3 consecutive location specified in the configuration. These locations are typically mapped onto the 'other' protocol such as a set of BACNet objects or Modbus registers.

A task may use multiple live data points.

Example

Task is defined as '#device 1,2,3
Driver sends that string with the terminator. If data is returned then each field separated by a comma is stored in consecutive locations

Eg driver sends #device 1,2,3CRLF
Lighting Controller responds ~ERROR,999
Driver stores 999 in configured location.

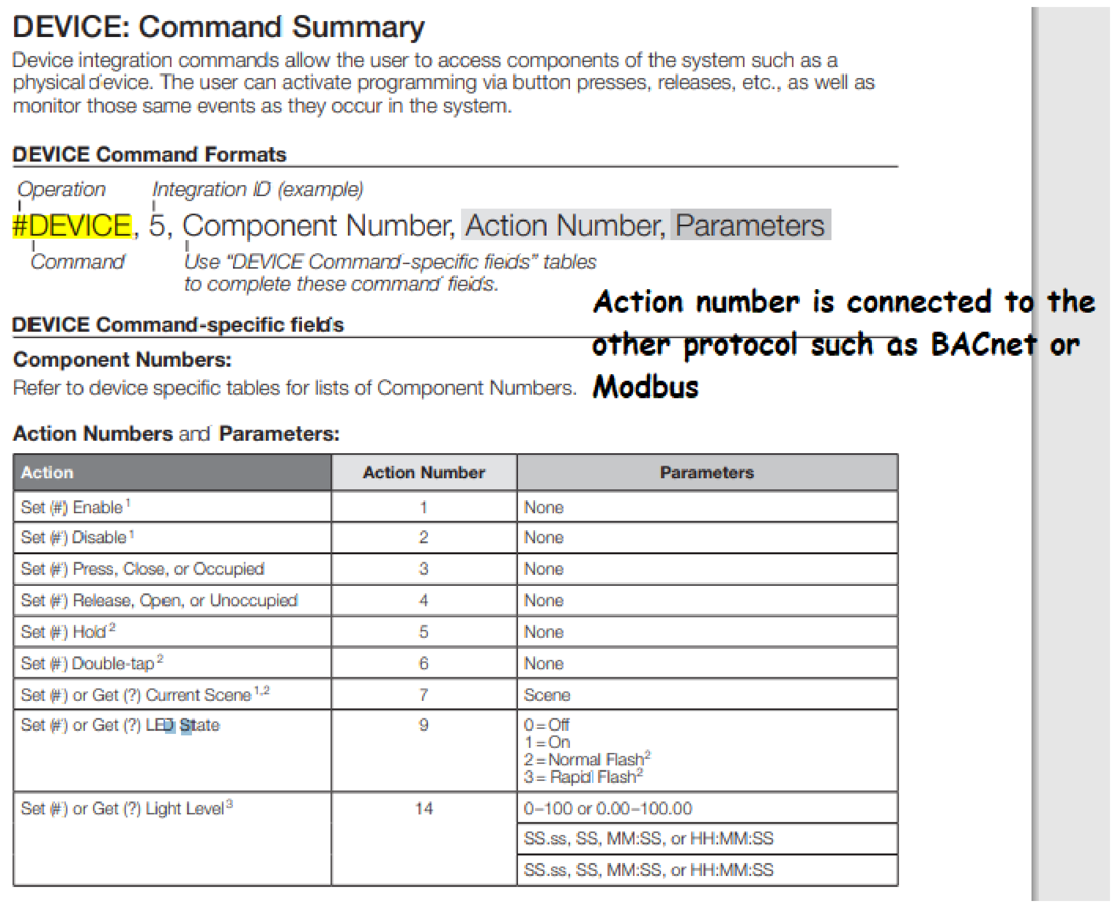
7 How the Driver Works – Multinode Version

This version is only capable of sending #device constant,other constant,live data value

Eg. #device 1,2,x   where x is the value found in configured data location.

Example : Device command.

The driver expects the IntegrationID and component numbers for each target of the command to be constant and part of the configuration. The action number is live data. The action number is connected to the other protocol such as Modbus or BACnet. When they update the data in the gateway, the gateway takes the updated value, uses it as the action number in sending the #device command.



DEVICE: Command Summary

Device integration commands allow the user to access components of the system such as a physical device. The user can activate programming via button presses, releases, etc., as well as monitor those same events as they occur in the system.

DEVICE Command Formats

Operation *Integration ID (example)*

#DEVICE, 5, Component Number, Action Number, Parameters

Command *Use "DEVICE Command-specific fields" tables to complete these command fields.*

Action number is connected to the other protocol such as BACnet or Modbus

DEVICE Command-specific fields

Component Numbers:
Refer to device specific tables for lists of Component Numbers.

Action Numbers and Parameters:

Action	Action Number	Parameters
Set (#) Enable[1]	1	None
Set (#) Disable[1]	2	None
Set (#) Press, Close, or Occupied	3	None
Set (#) Release, Open, or Unoccupied	4	None
Set (#) Hold[2]	5	None
Set (#) Double-tap[2]	6	None
Set (#) or Get (?) Current Scene[1,2]	7	Scene
Set (#) or Get (?) LED State	9	0 = Off 1 = On 2 = Normal Flash[2] 3 = Rapid Flash[2]
Set (#) or Get (?) Light Level[3]	14	0–100 or 0.00–100.00
		SS.ss, SS, MM:SS, or HH:MM:SS
		SS.ss, SS, MM:SS, or HH:MM:SS

8 Revision History

This table summarizes the update history for this document. Please contact Chipkin for an updated version of this document if required.

DATE	RESP	DOC. REV.	COMMENT
25 Nov 2020	ACF	1	Created initial document
2 June 2020	YC	2	Updated document format

Appendix A. Configuration

Connection Parameters

Caslutron_LanPort :
Default is 23.
May be assigned any positive integer.

Node Parameters

Caslutron_Prompt : This is the prompt the telnet server sends when it has completed an action and is ready for the next. The driver will use this setting to work out when a response message is complete.

Default = QNET>

Map Descriptor Parameters

Caslutron_CMD :

Specify the service and its parameters. Use a semi-colon to separate fields.

Eg.
Caslutron_CMD=”#device,1,2,3”
Driver sends =”#device,1,2,3CRLF”

The driver allows you to configure multiple tasks.
Each task may have one service specified.
In specifying the service a place holder for live data can be specified.
The service terminating chars can be specified. Normally they are CRLF

Example – fixed definition
Task is defined as: ?output,100123
Driver sends ?output,100123CRLF
Controller responds: ~output,11,22,33
Driver stores 11 22 and 33 in 3 consecutive location specified in the configuration. These locations are typically mapped onto the ‘other’ protocol such as a set of BACNet objects or Modbus registers.

Example – use live data
Task is defined as: ?output,<DA_Int_Ids[0]>
Driver replaces <…> with the value it extarcts from the Dasta Array named ‘DA_Int_Ids’ at location 0. Lets say it find the value 555.

Driver sends ?output,**555**CRLF
Controller responds: ~output,11,22,33
Driver stores 11 22 and 33 in 3 consecutive location specified in the configuration. These locations are typically mapped onto the ‘other’ protocol such as a set of BACNet objects or Modbus registers.

A task may use multiple live data points.

Example

Task is defined as '#device 1,2,3

Driver sends that string with the terminator. If data is returned then each field separated by a comma is stored in consecutive locations

Eg driver sends #device 1,2,3CRLF
Lighting Controller responds ~ERROR,999
Driver stores 999 in configured location.

Caslutron_Terminator :

One or more hex bytes which are added to the end of the command you specify with the CMD .
Should be set to '0x0d0x0a' meaning CRLF

Caslutron_Comma :

When a CMD is specified it requires the use of comma's to separate the field. The configuration file also uses comma's. This causes a conflict. For this reason you should specify the command using a ';' instead of a comma. Then when the driver executes the command it will replace the ';' with a comma. Actually you can use any character as the place holder for the comma

Specified in hex format.

This configuration files uses comma's to separator fields therefore we cant use a comma in the caslutron_CMD string

We have to use another char

eg semi colon	;	specify as 0x3b
colon	:	0x3a
dash	:	0x2d

if you want to send this command

#device 1,2,3

Then specify caslutron_CMD = #device 1;2;3 AND set casLutron_Comma = 0x3b

Caslutron_ResponseDA
Caslutron_ResponseDA_offset

If you execute a command that has response data this is how you specify where the response data will be stored.

Specify the name of a Data Array created in the Data Array section of the config.
The offset it the starting offset.

Eg.
Caslutron_ResponseDA = DA_STATUS
Caslutron_ResponseDA_offset = 10

Lets say you send a ?AREA command that has the following response

~AREA,2,6,3

Then the driver will store information about the success of the operation

Offset = 10	1=command failed. 0=Ok
Offset = 11	zero or error number
Offset = 12	Number of extracted data params
Offset = 13	First location used to store data from the response

So for this example

DA_STATUS[0] = 0 No Error
DA_STATUS[1] = 0 Err Num
DA_STATUS[2] = 4 The number of fields separated by a comma
DA_STATUS[3] = 0 The word ~AREA converted to integer = 0
DA_STATUS[4] = 2
DA_STATUS[5] = 6
DA_STATUS[6] = 3

Appendix B. Example Configuration

```
//================================================================================
//
//    Common Information
//

Bridge
Title
LutronIntegration1 testing

//================================================================================
//
//    Data Arrays
//

Data_Arrays
Data_Array_Name , Data_Format , Data_Array_Length
DA_Action_Nums  , UINT16      , 2000
DA_DATA         , UINT16      , 2000
caslutron_users   , BYTE        , 1000
caslutron_stats   , BYTE        , 1000
caslutron_Debug   , UINT16      , 1000

DA_response       , UINT16      , 1000
DA_response2      , UINT16      , 1000


// Offset 0 and 10 are used. Maxof 9 chars. The 10th must be a zero.

Preloads

Data_Array_Name ,Preload_Data_Format ,Preload_Data_Index ,Preload_Data_Value

caslutron_users   ,String             ,0                ,admin
caslutron_users   ,String             ,10               ,01F55ADE
//================================================================================
//
//    Client Side Connections
//
//    Default Port = 23
//
//
Connections
Adapter , protocol  , Poll_Delay , caslutron_LanPort
N1      , CasLutron , 0.05s      , 23


//================================================================================
//
//    Client Side Nodes
//
//
//    Node_ID is commonly used but is not used by this driver.
//    It can be used to monitor online/offline (See node_status in Configuration Manual)
//

Nodes
Node_Name , IP_Address    , Node_ID , Protocol      , Adapter
Lut01     , 192.168.1.174 , 1       , CasLutron     , N1

//================================================================================
//
//    Client Side Map Descriptors
//
//    <> Angle braces
//            Expect the contents to be the name of a Data Array defined earlier in the config
//            and a valid offset into that DA
//            Driver extracts value from that location and replaces the angle brace string with the numeric value.
//            eg. #device <DA_ID[0]>
//                  lets say the value found in DA_ID[0] is 999.
//            eg. #device <DA_ID[0]>
//               becomes #device
//               #device 999
//            Take care to have valid format
//
//
//    Terminator
```

```
//
//
//   Terminator
//           May be multiple chars
//           Specfied in hex format 0xnn where nn is the hex value of the charachter
//
//           CR = 0x0d
//           LF = 0x0a
//           If you want to terminate with CRLF then specify termination as 0x0d0x0a
//           Take care to have valid format
//
//
//   comma separator
//           This configuration files uses comma's to separater fields
//           Therefore we cant use a comma in the caslutron_CMD string
//           We have to use another char
//           eg semi colon   ;    specify as 0x3b
//              colon      :           0x3a
//              dash       :           0x2d
//
//           if you want to send this command
//
//           #device 1,2,3
//
//           Then specify  caslutron_CMD = #device 1;2;3   AND set casLutron_Comma = 0x3b
//
//
//


Map_Descriptors
Map_Descriptor_Name       , Scan_Interval , Data_Array_Name , Data_Array_Offset , Function , Node_Name , ResponseDA  , ResponseDA_offset  ,Length  ,Comma ,Terminator  ,CMD          ,

Set Device  - task 1      , 0s        , DA_Control    , 1           , wrb     , Lut01    , DA_response        , 0                   ,1     ,            ,                ,#device,<DA_Id[1]>;<DA_Comp_Nums[2]>;<DA_Action_Nums[3]>;

Set Device  - task 2      , 0s        , DA_Control    , 2           , wrb     , Lut01    , DA_response2       , 10                  ,1     ,0x3b        ,0x510x510x510x51    ,#device,<DA_ID[1]>;<DA_Comp_Nums[3]>;<DA_Action_Nums[4]>;

Set Device  - task 4      , 0s        , DA_Control    , 3           , wrb     , Lut01    , -              , 0                   ,1     ,0x3b        ,0x0d            ,#device;1;2;3

Set Device  - task 4      , 0s        , DA_Control    , 4           , wrb     , Lut01    , -              , 0                   ,1     ,            ,                ,#device;2;<DA_Comp_Nums[3]>;3

Set Device  - task 5      , 0s        , DA_Control    , 5           , wrb     , Lut01    , -              , 0                   ,1     ,0x3b        ,0x0d            ,#device 1;2;<DA_Action_Nums[2]>;




// The device number and the offset do not have to correspond
// Set Device  99          , 0s        , DA_Action_Nums  , 5           , wrbx    , Lut01    ,  1          , 11            , 1



//==============================================================================

//
```

Appendix C. Lutron Command Reference

Lutron
integration protocol

revision Y 2 April 2018

Google “Chipkin Lutron Integration Protocol” to find this document. (or a more current version)

Support

Please contact Chipkin Automation Systems directly for driver support.