Specification version 1

1 Overview

The light server process serves as the interface between multiple light client processes and the dimming hardware. It provides a means of combining lighting data from the clients to produce a single set of channel levels. It is also responsible for regularly sending the final lighting data to the dimmers.

1.1 Definitions and conventions

A **dimmer update** is defined as the process of sending a complete set of channel levels to the dimmers.

A range of values for an integral data type is given in the form:

[min, max]

where min is minimum value and max is the maximum value that can be represented by the type.

Hexadecimal values are written in C notation. For example, the decimal value 100 is written in hexadecimal as 0x64.

The type UInt16 refers to a 16-bit unsigned integer.
The type UInt32 refers to a 32-bit unsigned integer.

The word **shall** refers to behavior that the light server must implement in order to comply with this specification.

1.2 Versioning

Each revision of the software shall be identified by a version number consisting of a major and a minor version number.

The major version is equal to the version of this specification document which governs the software. This specification document corresponds to major version 1.

The minor version indicates the number of previous software revisions with the same major version. The first software revision for any major version shall be minor version 0.

1.3 Implementation requirements

The interface between the light server and the dimming hardware is determined by the dimming hardware, the hardware running the light server, and the operating system and libraries available to the light server. These factors are not discussed in this specification.

The light server shall communicate with clients via TCP and UDP over Ethernet using a 10Base-T or faster connection. The light server process shall be capable of running on dedicated hardware.

The light server shall perform dimmer updates at a rate of at least 60Hz. If the hardware is capable of supporting it, the update rate should be at least 120Hz. The update rate should not significantly exceed 120Hz because the dimmer hardware cannot update faster than this.

The light server shall be capable of receiving lighting data from all clients at a rate of at least 60Hz, except when limited by network bandwidth.

The light server shall support a minimum of 16 simultaneous client connections.

The light server shall support a minimum of 16 total layers. It shall support a minimum of 16 layers per client, subject to any limit on the total number of layers.

The light server shall support all ASCII characters from 32 to 126 inclusive in layer names. If a client provides a layer name containing an unsupported character, the light server shall replace each unsupported character with ASCII character 32 (space).

The light server shall support layer names of at least 16 characters.

Clients shall not send a text string including a NUL (ASCII 0) character. The light server shall replace any NUL characters in received strings with ASCII character 32 (space). The light server shall not send a NUL character in any text string.

2 Architecture

The light server combines lighting data from multiple clients by using a list of layers. Each client controls one or more layers. An ordered list of layers is maintained and is traversed before each dimmer update. As the list is traversed, the channel levels produced by each layer are fed into the next, where they may be modified. The input to the first layer is 0 for all channels. The output of the last layer is the final channel levels which are sent to the dimmers.

Channels are numbered consecutively from 0 to `(n - 1)`, where `n` is the total number of channels provided by the dimmers. All layers contain data for all channels.

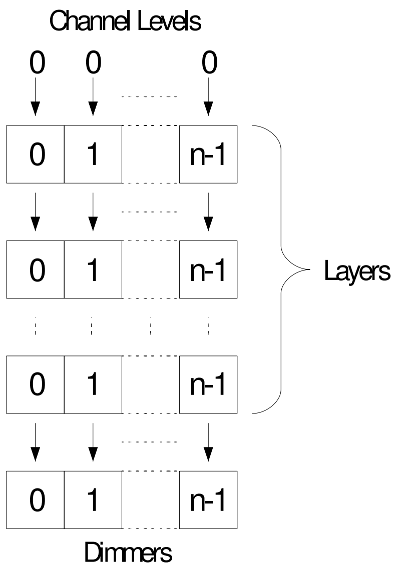

Figure 1. Flow of channel levels through layers.

2.1 Levels

A level is an integer on [0, 255] which represents the amount power delivered to the load on a dimmer channel. 0 corresponds to no power while 255 corresponds to full power. Actual power may not be a linear function of the level.

A level can be considered a fixed-point number where the real value is equal to:
```
Real_Value = Fixed_Value / 255
```
A real value of 0 corresponds to no power while 1 corresponds to full power.

The multiplication of two levels A and B to produce a resulting level is defined as follows:
```
Result = (A * B) / 255
```

2.2 Layers

A layer is the structure used by clients to control channel levels. Each layer consists of the following data:

- A layer ID
- A layer name
- Channel data for each channel

The layer ID shall be a 32-bit unsigned integer which shall be assigned when the layer is created. A layer's ID shall not change during the lifetime of the layer. No layer shall have an ID of zero. All layers which exist at the same time shall have different layer IDs. The light server may reuse the layer IDs of previously destroyed layers.

Channel data is described in section 2.3 below.

For each channel, a layer takes a level as input and produces a level as output. The operations performed on the input level are determined by the per-channel data described below.

2.3 Channel data

The data associated with each channel in a layer is:

- Scale
- Channel_Level
- Mode

Scale and Channel_Level are both levels. Mode is of an enumerated type which indicates how the output value for this channel is produced. Modes are defined as follows:

Mode Value	Channel Output
MAX	`Output = max(Input * Scale, Channel_Level)`
ADD	`Output = Input * Scale + Channel_Level`

The multiplication of levels shall be performed as defined in section 2.1.
The function `max(a, b)` shall return the greater of its two parameters.

2.4 Layer order

The layer order is an ordered list of layer names indicating the order in which those layer names shall appear in the layer list.

The special name `default` represents the position of all layer names which do not appear in the layer order. The name `default` shall always be present in the layer order. If a layer order is given without a `default` name, `default` shall be added to the beginning of the list.

2.5 Layer list

The layer list is an ordered list containing all layers. All layers with the same name shall appear consecutively in the layer list, in order of creation. The order of layers with different names shall be the same as the order of the names in the layer order.

3 Protocol

All data is transferred between a client and the light server in the form of packets. Packets may be sent on a TCP stream or individually in UDP packets.

TCP is used for packets which are typically infrequent but which must be delivered correctly and in order.
UDP is used for periodic updates which are sent frequently but which can be lost with little adverse effect.

All integers greater than 8 bits in size shall be sent in little-endian byte order (least significant byte first).

The light server shall listen for client control connections on a TCP port. The port number should be 19607. The light server may provide multiple TCP ports for control connections.

The light server should listen for client update packets on a UDP port. If a UDP update port is available, the port number shall be provided to the client in the connection information packet described in section 3.2. The light server may use multiple UDP ports and may provide a UDP port only to certain clients.

A client may close its control connection immediately after receiving the connection information packet if no UDP update port is provided by the light server.

3.1 Packet structure

All packets shall begin with the following header:

Offset	Type	Description
0	Byte	Signature
1	Byte	Packet type
2	UInt16	Packet length

The signature byte shall always have the value 0x5A.
The packet length shall be the total length of a packet including this header. The minimum packet length is 4 bytes.

The packet type byte determines the meaning of the packet. The direction of a packet and the protocols over which it may be sent are indicated by the packet type as follows:

Packet Type	Direction	Protocol
0x00 - 0x3f	Client to server	TCP
0x40 - 0x7f	Client to server	TCP, UDP
0x80 - 0xbf	Server to client	TCP
0xc0 - 0xff	Server to client	UDP

The light server shall accept all types of client to server packets on a client's control connection. The light server shall accept packets with types 0x40 – 0x7f from a client on the UDP port given to the client in the client connection information packet (see section 3.2). The light server shall consider packets received on a UDP port with types 0x00 – 0x3f to be erroneous.

The following packet types are defined:

Packet Type	Description	Section
0x01	Create layer	4.1.1
0x02	Destroy layer	4.1.2
0x41 - 0x47	Update channel data	4.1.3
0x80	Connection information	3.2
0x81	Create layer response	4.1.1
0xbf	Error	3.4

All packet types not shown in the above table are reserved.

In response to some client to server packets, the light server sends a response to the client. In the event that the light server cannot successfully process the first packet, it shall either disconnect the client or send a packet indicating an error to the client as described in the section describing the first packet.

A client shall not send a packet to the light server on the control connection if it has already sent a packet for which the light server will return a response and the client has not received the response. If the light server receives such a packet, it shall either ignore the packet, disconnect the client, or process the second packet after it has sent the response to the first packet.

3.2 Connecting

A client connects to the light server by opening a TCP connection to the light server's control port. This connection is referred to as the control connection. The client is considered to be connected to the light server as long as this control connection is open.

A client may send certain packet types to the server via UDP to reduce latency. This UDP port, referred to as the update port, is given to the client in the client information packet described below.

The light server may refuse a control connection from a client for security reasons or if insufficient resources are available to support the client. In this case, the light server shall either not accept the connection, close the connection immediately upon accepting it, or send an error to the client (see section 3.4) and then close the connection.

Immediately after accepting a control connection, the light server shall send a connection information packet to the client on the control connection. The connection information packet has the following format:

Offset	Type	Description	Value
0	Byte	Signature	0x5A
1	Byte	Packet type	0x80
2	UInt16	Packet length	0x0008
4	Byte	Server's major version	1
5	Byte	Server's minor version	
6	UInt16	UDP update port	

If the UDP update port is 0, then no UDP update port is provided by the server.

If a client receives an erroneous connection information packet, a packet indicating an unsupported light server version, or if it detects that it cannot send to the given update port, it shall close the control connection without sending any data.

The client shall send packet types 0x40 – 0x7f only on the control connection and to the given UDP update port on the light server. The light server may assign different update ports to different clients.

3.3 Disconnecting

When a client's control connection to the light server is broken, all layers that were created by that client shall be destroyed.

A client shall not send any data to the update port after its control connection has been closed. The light server should ignore any packets received on an update port from a client whose control connection is no longer open.

3.4 Error handling

A packet shall be considered erroneous if any of the following is true:

- The packet has an incorrect signature byte.
- The packet type is not recognized or supported by the light server.
- The packet length is inappropriate for the packet type.
- Any other condition which is given in another section of this specification as causing the packet to be considered erroneous.

The correct packet length for each packet type is given in the packet details in section 4.

The light server may send an error packet in response to an erroneous packet from a client or if an error occurs during the processing of a valid packet for which a response would normally be sent. The error packet has the following format:

Offset	Type	Description	Value
0	Byte	Signature	0x5A
1	Byte	Packet type	0xbf
2	UInt16	Packet length	0x0004 + `len`
4	`len` bytes	Error message	

`len` is the length of the error message. The error message shall consist entirely of ASCII characters 32-126.

When the light server receives an erroneous packet, it shall either disconnect the client or perform no further processing of the erroneous packet.

If a client receives a packet from the light server when it is not expecting any packet, it shall treat the packet as erroneous.

A client which receives an erroneous packet may disconnect from the light server. If the client does not disconnect, the erroneous packet shall be ignored.

4 Operations

4.1 Layer operations

4.1.1 Create layer

Offset	Type	Description	Value
0	Byte	Signature	0x5A
1	Byte	Packet type	0x01
2	UInt16	Packet length	0x0004 + `len`
4	`len` bytes	Layer name	

The packet shall be considered erroneous if any of the following conditions are true:
- The length of the layer name is greater than the maximum length supported by the light server.
- The length of the name is zero.
- The name contains characters not supported by the light server.

When this packet is received, the light server shall create a new layer and insert it into the layer list in the correct position as described in section 2.5. The new layer shall have the name given by the client, subject to the limitations in section 1.3. For each channel in the new layer, the Scale parameter shall be 255, the Channel_Level parameter shall be 0, and the Mode parameter shall be MAX.

Dimmer updates shall not occur while any incompletely initialized layer is in the layer list.

If the creation of the layer would exceed any limit on the total number of layers or the number of layers for the client creating the layer, then an error packet shall be sent to the client.

If the layer is created successfully, the following packet shall be sent to the client:

Offset	Type	Description	Value
0	Byte	Signature	0x5A
1	Byte	Packet type	0x81
2	UInt16	Packet length	0x0008
4	UInt32	Layer ID	

The layer ID is the ID of the newly created layer.

4.1.2 Destroy layer

Offset	Type	Description	Value
0	Byte	Signature	0x5A
1	Byte	Packet type	0x02
2	UInt16	Packet length	0x0008
4	UInt32	Layer ID	

When this packet is received, the light server shall destroy the layer with the given layer ID. The packet shall be considered erroneous if no layer exists with the given ID.

4.1.3 Update channel data

Seven packet types are defined for allowing clients to set the parameters associated with each channel in a layer.

Offset	Type	Description	Value
0	Byte	Signature	0x5A
1	Byte	Packet type	0x41 - 0x47
2	UInt16	Packet length	
4	UInt32	Layer ID	
8	Byte	First channel	`f`
9	Byte	Number of channels	`n`
10	See below	Channel data	

The channels affected by this packet are `f` through `(f+n-1)`.

The packet is considered erroneous if any of the following is true:

- `(f+n-1)` is greater than the highest valid channel number.
- `n` is zero.
- No layer exists with the given layer ID.
- The packet length is not correct.

The least significant three bits of the packet type indicate which parameters are to be set by this packet:

Bit	Bit mask	Parameter to be set
0	0x01	Channel level

Bit	Bit mask	Parameter to be set
1	0x02	Scale
2	0x04	Mode

The channel data consists of an array of data for `n` channels for each parameter. The arrays are in order by bit number, starting with the least significant bit of the packet type.

The data for channel level and scale is sent as an array of bytes, with one byte for each channel.

Mode data is sent as bits packed into bytes, with the mode for channel `f` being the least significant bit of the first byte. Each bit is interpreted as follows:

Value	Mode
0	MAX
1	ADD

The number of bytes of mode data is `(n/8)`, rounded up to the nearest integer. When `n` is not a multiple of 8, the light server shall ignore the value of the extra bits.

The length of the packet shall be 10 plus the total number of bytes of data. For example, the longest packet (type 0x47) has a length of `(10+2*n+(n+7)/8)` (truncated to an integer value).

When this packet is received, the light server shall update the channels and parameters as indicated in the packet. No dimmer updates shall occur while the layer is being modified.