ASLS

Autonomous Stage Lighting System

Q PROTOCOL PROPOSAL

Contents

List of Figures

List of Tables

1 Preface

This document aims to specify the operation of ASLS's solution for universal show control. Please note that the elements discussed in this document must be kept in mind at all time during the development phase. Any changes that might occur after the validation of this document should be discussed with the team manager.

Scope

This document covers the following topics:

- Specification of the Q protocol
- Definition of node's features, and components
- Definition of a controller's features and components

Note: This document is specified using the latest UML specification (Version 2.5). For additional information please see documentation available at http://www.omg.org/spec/UML/2.5/PDF/

Target Audience

This document is intended to be used, reviewed and modified by members of the ASLS project only. Any modification to this document must be reported into the *Revision Information* section

1.1 Prerequisite

In this document, it is assumed that the reader is familiar with the following:

- Unified Modeling Language (UML)
- Digital & Analog electronics
- Ethernet LANs
- TCP protocol
- HTTP protocol

2 Introduction

2.1 Background of the solution

2.1.1 Background

Sound and Light shows have been entertaining millions of crowds around the world for decades. However, it is not until the late 70s that the technology involved in the design of such shows took a big leap forward to bring us the complex shows that we know today. Thanks to the now popular use of computerized control systems, it is possible to accurately cue[1] a large set of different effects to a specific time-code, thus offering accuracy and precision in the timing of a show.

Such shows can be divided into three obvious categories: pyrotechnics, stage lighting and more recently, video mapping. While each of these categories are commonly used individually one from another, there has recently been a major rise in the demand for shows mixing those three categories together to bring spectators a truly unique experience by offering a more in depth approach of the sound and light show. However, getting to mix and synchronize these three aspects of the Sound and light show is not an affordable nor easy task as it involve getting a specific type of controller for each individual category, which are often complex or impossible to synchronize together as protocols may vary from a controller to another.

While theme parks and other large organizations might have the time and budget to develop in-house case-specific solutions to get around these problems, no real standard allowing for truly reliable coordination of the many aspects of sound and light shows has yet been established. Methods of synchronization are limited and mostly consist in time-code cue generation or oral cues given over walkie talkie from an operator to another. This limitation in synchronization drastically increases error probabilities while greatly limiting the creativity of show designers.

2.1.2 Prior Art

Created in 1986 in the aim to standardize method for controlling light dimmers which employed various incompatible proprietary protocols, the DMX512 standard is broadly used to control stage lighting equipment and effects. However, its lack of automatic error checking and correction makes it not appropriate to control hazardous applications such as pyrotechnics. Still widely used nowadays for its undeniable efficiency, it is being slowly replaced with more convenient protocols allowing for transport of DMX512 data over IP such as Art-Net, a royalty-free communication protocol for transmitting DMX and RDM[2] protocol over UDP[3] or E1.31 / sACN[4] which is expected to become the new standard for transporting DMX over IP.

While standard EIA-485[5] differential signaling allows for the control of single universes per cables, encapsulation of DMX512-A protocol over IP allows for the control of many universes at a time. However, the amount of universes to be controlled depends directly on the bandwidth limitation of the physical layer used as a mean of transportation of the DMX512-A data. Sending DMX512-A (either over standard differential signaling or through more bandwidth capable protocol such as Ethernet) requires large bandwidth as to update one or many universes of fixtures live, thus causing great limitations in finding adequate wireless solutions for transporting high data rates over long distances.

Moreover, the lack of safety features and the specificity of the DMX512-A protocol to update channel values in bulk of 255 at a time causes safety concern as to use the protocol for controlling hazardous

[1] The trigger for an action to be carried out at a specific time
[2] Remote Device Management
[3] User Datagram Protocol
[4] Streaming Architecture For Control
[5] standard defining the electrical characteristics of drivers and receivers for use in serial communications systems

equipment such as pyrotechnics. As a cause for these issues, there is no standard for controlling pyrotechnic equipment. Firing systems available on the market each rely on their own proprietary protocols. While these protocols might differ quite drastically from one manufacturer to another, they must necessarily go by the following rules: Data sent from a controller to a node must be unbuffered, meaning that cues for a single node must be sent separately one from another in opposition to the DMX512 protocol. Data must be unbuffered in such a way that a push of a kill-switch terminates the firing sequence immediately. Communication between controller and nodes is bi-directional as to receive error messages and information about the status of the nodes.

While the display of videos over screens, buildings or dedicated three-dimensional sculptures has become more and more common over the past few years, nonetheless, it is still quite abstract as how to control it. While most softwares dedicated to video mapping and media triggering allow external cues to be triggered over MIDI or OSC[6], there is no real standard for controlling media cues.

Not to be confused with MSC[7], the MIDI protocol has seen its popularization greatly improving in the world of show control with the rise in popularity of USB MIDI controllers. Particularity useful for live cue triggering and parameter modification, MIDI controllers are only used to bring in more control features to systems which might lack knobs or buttons in the same way a keyboard and mouse helps users interact with computers. Similarly to MIDI, OSC is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology. with the recent breakthrough of IoT[8], an increasingly number of show controllers are built with the ability to communicate through OSC, thus allowing for cue trigger and control of specific element of a show from devices such as computer or mobile devices wirelessly through wifi or via Ethernet cables.

In an effort to enable show designers to link together and operate these different types of control system in a coordinated manner, a various set of show control technologies have been developed. However none of these technologies was ever considered to be set as a standard for universal show control, both because of the technical limitations that these encountered at the time that were being developed and because of the time and effort that would be involved into putting in place such logic to enable the control of various pieces of equipment. Specified by the MIDI Manufacturers Association in 1991, MSC goal was to solve this particular issue by offering a way for MIDI systems to communicate with and to control dedicated intelligent control equipment in theatrical, live performance, multi-media, audio-visual and similar environments. While it is now rarely used due to the lack of software support and bandwidth limitation, it is still common in larger static installation such as theme parks shows and parades.

Show control essentially relates to the act of triggering an event or a sequence of events as a mean to link together and operate multiple entertainment control systems in a coordinated manner. It is usually achieved through the generation of uniquely identifiable cues sent over a network of entertainment system to (re)trigger singular events. While the control of media and pyrotechnics relies essentially in single (re)triggerable cues, the control of stage equipment, more particularly stage lighting equipment is rather more complex, hence the lack of Show control protocols fully integrating the control of DMX apparatus.

In these respects, the autonomous system according to the present invention substantially departs from the conventional concepts and designs of the prior art, and in so doing provides an apparatus primarily developed for the purpose of providing a solution for communicating with and for controlling dedicated intelligent stage, multi-media and audio-visual equipment.

2.1.3 Scope

In view of the disadvantages inherent in the known types of show control solution now present in the prior art, the present system provides a new universal solution allowing communication and control of various equipment including lighting, pyrotechnics and medias such as audio or video through a single communication protocol.

[6]Open Sound Control
[7]Midi Show Control
[8]Internet Of Things

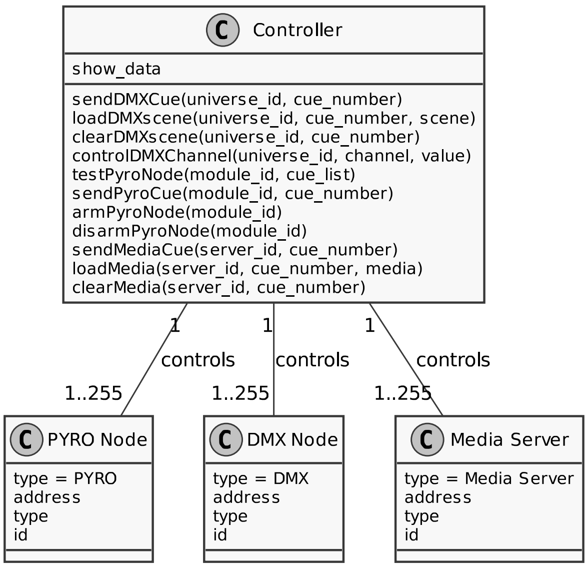
3 System Architecture

3.1 Aims

In an effort to bring a new dimension to show control, ASLS aims to offer its users new ways to control sound and light shows while trying to establish new standards for show control and automation. By bringing ready to use systems relying on its own open source universal show control protocol "Q" onto the market, ASLS wishes to popularize its solution and encourage manufacturers to build their new ranges of products around this particular protocol. In doing so, ASLS wishes to get its solution to be considered as a reliable standard option for universal show control within 2024.

3.2 Controller

Controllers consist of TCP servers communicating with nodes through the use of the Q protocol. According to the network topology discussed in section 5.2.1.1, controllers are able to communicate with up to 255 nodes of each preset type (DMX, PYRO, MEDIA SERVER).



Controller
show_data
sendDMXCue(universe_id, cue_number)
loadDMXscene(universe_id, cue_number, scene)
clearDMXscene(universe_id, cue_number)
controlDMXChannel(universe_id, channel, value)
testPyroNode(module_id, cue_list)
sendPyroCue(module_id, cue_number)
armPyroNode(module_id)
disarmPyroNode(module_id)
sendMediaCue(server_id, cue_number)
loadMedia(server_id, cue_number, media)
clearMedia(server_id, cue_number)
1
1
1
controls
controls
controls
1..255
1..255
1..255
PYRO Node
type = PYRO
address
type
id
DMX Node
type = DMX
address
type
id
Media Server
type = Media Server
address
type
id

Fig.4.1 Architecture of controller

3.2.1 Ethernet Implementation

3.2.1.1 Physical

The system is thought to operate over Ethernet either through a wired or wireless configuration. Network configuration is comprised of a controller (gateway) and nodes (hosts) routed one from another in a star topology. Every node is given a class B private static IP address which least significant two bytes are used for node type and node identification (*see 5.2.1.3*).

For shows in which wires are a constraint, communication between controllers and nodes can be established over wifi. To do so, the controller is put in Access Point mode and nodes may be associated to the network using WPS[1]. When wires are not a problem, communication between controllers and nodes can be established over Ethernet cables. If more than one node is being controlled routers and network switchs may be used to forward data to their destination.

3.2.1.2 Transport

The communication chosen for transport of the protocol over Ethernet is TCP. The packet formats defined within this document constitute the data field of en enclosing TCP packet. Each data item constituting a packet is considered to be an unsigned integer which subtype (uint8,uint16 or uint32) is determined by the number of bits of the item. extra bytes at the end of a valid packet will be ignored.

Unused positions contained within some data fields are reserved for future versions of the protocol. These bits should be transmitted as zero and should not be tested by the nodes. Whenever port number is not specified, data traffic should be defaulted to port 5214 (0x145E), default port for transporting Q packets.

3.2.1.3 Routing

Similar illy to Art-Net, this standard uses network unicast addressing in order to direct Q packets to their specific destination. Addressing and partitioning of unicast traffic is achieved by setting the least significant two bytes of the unicast B class IP address to the desired type and node address. While the type of node is defined by the manufacturer, the node address shall be defined by the user himself. This configuration allows for type-specific subnets, thus lowering the chances of getting packets routed to the wrong destination.

Node - IP mapping

IP Address Byte	Value
1	172 (Private IP address)
2	16 (Private IP address)
3	Node type (see table below)
4	Node Address

The sub-net mask shall always initialized to 255.255.0.0, unless a custom IP address is in use. This means that the network address is the most significant 16 bits and the Node address is the least significant 16 bits of the IP address.

Node - Type mapping

Node type	Value	Node Address
RESERVED	0x00	Controller
DMX	0x01	Universe id (0 - 253)
PYRO	0x02	Module id (0 - 254)
MEDIA SERVER	0x03	Media Server id (0 - 254)
UNUSED (FUTURE USE)	0x04 - 0xFE	UNDEFINED

[1] Wifi Protected Setup

3.2.2 Protocol definition

TCP packets sent by a controller should conform to the Q protocol standards as defined below. Any other packets should be ignored.

Q command packet definition

Field	Name	Size	Description
1	ID[]	8	Array of 8 characters, the final character is a null termination. Value = ’A’,’S’,’L’,’S’-’Q’ 0x00
2	Command	4	Node type specific command
3	Payload	-	The data payload to be sent for the specific command sequence

3.2.2.1 Command List

Command for Q protocol

Mnemonic	Hex Code	Type
CUE_START	0x00	General
CUE_PAUSE	0X01	General
CUE_RESUME	0X02	General
CUE_STOP	0X03	General
CUE_LOAD	0X04	General
CUE_CLEAR	0X05	General
ALL_OFF	0X06	General
RESTORE	0X07	General
CONTROL	0X10	Lights
TEST	0X20	Pyro
TEST_REPLY	0X21	Pyro

3.2.2.2 General Commands

The following commands are basic to the current implementation of Memory Lighting systems and probably apply to all dedicated theatrical show control systems in a general sense. Although it is not required that Controlled Devices incorporate all of these commands, it is highly recommended:

General commands for Q protocol

Mnemonic	Hex Code	Brief Description
CUE_START	0x00	Starts a cue playing
CUE_PAUSE	0X01	Pauses the playing of a cue
CUE_RESUME	0X02	Resumes the playing of a paused cue
CUE_STOP	0X03	Stops the playing of a cue
CUE_LOAD	0X04	Loads a cu-identified file
CUE_CLEAR	0X05	Clears a cue-identified file
ALL_OFF	0X06	Disable outputs
RESTORE	0X07	Enable outputs

3.2.2.3 General Commands Description

CUE_START

Sends a cue start message to the node. Payload for "CUE_START" command simply consist of an hexadecimal value within the interval [0x00 - 0xFF] refering to the ID the cue to be triggered. Command should be ignored if the payload is empty or out of bounds.

CUE_START Packet example

ID	COMMAND	PAYLOAD
'A','S','L','S','-','Q',0x00	0x00	*see table below*

CUE_START payload

CUE NUMBER
uint8

CUE_PAUSE

Sends a cue stop message to the node. Payload for "CUE_PAUSE" command simply consist of an hexadecimal value within the interval [0x00 - 0xFF] refering to the ID the cue to be stopped. Command should be ignored If the cue wasn't started in the first place or if the payload is empty.

Note: This command does not affect non retriggerable effect such as pyrotechnics and will be ignored by pyrotechnic nodes. Such effects cannot be stopped once they have been triggered.

CUE_PAUSE Packet example

ID	COMMAND	PAYLOAD
'A','S','L','S','-','Q',0x00	0x01	*see table below*

CUE_PAUSE payload

CUE NUMBER
uint8

CUE_RESUME

Causes stopped cue to continue running. Payload for "CUE_RESUME" command simply consist of an hexadecimal value within the interval [0x00 - 0xFF] refering to the ID the cue to be stopped. If no cue id is specified, all stopped cues will RESUME. Command should be ignored If the cue wasn't started in the first place.

Note: This command does not affect non retriggerable effect such as pyrotechnics and will be ignored by pyrotechnic nodes. Such effects cannot be stopped nor resumed once they have been triggered.

CUE_RESUME Packet example

ID	COMMAND	PAYLOAD
'A','S','L','S','-','Q',0x00	0x02	*see table below*

CUE_RESUME payload

CUE NUMBER
uint8

CUE_STOP

Sends a cue stop message to the node. Payload for "CUE_STOP" command simply consist of an hexadecimal value within the interval [0x00 - 0xFF] refering to the ID the cue to be stopped. Command should be ignored If the cue wasn't started in the first place or if the payload is empty.

Note: This command does not affect non retriggerable effect such as pyrotechnics and will be ignored by pyrotechnic nodes. Such effects cannot be stopped once they have been triggered.

CUE_STOP Packet example

ID	COMMAND	PAYLOAD
'A','S','L','S','-','Q',0x00	0x03	*see table below*

CUE_STOP payload

CUE NUMBER
uint8

CUE_LOAD

Cue-identified files can be streamed over the network and stored directly into a memory-enabled node using the The CUE_LOAD command. CUE_LOAD must be sent prior to the beginning of the show so that pre-programmed cues can be triggered using a CUE_START command.

CUE_LOAD Packet example

ID	COMMAND	PAYLOAD
'A','S','L','S','-','Q',0x00	0x04	*See tables below*

CUE_LOAD payload

CUE NUMBER	FILE BUFFER	
uint8	uint8	variable

CUE_CLEAR

Clears a cued file from a node. Cue number must be specified. CLEAR must be sent prior to the beginning of the show to prevent conflicts. The command payload only consists CUE NUMBER to be deleted.

CUE_CLEAR Packet example

ID	COMMAND	PAYLOAD
'A','S','L','S','-','Q',0x00	0x05	*see table below*

CUE_CLEAR payload

CUE NUMBER
int8

ALL_OFF

Independently turns all functions and outputs off without changing the control settings. Operating status prior to ALL_OFF may be reestablished by RESTORE. This command requires no payload as it affects the entire node.

ALL_OFF Packet example

ID	COMMAND	PAYLOAD
’A’,’S’,’L’,’S’,’-’,’Q’,0x00	0x06	*NONE*

RESTORE

Independently turns all functions and outputs off without changing the control settings. Operating status prior to ALL_OFF may be reestablished by RESTORE. This command requires no payload as it affects the entire node.

RESTORE Packet example

ID	COMMAND	PAYLOAD
’A’,’S’,’L’,’S’,’-’,’Q’,0x00	0x07	*NONE*

3.2.2.4 DMX specific commands

DMX specific commands for Q protocol

Mnemonic	Hex Code	Brief Description
CONTROL	0X10	Sets a a universe’s channel value.

CONTROL

The CONTROL command is used to update a node’s DMX channel values live. While it is not intended to be used during to control a large amount of DMX apparatus, it is very useful for getting visual feedback while programming a show.

Note: The specification for this CONTROL command is due to major changes in the next version of the protocol. It has been specified in the aim to demonstrate that using the Q protocol still offers a possibility for live control lighting equipment (not over pre-loaded cues).

CONTROL Packet example

ID	COMMAND	PAYLOAD
’A’,’S’,’L’,’S’,’-’,’Q’,0x00	0x10	*see table below*

CONTROL command payload

CHANNEL ID	CHANNEL VALUE
uint8	uint8

3.2.2.5 Pyro specific commands

PYRO specific commands for Q protocol

Mnemonic	Hex Code	Brief Description
TEST	0X20	Launches a test sequence on PYRO nodes .
TEST_RESPONSE	0X21	Response to a TEST command .

TEST

TEST command launches a test sequence on the desired PYRO node. PAYLOAD for a TEST command is comprised of a list of the cues which should be tested. This commands expects a TEST_RESPONSE packet containing information regarding the tested cues as a response. A maximum timeout of 3 seconds between sending TEST command and receiving TEST_RESPONSE may be assumed.

TEST Packet example

ID	COMMAND	PAYLOAD
'A','S','L','S','-','Q',0x00	0x20	*see table below*

CUE_TEST payload

CUE NUMBER
int8

TEST_RESPONSE

Following the reception of a TEST command, a PYRO Node sends a TEST_RESPONSE to the controller. PAYLOAD contained within this packet delivers information regarding the status (OK or nok) of each pyrotechnic cue which was tested.

CUE_TEST_RESPONSE packet example

ID	COMMAND	PAYLOAD
'A','S','L','S','-','Q',0x00	0x21	*see table below*

CUE_TEST payload

CUE NUMBER
int8

3.2.2.6 Discovery commands

Discovery commands for Q protocol

Mnemonic	Hex Code	Brief Description
POLL	0X30	Device discovery command.
POLL_REPLY	0X31	Response to a POLL command containg statistics regarding the discovered node.

POLL packet

POLL packets should be sent to nodes by controllers every 3 seconds to ensure that there was no node disconnection. A maximum timeout of 3 seconds between sending QPOLL and receiving POLL_REPLY packets may be assumed. Node should be considered as disconnected if response hasn't been received by this time.

POLL packet example

ID	COMMAND	PAYLOAD
'A','S','L','S','-','Q',0x00	0x30	*NONE*

POLL_REPLY packet

POLL_REPLY packets are sent back to a controller by a polled node. Such packets come with a PAYLOAD containing information regarding the node such as battery status, RSSI[2] and uptime.

POLL_REPLY packet example

ID	COMMAND	PAYLOAD
'A','S','L','S','-','Q',0x00	0x31	*see table below*

POLL_REPLY payload

battery status (%)	RSSI (db)	uptime (s)
uint8	int8	int16

[2] Received Signal Strength Indicator

3.3 Nodes

3.3.1 Features

Nodes are autonomous slave devices capable of triggering effects or sequences of effects. Triggering effects is done by sending uniquely identified cues from a controller to a node. Each node is able to trigger up to 254 cues. Depending on the type of node, a cue may trigger a pyrotechnic departure (line), a time-coded DMX sequence (scene). Every node, unregarding its type comes with a set of common features.

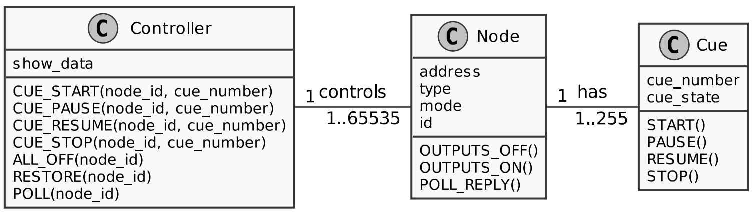


Fig.4.3 - Node class diagram

3.3.1.1 Cue Control

A node's cue may circle through three different states. Prior to receiving any cue command from the controller, a cue is initialized in a *"stopped"* state. A cue in stopped state can only transition to a *"running"* state. Receiving a CUE_START instruction initiate the transition to a running state. Entering the running state instantly initiates the playing of the cue in loop mode until a CUE_STOP or CUE_PAUSE instruction is received.

A cue in running state can either transition to a *"paused"* state or a stopped state. Receiving a CUE_PAUSE instruction initiate the transition to a paused state. Entering the paused state, instantly stops the cue from playing. Its playtime value is temporarily stored into the node's memory so it can be resumed at the same point in time. A paused cue is put back into its running state as soon as a CUE_RESUME instruction is received. It may also be reset to its stopped state when receiving a CUE_STOP instruction.

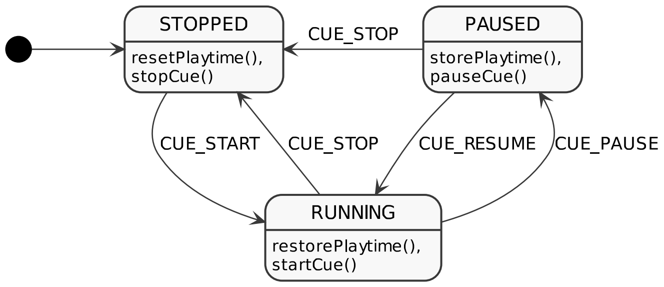


Fig.4.4 - Cue control state machine

3.3.1.2 Output Control

A node is able to operate in OUTPUTS_ON and OUTPUTS_OFF modes. In its default OUTPUTS_ON mode, a node operates normally, cue Control messages are interpreted and routed through the node's outputs. Putting the node in its OUTPUTS_OFF mode momentarily disable the routing of the cues to the node's output, however, messages will still interpreted. The reception of a RESTORE packet will cause the node to return to its OUTPUTS_ON state.

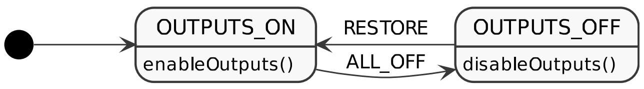


Fig.4.5 - Output Control state machine

3.3.2 Configuration

3.3.2.1 Pairing

Wired

In the case of a wired environment, communication between controllers and nodes can be established over Ethernet cables (cat-5 or upper). If more than one node is being controlled network switchs and routers may be used to forward data to their destination.

Wireless

A node can be added onto the wireless network using WPS. To do so, the controller is put in pairing mode, nodes can be associated to the network by pressing a "pair" button. The information about whether or not the node has been added onto the network should be displayed on the node itself. For obvious safety reasons, the pairing sequence is automatically terminated as soon as a device has been paired. this operation must be repeated for each node.

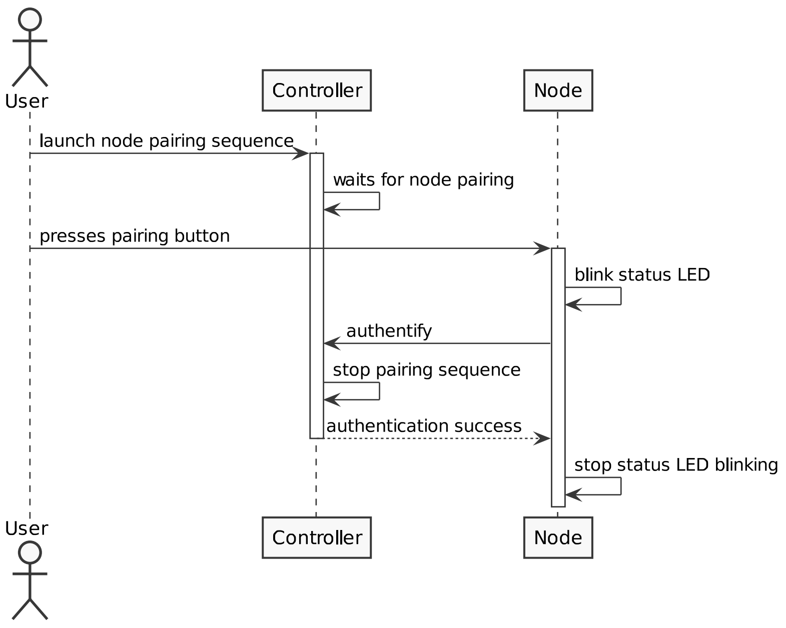


Fig.4.6 - WPS pairing sequence

3.3.2.2 Identification

Every node is given a class B private static IP address which least significant byte is used for identification (*see 5.2.1.3*). This identification byte corresponds to the node id *(see Fig.5.1)* and must be user-defined through the node's HMI using "plus" and "minus" buttons to select an id value between 0 and 255. Currently selected value will be displayed over the node's 16x2 display.

3.3.3 DMX Nodes

3.3.3.1 About DMX

DMX controlled apparatus, commonly called fixtures which come with *"n"* set of variable parameters each controllable over *"n"* amount of channels are dispatched into groups called DMX universes, which comprise of 512 DMX channels each independently controllable over 512 different values. Manually controlling a large set of fixtures is beyond human capabilities. The behavior for one or many fixtures parameters can be programmed into groups of time-coded steps called scenes. These scenes are manually or automatically triggered over specific cues. For additional customization, specific scene-relative parameters such as speed, color or strobe frequency can be controlled while playing back the scene.

3.3.3.2 Special Features

The DMX nodes' purpose is to manage DMX controlled stage apparatus. They come with a single standard DMX output for controlling a DMX512 universe. DMX nodes operate using cue-triggerable DMX sequence presets (scenes) which are uploaded into the node by a controller prior to a show.

The specificity of the present system lies in its feature allowing DMX scenes to be stored into a node's memory. This allows for Stage equipment to be controlled over triggers of cue-identified scenes rather than stream of live DMX values. Each stored scene is identified by a cue number which is used for routing incoming messages transporting information whether to play, pause or resume a scene.

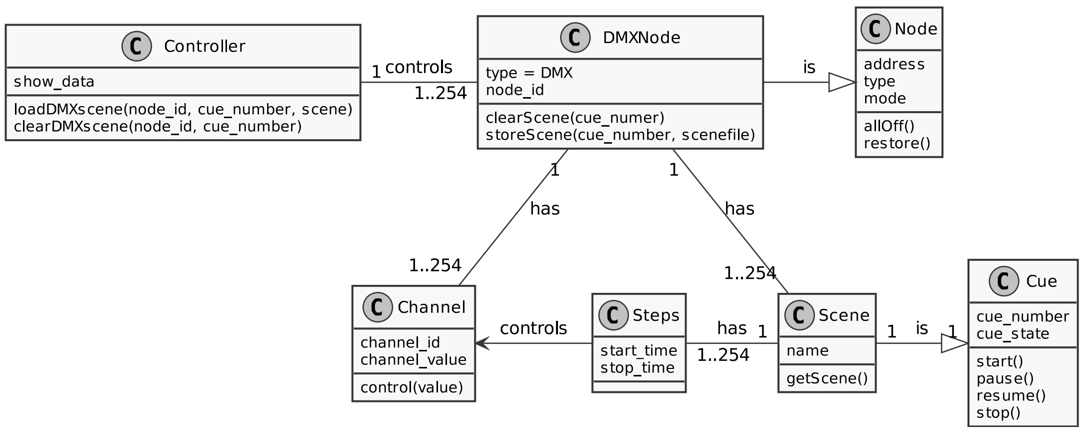


Fig.4.12 - Class diagram of a DMX node

Scene definition

Key	Type	Description
"steps_count"	uint16	the number of steps in the scene
"steps"	Step Object Array	an array containing step objects

Step definition

Key	Type	Description
"start_time"	uint16	value of the step starting time in ms
"stop_time"	uint32	value of the step stopping time in ms
"channels_count"	unint16	the number of channels in a step
"channels"	Channel Object Array	an array containing channel objects

Channel definition

Key	Type	Description
"channel_id"	uint16	the id of the channel
"channel_value"	uint8	the value of the channel

Scene Management

Scenes may be stored into a DMX node using the CUE_LOAD command. The payload of a CUE_LOAD packet destined to a DMX node contains a cue number used to identify the scene and a json-like "scene" file containing the scene information. A scene may be cleared from a node's memory using a CUE_CLEAR command. Scene to be deleted is identified using the cue_number field of the received payload. Scenes are controlled using *Cue Control* features as defined in section 5.3.1.1. The structure of a scene file is based on the object definition of a scene environment *(see example below)*.

```
{
  "steps_count": 1,
  "steps": [
    {
      "start_time": 0,
      "stop_time": 5000,
      "channels_count": 1,
      "channels": [
        {
          "channel_id": 18,
          "channel_value": 120
        }
      ]
    }
  ]
}
```

Fixture Control

While the control of a lighshow essentially relies on the control of pre-recorded scene cues, DMX channels for a given node may be manipulated using the CONTROL command. Channel's value is updated subsequently to the reception of a payload issued by a CONTROL command. Only the desired channel will be affected by the value change. Other channels in the universe will remain unchanged.

3.3.4 PYRO Node

3.3.4.1 About electronic pyrotechnic firing

Pyrotechnic firing system consist of uniquely addressed modules capable of triggering a specific amount of pyrotechnic cues more commonly known as lines. each of these lines consists of a switching device used to forward firing voltage to single electronic igniters (most commonly known as e-match) or many igniters connected in series.

3.3.4.2 Special Features

While the Q protocol specifies a maximum amount 255 cues per-nodes, only 32 will be used for Pyro nodes. Contrary to DMX nodes, Pyro nodes will not be storing any Pyrotechnic sequence presets for safety reasons. In opposition to lightshows, pyrotechnic shows may be controlled fully manually. However, for more complex shows requiring specific timings, pyrotechnic lines must be triggered automatically through an internal or external source of time-coded cues.Z

Notes: It is to be noted that in no way is this specification is intended to replace any aspect of normal performance safety which is either required or makes good sense when hazardous equipment is in use. Automatic safety devices and manual controls such as deadman switches, emergency stops, proximity sensors, limit switches and complementary safety devices shall be used for ensuring maximum safety. CUE_PAUSE, CUE_RESUME and CUE_STOP command should be ignored by pyrotechnic nodes as pyrotechnic cues cannot be paused, stopped or resumed.

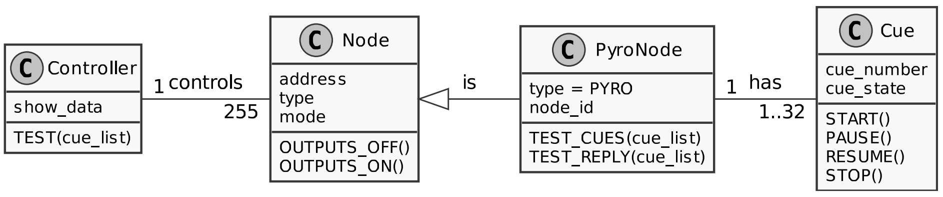


Fig.4.16 PYRO Node class diagram

Arming Node

Pyrotechnic nodes may be ARMED or DISARMED using Output Control features defined in section 5.3.1.2. A node running in OUTPUTS_OFF mode is considered as "disarmed" while A node running in OUTPUTS_ON mode is considered as "armed". Pyrotechnic nodes should always be started in OUTPUTS_OFF mode.

Testing Lines

Testing operation may only be run when a node is disarmed. Following the reception of a TEST packet, a pyrotechnic node should reply with a TEST_REPLY packet which payload contains information regarding the validity of each tested line.

Firing Lines

Firing operation is done by injecting a Firing voltage through the desired line. A firing operation is instantiated following the reception of a CUE_START packet. A Firing operation may only be run when a node is armed.

3.3.5 Media Servers

3.3.5.1 Features

A media server refers either to a dedicated computer appliance or to a specialized application software, ranging from an enterprise class machine providing video on demand, to, more commonly, a small personal computer or NAS (Network Attached Storage) for the home, dedicated for storing various digital media (meaning digital videos/movies, audio/music, and picture files). Features for controlling such systems are in every aspect similar to the features detailed in section 5.3.1.As the Q protocol hasn't been yet implemented into any non-generic system ye this section does not reveal any details regarding their operation.

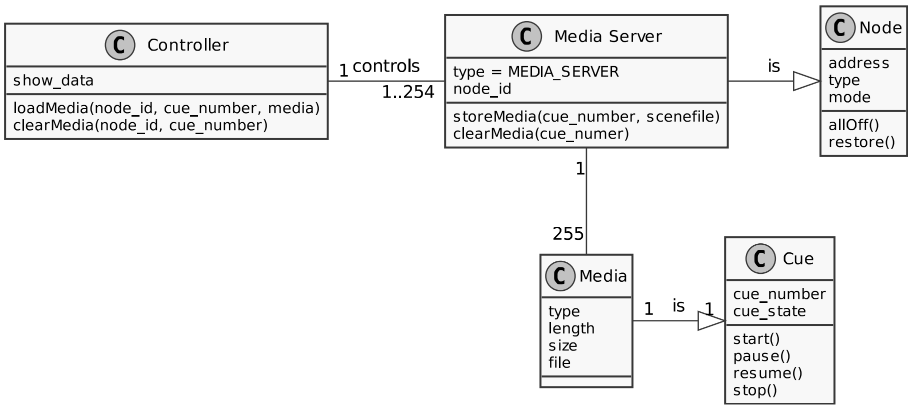


Fig.4.19 Media Server architecture

Storing Media Files

Media files may be uploaded from a controller straight into a media server following the reception of a CUE_LOAD packet. The Q protocol allows for any file format to be sent over the networks. For security purposes, file types should be checked by media servers prior to storing the file onto the system.

Controlling Media Files

A stored media is identified using a cue number and may later be played back, stopped, paused or resumed using the Cue Control features of generic nodes as defined in section 5.3.1.1. Subsequently to the reception of an ALL_OFF packet the controlled media server should immediately disable the signal output while continuing to receive and process incoming information. Signal output shall be restored once a RESTORE packet has been received.

References

[1] Isabelle Chabinian, Bruno Chevallet. *Manuel de formation F4/T2 de niveau 1 et 2, 2018*.

Protocols specifications:

[2] MIDI Manufacturers Association *Midi Show Control Specification V1.0, 1990*

[3] Artistic Licence, *Specification for the Art-Net 4 Ethernet Communication Protocol, 2015*

[4] ESTA, *Entertainment Technology Lightweight streaming protocol for transport of DMX512 using ACN, 2016*

[5] ESTA, *Entertainment Technology USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories, 2004*

Websites:

[6] ESTA, *https://www.esta.org/*

[7] Art-Net *https://art-net.org.uk/*

[8] Artistic License *https://artisticlicence.com/*

[9] Wikipedia *https://www.wikipedia.org/*

[10] Stackoverflow *https://stackoverflow.com/*