Art-Osc

Specification for the

Art-Osc Communication Protocol

ART OSC

COMMENDED AT THE 2013
plasalondon
AWARDS FOR INNOVATION

Artistic Licence

Revision History:

V1.0 (9/9/13)
First release.
V1.1 (14/9/13)
Minor corrections to Parameter Table. Expanded discussion of purpose. Added ports information.
V1.2 (1/10/13)
Formatting improvements.
V1.3 (27/12/14)
Formatting improvements.
Addition of DALI
Expanded discussion of timing.
Expanded discussion of Argument list.
Added optional time stamp
V1.4 (20/2/15)
Second release.
V1.5 (20/10/15)
Corrected error in Dali circuit broadcast description
Added more Dali examples.
Added Dali function call example
V1.6 (18/06/18)
Renamed gateway-address to port-address for DALI to provide common terminology between Art-Net and Art-Osc.
Added more Dali examples.
V1.7 (5/06/22)
Renamed gateway-address to port-address for DALI to provide common terminology between Art-Net and Art-Osc.
Added more Dali examples.

Abstract

Art-Osc is designed to provide a standardised syntax for transmission of lighting control data over the Open Sound Control (OSC) protocol. OSC is a transport independent mechanism and by definition, so is Art-Osc. However it is considered likely that UDP will be the preferred transport.

Implementation

Art-Osc provides a common syntax which allows three separate ‘languages’ to be encapsulated by OSC. These are:

1. DALI: Transfer of both forward and reverse DALI frames using a per-circuit addressing mechanism with loosely coupled poll and response transactions.
2. Automation: Transfer of remote control and triggering information.
3. RDM: Transfer of RDM data, in a human readable form, out to devices capable of displaying or acting upon the data.

Initially, Art-Osc servers are likely to be iPad, iPhone and Android tablet devices.

Ports

Art-Osc Servers (receivers) listen on port 7000.
Art-Osc Clients (transmitters) send on port 8000.

Bandwidth

It is important that Art-Osc Clients manage efficiently the available network bandwidth. This fundamentally means that data shall not be sent more frequently than can be processed or used.

The specific considerations are language dependent:

1. DALI: DALI packets shall not be sent to a specific Vector-Instance more frequently than the packet can be transferred to the DALI physical layer.
2. Automation: There is no specific timing constraint, but designers should endeavour to minimise redundant data.
3. RDM: RDM packets shall not be sent to a given Address Pattern more frequently than the related RDM data can be transferred to the DMX512 physical layer. This means a minimum update repeat time of 23mS. It is envisaged that real update rates will be substantially slower. Designers should keep in mind the

intent to transfer human readable data, so update rates measured in seconds will suffice.

Casting

The preferred cast is Unicast. Broadcast is allowed but not preferred. Multicast is allowed but no specific joining mechanism has been defined.

Terms and conventions

- In line with the OSC standard, any device that sends Art-Osc packets is an Art-Osc Client; any device that receives Art-Osc Packets is an Art-Osc Server.
- All multi-byte data is transmitted big-endian.
- Floating point data is formatted as single precision to the IEEE 754 standard
- This protocol references ANSI E1.20 -2010 Entertainment Technology RDM Remote Device Management Over DMX512 Networks by the abbreviation RDM.
- This protocol references the CNMAT, UC Berkeley, Open Sound Control Specification v1.1 by the abbreviation OSC.
- This protocol references the IEC 62386 standard by the abbreviation DALI.

OSC Message

The *References* section of this document lists the OSC specification URL. This section is intended to provide a very brief overview of the OSC packet structure and how it relates to Art-Osc. The reader is urged to review the entire OSC document.

An OSC message is a string of contiguous bytes of size which is a multiple of 4. The OSC message comprises three parts:

1. Address Pattern
2. Type Tag String
3. Argument List

Address Pattern

The Address Pattern is a string beginning with the '/' (forward slash) character (47_{10}). It is the contents of the Address Pattern that forms the heart of the Art-Osc protocol.

Type Tag String

The Type Tag is a string beginning with the ',' (comma) character (44_{10}). Each character in the string defines the type and order of the data in the Argument List. OSC defines numerous data types, but currently Art-Osc implements four:

Tag Type Character	Decimal	Meaning
f	102_{10}	Float 32-bit
s	115_{10}	String containing data
S	83_{10}	String containing label
t	116_{10}	Time stamp in NTP format

Argument List

The Argument List is a contiguous sequence of binary data, ordered and formatted as defined by the Type Tag String. The quantity, sequence and purpose of the Argument List varies based on the Language field of the Address Pattern.

Art-Osc Syntax

Art-Osc data is carried by OSC and the syntactic details are defined by that protocol. Devices implementing Art-Osc shall adhere to the OSC protocol. It should also be noted that the syntax of Art-Osc has been designed to be compatible with the TouchOSC application by Hexler, which can be considered a reference design to test against.

Art-Osc Syntax for Address Pattern Field

The Address Pattern starts with the '/' forward slash character. This character is also used to provide a hierarchical separator in the string. Art-Osc subdivides the Address Pattern into five sub fields:

- Signature
- Language
- Object Vector
- Instance
- Parameter

These sub fields are delimited by the forward slash as follows:

/Signature/Language/ObjectVector/Instance/Parameter

Signature

The signature is the two character pair of 'AL', both uppercase. The purpose of this field is to allow the Art-Osc server (receiver) to discriminate Art-Osc from other OSC traffic.

Language

The Language field is used to describe the meaning and format of the three remaining fields (Object Vector, Instance and Parameter). The field is fixed length and contains three uppercase alpha characters.

Currently, four languages are defined:

Language Code	Name	Meaning
DAF	DALI forward	Designed to transmit DALI forward frame data across the network.
DAR	DALI reverse	Designed to transmit DALI reverse frame data across the network.
AUT	Automation	Designed to transmit general purpose remote control messages for automating and triggering lighting systems.
RDM	Remote Device Management	Designed to transmit human readable status and sensor data from an original RDM source.

Object Vector

The Object Vector meaning is dependent upon the Language as defined below.

Object Vector (Language = DAF/DAR)

The Object Vector defines the 'port-address' of the Art-Osc to DALI gateway. Each circuit of a DALI gateway should be assigned a unique port-address.

The port-address is expressed in dot notation with fixed field length as shown below:

bbb.aaa

These fields are defined in the following table.

Field	Name	Description
bbb	MSB port-address	The most significant byte of the 16-bit port-address. (In Art-Net terms, this is the 'Net').
aaa	LSB port-address	The least significant byte of the 16-bit port-address. (In Art-Net terms this is the 'Sub-Net+Universe').

The port-address 255.255 is the broadcast address which means the Art-Osc packet shall be accepted by all port-addresses.

Object Vector (Language = AUT)

The Object Vector defines the address of the object to which the Art-Osc packet is directed. The object Vector is expressed in dot notation with fixed field length as shown below:

mmm.mmm.ppp.ppp.ttt

These fields are defined in the following table.

Field	Name	Description
mmm.mmm	Manufacturer ID	The Manufacturer's ID code as assigned by PLASA, expressed as two, three-digit decimal numbers, separated by a dot.
ppp.ppp	Product Code	The Manufacturer's product code, expressed as two, three-digit decimal numbers, separated by a dot. Product codes are assigned by a manufacturer to uniquely differentiate the way in which their products respond to Automation commands.
ttt	Tag Number	This field is provided for future address scope expansion and is currently set to 001.

Example: The Artistic Licence manufacturer ID is 65_{10} 76_{10}. The product code for Artistic Licence product Colour-Tramp is 100_{10} 1_{10}. So the Object Vector will be:

065.076.100.001.001

Object Vector (Language = RDM)

The Object Vector defines the RDM (Remote Device Management) UID (Unique Identification) and sub-device to which the Art-Osc packet is related. The object Vector is expressed in dot notation with fixed field length as shown below:

mmm.mmm.ddd.ddd.ddd.ddd.sss

These fields are defined in the following table.

Field	Name	Description
mmm.mmm	Manufacturer ID	The Manufacturer's ID code as assigned by PLASA, expressed as two, three-digit decimal numbers, separated by a dot.
ddd.ddd.ddd.ddd	Product Code	The least significant four bytes of the UID. Expressed as four, three-digit decimal numbers, separated by a dot.
sss	Sub-device	The sub-device in the range 001 to 512. A value of 000 represents the root device.

Example: The Artistic Licence manufacturer ID is 65_{10}-76_{10}. To address a message from the root of Artistic Licence product with UID 65.76.1.2.3.4 the following Object Vector is used:

065.076.001.002.003.004.000

Instance

The Instance meaning is dependent upon the Language as defined below.

Instance (Language = DAF/DAR)

The Instance field is a fixed width field of three numeric characters representing an integer in the range 0 to 999, encoded as follows:

1. Values 0-254 are deprecated and should be interpreted as 255.
2. Value 255 instructs the device to process the DAF/DAR message.
3. Values 256-999 are reserved for future expansion.

Instance (Language = AUT)

The Instance defines the actual instance of an object used in automation. For example, a screen may contain multiple switches. The Instance identifies which switch is which. The instance field is a fixed width field of three numeric characters representing an integer in the range 0 to 999.

Instance (Language = RDM)

The Instance defines the RDM sensor number. The instance field is a fixed width field of three numeric characters representing an integer in the range 0 to 999.

Parameter

The Parameter meaning is dependent upon the Language as defined below.

Parameter (Language = DAF)

The Parameter contains the address byte of the forward frame.

Parameter (Language = DAR)

The Parameter contains the address byte of the forward frame that caused this reverse frame to be issued.

The parameter field is a fixed width field of three numeric characters representing an integer in the range 0 to 999.

Values in the range 0 – 255 are defined in IEC62386-102. Values 256-999 are reserved for future expansion and implementation of DALI discovery and commissioning systems.

The Address Byte is described in binary as YAAA AAAS.

Y is the short address indicator. Y=0 for short address, Y=1 for group, broadcast or special.

S is the selector bit. S=0 means that the data byte is direct Arc data. S=1 means the data byte is a command.

A is the DALI ballast, group or scene address.

Field	Y	A	A	A	A	A	A	S
64 short addresses 0 - 63	0	A	A	A	A	A	A	S
16 group addresses 0 - 15	1	0	0	A	A	A	A	S
Broadcast	1	1	1	1	1	1	1	S

Address-Byte	Purpose
999	Start full discovery.

Parameter (Language = AUT)

The Parameter field describes the parameter which is addressed. The field is a variable length string selected from the following table.

Parameter	Description
FADER_A	A fader or slider with an absolute output in range 0-255
XY_A	An X-Y trackpad with two absolute outputs in range 0-255
SWITCH_A	A selection switch with absolute output of 0 or 1
MACRO_A	A momentary switch with absolute output 0 or 1
LED_A	An LED indicator with analogue display range 0-255
LED_B	An LED indicator with binary display range 0 or 1
LAB_A	A text label.
LAB_FADER	A text label attached to a FADER_A
LAB_XY_A	A text label attached to a XY_A
LAB_SWITCH_A	A text label attached to a SWITCH_A
LAB_MACRO_A	A text label attached to a MACRO_A
LAB_LED_A	A text label attached to a LED_A
LAB_LED_B	A text label attached to a LED_B

The parameters can be further subdivided to access specific properties of a parameter, such as colour. See examples section for details.

Parameter (Language = RDM)

The Parameter defines the RDM PID (Parameter ID). The field is a variable length string selected from the following table.

Parameter	Description	Allowed Argument Tag-Types
COMMS_STATUS_S	Report on communication errors	s,S
STATUS_MESSAGES_S	Last status message sent by device	s,S
DEVICE_INFO_S	Summary of device info	s,S
DEVICE_MODEL_DESCRIPTION_S	Description of the device model	s,S
MANUFACTURER_ LABEL_S	Manufacturer name for this device.	s,S
DEVICE_LABEL_S	User name for this device	s,S
SOFTWARE_VERSION_LABEL_S	Software version	s,S
BOOT_SOFTWARE_VERSION_ LABEL_S	Boot software version	s,S
PERSONALITY_F	Personality number (1 - x)	f,S
PERSONALITY_S	Personality number (1 - x)	s,S
PERSONALITY_ DESCRIPTION_S	Description of current personality	s,S
START_ADDRESS_F	Start address of fixture	f,S
START_ADDRESS_S	Start address of fixture	s,S
SENSOR_F	Value of sensor defined by Instance	f,S
SENSOR_S	Value of sensor defined by Instance	s,S
SENSOR_QUANTITY_F	Quantity of sensors	f,S
SENSOR_QUANTITY_S	Quantity of sensors	s,S
SENSOR_DEFINITION_S	Description of Sensor	s,S
DEVICE_HOURS_F	Hours that device has been powered	f,S
DEVICE_HOURS_S	Hours that device has been powered	s,S
LAMP_HOURS_F	Hours that lamp has been energised	f,S
LAMP_HOURS_S	Hours that lamp has been energised	s,S
LAMP_STRIKES_F	Number of times lamp struck	f,S
LAMP_STRIKES_S	Number of times lamp struck	s,S
PAN_INVERT_F	Moving light pan inverted	f,S
PAN_INVERT_S	Moving light pan inverted	s,S
TILT_INVERT_F	Moving light tilt inverted	f,S
TILT_INVERT_S	Moving light tilt inverted	s,S
PAN_TILT_SWAP_F	Moving light pan and tilt swapped	f,S
PAN_TILT_SWAP_S	Moving light pan and tilt swapped	s,S
DISPLAY_INVERT_F	Display inversion status	f,S
DISPLAY_INVERT_S	Display inversion status	s,S
DISPLAY_LEVEL_F	Brightness of display	f,S
DISPLAY_LEVEL_S	Brightness of display	s,S
LAMP_STATE_F	Status of lamp	f,S
LAMP_STATE_S	Status of lamp	s,S
LAMP_ON_MODE_F	Condition to trigger a lamp strike	f,S
LAMP_ON_MODE_S	Condition to trigger a lamp strike	s,S

Art-Osc Syntax for Argument List

The Tag-Type, quantity, order and meaning of the Argument List is defined by the Language as follows:

Language	Argument 1 detail	Arg 1 Tag	Argument 2 detail	Arg 2 Tag	Argument 3 detail	Arg 3 Tag
DAF	Data field of the forward frame.	f	Optional Time Stamp in NTP format.	t	N/A	
DAR	Data field of the forward frame which caused this reverse frame.	f	Data field of the reverse frame.	f	Optional Time Stamp in NTP format. Type Tag = t.	t
AUT	Contains the data required by the parameter.	f,s,S	Optional Time Stamp in NTP format.	t	N/A	
RDM	Contains the data required by the parameter.	f,s,S	Optional Time Stamp in NTP format.	t	N/A	

Please refer to the OSC document for details of how the arguments are delimited.

Time Stamps

The table above shows that an additional, final argument of a time stamp can be appended to the packet.

The time stamp is in NTP format as defined by the OSC document. The purpose of this packet is twofold:

1. Encode the time at which the event, described by the packet, occurred.
2. Provide a temporal sequence such that the receiving application can detect if any packets have been received out of order.

See References at the end of this document for further information on NTP.

Art-Osc Examples

Art-Osc data is carried by OSC and the syntactic details are defined by that protocol.

Language = AUT

Address Pattern: /AL/AUT/065.076.100.001.001/003/LAB_A
Tag Type 1 s
Argument 1 The cat jumped over the fox
Description: Automation command to display label text.

Address Pattern: /AL/AUT/065.076.100.001.001/003/LAB_A/color
Tag Type 1 s
Argument 1 red
Description: Automation command to change font colour of label.

Address Pattern: /AL/AUT/065.076.100.001.001/003/MACRO_A
Tag Type 1 f
Argument 1 1
Description: Automation command to trigger macro.

Language = RDM

Address Pattern: /AL/RDM/065.076.000.000.000.001.007/000/DEVICE_INFO_S
Tag Type 1 s
Argument 1 Device is operating normally
Description: Transmit the DEVICE_INFO string for RDM device with UID 65.76.0.0.0.1, sub-device 7.

Address Pattern: /AL/RDM/065.076.000.000.000.001.000/002/SENSOR_DEFINITION_S
Tag Type 1 s
Argument 1 Input Voltage
Description: Transmit the SENSOR_DEFINITION for root device with UID 65.76.0.0.0.1, sensor 2.

Language = DAF

Address Pattern: /AL/DAF/000.001/255/126
Tag Type 1 f
Argument 1 100
Description: Set ARC level to 100 for the ballast with short address 63 of DALI circuit at port-address 000.001

Address Pattern: /AL/DAF/000.000/255/000
Tag Type 1 f
Argument 1 100
Description: Set ARC level to 100 for the ballast with short address 0 of DALI circuit at port-address 000.000

Address Pattern: /AL/DAF/255.255/255/254
Tag Type 1 f

Argument 1	150
Description:	Set ARC level to 150 for all ballasts on all DALI circuits at all port-addresses. Note that this would require a broadcast network packet.

Address Pattern:	/AL/DAF/000.001/255/254
Tag Type 1	f
Argument 1	170
Description:	Set ARC level to 170 for all ballasts of DALI circuit at port-address 000.001

Address Pattern:	/AL/DAF/000.001/255/254
Tag Type 1	f
Argument 1	254
Description:	Set ARC level to 254 for all ballasts of DALI circuit at port-address 000.001

Software Development Kit (SDK)

The Art-Net and Art-Osc SDK is downloaded and installed with DMX-Workshop. The SDK comprise a dynamic link library called ArtNetAcn.dll which contains additional call for transmitting and receiving Art-Osc

Send Art-Osc packet with float

The following call is used to send an Art-Osc packet with float data.

```
Bool ArtNetSendOscFloatExt    (
                              char* IpOverride,
                              char* Vector,
                              int Tag,
                              int Instance,
                              char* Parameter,
                              float Data
                              )
```

Argument	Type	Purpose
IpOverride	char*	Optional IP address in dot format. Use NULL to instruct SDK to direct datagram to the system's default Art-Osc address.
Vector	char*	The manufacturer and product
Tag	int	Set to 001
Instance	int	The control's instance
Parameter	char*	The type of control
Data	float32	Data

The following example call results in the floating point number 23.45 being sent to Art-Osc Address Pattern: `AL/AUT/065.076.100.001.001/003/LAB_MACRO_A`.

```
ArtNetSendOscFloatExt  (
                              NULL,
                              "/AL/AUT/065.076.100.001",
                              1 /*Tag*/,
                              3 /*Instance*/,
                              "LAB_MACRO_A",
                              23.45
                              );
```

Send Art-Osc packet with string

The following call is used to send an Art-Osc packet with string data.

```
bool ArtNetSendOscStringExt   (
                              char* IpOverride,
                              char* Vector,
                              int Tag,
                              int Instance,
                              char* Parameter,
                              char* Data
                              )
```

The following example call results in the string “Hello World” being sent to Art-Osc Address Pattern: `AL/AUT/065.076.100.001.001/002/LAB_MACRO_A`.

```
ArtNetSendOscStringExt (
                              NULL,
                              "/AL/AUT/065.076.100.001",
                              1 /*Tag*/,
                              2 /*Instance*/,
                              "LAB_MACRO_A",
                              "Hello World"
                              );
```

The following example call results in the floating point number 23.45 being sent to Art-Osc Address Pattern: `AL/AUT/065.076.100.001.001/003/LAB_MACRO_A`.

```
ArtNetSendOscFloatExt  (
                              NULL,
                              "/AL/AUT/065.076.100.001",
                              1 /*Tag*/,
                              3 /*Instance*/,
                              "LAB_MACRO_A",
                              23.45
                              );
```

Send Art-Osc DALI packet with uchar

The following call is used to send an Art-Osc packet for language DAF and DAR.

```
bool ArtNetSendOscDali1 (
                                char* IpOverride,
                                char* Vector,
                                int Instance,
                                unsigned char Ballast,
                                unsigned char Data
                                );
```

Argument	Type	Purpose
IpOverride	char*	Optional IP address in dot format. Use NULL to instruct SDK to direct datagram to the system's default Art-Osc address.
Vector	char*	The Object Vector which includes language and port-address
Instance	int	Set to 255
Ballast	uchar	The short address
Data	uchar	Data

The following example call sets ARC level to 100 for all ballasts of DALI circuit at port-address 000.001

```
ArtNetSendOscDali1      (
                        NULL,
                        "/AL/DAF/000.001",
                        255
                        0xfe"  /*ballast address*/
                        100    /*data*/
                        );
```

The following example commands all ballasts in Group 1 of DALI circuits on port-address 000.015 to Go To Scene 2

```
ArtNetSendOscDali1      (
                        NULL,
                        "/AL/DAF/000.015",
                        255,
                        0x83"  /*group address*/
                        0x13   /*data*/
                        );
```

References:

The latest revision of this document:
http://www.artisticlicence.com/WebSiteMaster/User%20Guides/art-osc.pdf

The Open Sound Control Organisation:
http://opensoundcontrol.org/

The Open Sound Control v1.1 specification:
http://cnmat.berkeley.edu/system/files/attachments/Nime09OSCfinal.pdf

TouchOSC (Hexler) home page:
http://hexler.net/software/touchosc

DMX-Workshop (Artistic Licence) provides test transmission and reception of Art-Osc packets:
http://www.artisticlicence.com/WebSiteMaster/Software/dmxworkshopsetup.msi

The following link provides a good overview of the Network Time Protocol:
http://www.meinbergglobal.com/english/info/ntp-packet.htm

The Network Time Protocol is defined by RFC 5905
http://tools.ietf.org/html/rfc5905

Artistic Licence

E: Support@ArtisticLicence.com
W: www.ArtisticLicence.com