Open Sound Control

Presentation for P1722.1

Jeff Koftinoff <jeffk@meyersound.com> with assistance from Andy Schmeder <andy@cnmat.berkeley.edu>

Meyer Sound Laboratories, Inc.
www.meyersound.com

February 10, 2010
THIS DOCUMENT HAS BEEN PREPARED TO ASSIST THE IEEE P1722.1 WORKING GROUP, AND MAY BE ALTERED OR AMENDED AT A LATER DATE. NEITHER MEYER SOUND LABORATORIES, INCORPORATED NOR ANY OF ITS SUBSIDIARIES OR AFFILIATES ASSUME ANY LIABILITY IN CONNECTION WITH THE USE OF INFORMATION OR DATA CONTAINED IN THIS DOCUMENT. THE INFORMATION AND DATA CONTAINED IN THIS DOCUMENT ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, ALL OF WHICH ARE EXPRESSLY DISCLAIMED.
Items covered in this presentation:
Items covered in this presentation:

- OSC Overview
Items covered in this presentation:

- OSC Overview
- History of OSC
Items covered in this presentation:

- OSC Overview
- History of OSC
- Communicating OSC
Contents

Items covered in this presentation:

- OSC Overview
- History of OSC
- Communicating OSC
- Bundles and Timing
Items covered in this presentation:

- OSC Overview
- History of OSC
- Communicating OSC
- Bundles and Timing
- Example OSC messages
Items covered in this presentation:

- OSC Overview
- History of OSC
- Communicating OSC
- Bundles and Timing
- Example OSC messages
“A communication protocol that allows musical instruments, computers and other multimedia devices to share music performance data in realtime over a network...” –Wikipedia
OSC:

- Is an extensible encoding scheme
OSC:
- Is an extensible encoding scheme
- Is designed to be used with packets
OSC:

- Is an extensible encoding scheme
- Is designed to be used with packets
- Is designed to be used with streams with SLIP framing (rfc1055)
OSC:

- Is an extensible encoding scheme
- Is designed to be used with packets
- Is designed to be used with streams with SLIP framing (rfc1055)
- Integrates with the “Presentation” layer in the OSI model
- Is simpler to encode and decode than XML or ASN.1
OSC:
- Is an extensible encoding scheme
- Is designed to be used with packets
- Is designed to be used with streams with SLIP framing (rfc1055)
- Integrates with the “Presentation” layer in the OSI model
- Is simpler to encode and decode than XML or ASN.1
1993: Adrian Freed at CNMAT creates SynthControl
1993: Adrian Freed at CNMAT creates SynthControl
1997: OSC Specification at CNMAT by Matt Wright and Adrian Freed
1993: Adrian Freed at CNMAT creates SynthControl
1997: OSC Specification at CNMAT by Matt Wright and Adrian Freed
1998: “Typetags” contributed by James McCartney
1993: Adrian Freed at CNMAT creates SynthControl
1997: OSC Specification at CNMAT by Matt Wright and Adrian Freed
1998: “Typetags” contributed by James McCartney
1999: OSC-route for MaxMSP
- 1993: Adrian Freed at CNMAT creates SynthControl
- 1997: OSC Specification at CNMAT by Matt Wright and Adrian Freed
- 1998: “Typetags” contributed by James McCartney
- 1999: OSC-route for MaxMSP
- 2002: LCS (now Meyer Sound) supports OSC in the LX-300 system
- Now there are more than 80 implementations of the OSC protocol in hardware, software, firmware, in many programming languages on systems ranging from tiny microcontrollers to large synthesis engines: [www.opensoundcontrol.org/implementations](http://www.opensoundcontrol.org/implementations)
1993: Adrian Freed at CNMAT creates SynthControl
1997: OSC Specification at CNMAT by Matt Wright and Adrian Freed
1998: “Typetags” contributed by James McCartney
1999: OSC-route for MaxMSP
2002: LCS (now Meyer Sound) supports OSC in the LX-300 system
2004: JazzMutant releases the Lemur multi-touch OSC controller
1993: Adrian Freed at CNMAT creates SynthControl
1997: OSC Specification at CNMAT by Matt Wright and Adrian Freed
1998: “Typetags” contributed by James McCartney
1999: OSC-route for MaxMSP
2002: LCS (now Meyer Sound) supports OSC in the LX-300 system
2004: JazzMutant releases the Lemur multi-touch OSC controller
2005: Meyer Sound supports OSC in the Galileo product
1993: Adrian Freed at CNMAT creates SynthControl
1997: OSC Specification at CNMAT by Matt Wright and Adrian Freed
1998: “Typetags” contributed by James McCartney
1999: OSC-route for MaxMSP
2002: LCS (now Meyer Sound) supports OSC in the LX-300 system
2004: JazzMutant releases the Lemur multi-touch OSC controller
2005: Meyer Sound supports OSC in the Galileo product
2009: Meyer Sound supports OSC in the D-Mitri product

Now there are more than 80 implementations of the OSC protocol in hardware, software, firmware, in many programming languages on systems ranging from tiny microcontrollers to large synthesis engines: www.opensoundcontrol.org/implementations
1993: Adrian Freed at CNMAT creates SynthControl
1997: OSC Specification at CNMAT by Matt Wright and Adrian Freed
1998: “Typetags” contributed by James McCartney
1999: OSC-route for MaxMSP
2002: LCS (now Meyer Sound) supports OSC in the LX-300 system
2004: JazzMutant releases the Lemur multi-touch OSC controller
2005: Meyer Sound supports OSC in the Galileo product
2009: Meyer Sound supports OSC in the D-Mitri product

Now there are more than 80 implementations of the OSC protocol in
hardware, software, firmware, in many programming languages on systems
ranging from tiny microcontrollers to large synthesis engines:
www.opensoundcontrol.org/implementations
An OSC message contains:
- OSC Address
An OSC message contains:

- OSC Address
- Type Tags describing the count and types of the values
An OSC message contains:

- OSC Address
- Type Tags describing the count and types of the values
- The parameter values
OSC Address

The OSC Address is a path-like UTF-8 string describing the control point. Examples:

- /oscillator/1/freq
- /in/1/phantom
- /bus/2/mute
- /bus/2/eq/3/gain
OSC Address

The OSC Address is a path-like UTF-8 string describing the control point. Examples:

- /oscillator/1/freq
- /in/1/phantom
- /bus/2/mute
- /bus/2/eq/3/gain

When an OSC Client sends a message to an OSC Server it has the option to use simple wildcards:

- /oscillator/*/freq - set all oscillators to one frequency
- /in/[1-8]/phantom - set phantom power modes for inputs 1-8
- /bus/2/eq/*/gain - set the gains for all the eq’s for bus 2
OSC Address

The OSC Address is a path-like UTF-8 string describing the control point. Examples:

- /oscillator/1/freq
- /in/1/phantom
- /bus/2/mute
- /bus/2/eq/3/gain

When an OSC Client sends a message to an OSC Server it has the option to use simple wildcards:

- /oscillator/*/freq - set all oscillators to one frequency
- /in/[1-8]/phantom - set phantom power modes for inputs 1-8
- /bus/2/eq/*/gain - set the gains for all the eq’s for bus 2
The OSC Typetags are ASCII characters describing a type. Standard typetags include:

- “b” Blob
- “T” True
- “F” False
- “N” Nil
- “I” Impulse
- “i” 32 bit integer
- “s” Null terminated UTF-8 String
- “f” float in IEEE754 single precision format
All values are aligned to 4 byte boundaries, including null terminated strings.
OSC Values

- All values are aligned to 4 byte boundaries, including null terminated strings.
- All values are in network byte order
Example Messages

```
"/matrix/2/3/gain" "f" -6.0
```
Example Messages

- "/matrix/2/3/gain" "f" -6.0
- "/in/5/mute" "T"
Example Messages

- "/matrix/2/3/gain" "f" -6.0
- "/in/5/mute" "T"
- "/in/5(scale" "i" 10
Example Messages

- "/matrix/2/3/gain" "f" -6.0
- "/in/5/mute" "T"
- "/in/5/scale" "i" 10
An OSC Bundle contains:

- A Time Stamp
An OSC Bundle contains:

- A Time Stamp
- One or more OSC Messages to be executed at the specified time
An OSC Bundle contains:

- A Time Stamp
- One or more OSC Messages to be executed at the specified time
Time Stamps

- OSC Timestamps are in NTP format (32 bit seconds and 32 bit fraction of a second), but can be extended to use gPTP format.
**Time Stamps**

- OSC Timestamps are in NTP format (32 bit seconds and 32 bit fraction of a second), but can be extended to use gPTP format.
- All messages in an OSC bundle are to be executed atomically at the specified time.
OSC Timestamps are in NTP format (32 bit seconds and 32 bit fraction of a second), but can be extended to use gPTP format.
OSC Timestamps are in NTP format (32 bit seconds and 32 bit fraction of a second), but can be extended to use gPTP format.

All messages in an OSC bundle are to be executed atomically at the specified time
Set audio input 1 level to -10.0 dB
Example 1’s Request

1. 

```
"/in/1/level/db",f -10.0f
```
Example 1’s Request

Example 1

"/in/1/level/db",f -10.0f

Encoded as:

1 2f696e2f 312f6c65 76656c2f 64620000 ("/in/1/level/db")
2 2c660000 (",f")
3 c1200000 (-10.0f in IEEE754)
Example 2 - Bundle

Set audio input 1 level to -10.0 dB and set audio output 1 level to +40 dB at the time XXX.YYY
Example 2’s Request

1. at time XXX.YYY
2. "/in/1/level/db",f -10.0f
3. "/out/3/level/db",f 40.0f
Example 2’s Request

1. at time XXX.YYY
2. "/in/1/level/db",f -10.0f
3. "/out/3/level/db",f 40.0f

Encoded as:

1. 2362756e 646c6500 ("#bundle")
2. XXXXXXXX YYYYYYYY (Time XXX.YYY)
3. 000000018 (Msg Size 24 bytes)
4. 2f696e2f 312f6c65 76656c2f 64620000 ("/in/1/level/db")
5. 2c660000 (","f")
6. c120000 (-10.0f)
7. 00000018 (Msg Size 24 bytes)
8. 2f6f7574 2f312f6c 6576656c 2f64620000 ("/out/1/level/db")
9. 2c660000 (","f")
10. 42200000 (40.0f)
11. 00000000 (Msg Size 0 = none)