RTMP Decoded and Annotated

RTMP messages utilize a scheme where up to 64 channels are supported in a single network connection. This network connection must be persistent during the entire RTMP session. Channel 2 is used exclusively for system messages.

System Messages

A Ping message is sent from the server to the client to test the connection, synchronize the network timestamps, and a few other unknown control types of things. So far, 5 Ping message types are known. There are up to 4 parameters associated with a Ping message. The first two, both shorts, are mandatory, although it appears the first parameter is often just padding. The last 2 parameters are also shorts, and are optional.

- 0x1 Clear Stream

  This clears the currently playing stream.

- 0x3 Buffer Time

  The 3rd parameter is the buffer time from the client. This becomes the BufferTime property for the NetStream ActionScript class.

- 0x4 Reset Stream

  Reset the stream. This is often sent before clearing the stream.

- 0x6 Ping

  Ping the client. The 2nd parameter is the current time in milliseconds according to some documentation. I think the time stamp is actually the last two fields, totalling 4, which would be the size of time_t (usually a long) on an older system.

- 0x7 Pong

  Return in the 2nd parameter the timestamp received in the Ping message from the server.

Ping Message

02	12 byte header, system channel
00 00 00	timestamp
00 00 06	message size in bytes
04	message type, Ping
00 00 00 00	stream ID (value 0.0)
00 00	ping type, reset stream
00 00	second parameter
00 00	third parameter

Ping Timestamp Message

c2	1 byte header, system channel
00 06	ping type, ping client
ce 75	second parameter
ba 00	third parameter

Here is another example of a Ping message. This is a pong messages sent by the client to the server after a ping (type 0x6) message.

02	12 byte header, system channel
ff e3 6c	timestamp
00 00 06	message size in bytes
04	message type, Ping
00 00 00 00	stream ID (value 0.0)
00 07	ping type, pong
b4 dc	Second parameter
d5 00	third parameter

More Ping examples. This I saw after a Stream was deleted. At other times the current time was 00 01 00 00 00 01

02	12 byte header, system channel
00 00 00	timestamp
00 00 06	message size in bytes
04	message type, Ping
00 00 00 00	stream ID (value 0.0)
00 01	ping type, clear stream
00 00	Second parameter
00 02	third parameter

Other Message Types

03	RTMP 12 byte header

00 00 00	timestamp
00 00 bf	message size in bytes
14	message type, Invoke
00 00 00 00	stream ID (value 0.0)

An AMF Object uses properties to store its data. Each property has a name, followed by the data, which can be of any type. All of these messages are in the AMF0 format. AMF3 introduced a more compact encoding to save on bandwidth, but AMF0 is the default unless an RTMP message to switch to AMF3 is received.

NetConnection.connect() Message

03 00 00 01 00 01 23 14 00 00 00 00

02 00 07 63 6f 6e 6e 65 63 74	“connect”
00 3f f0 00 00 00 00 00 00	Connection ID (value 1.0)
03	(start of object)
00 03 61 70 70	“app”
02 00 08 6f 66 6c 61 44 65 6d 6f	“oflaDemo ”
00 08 66 6c 61 73 68 56 65 72	“flashVer”
02 00 0c 4c 4e 58 20 39 2c 30 2c 33 31 2c 30	“LNX 9,0,31,0”
00 06 73 77 66 55 72 6c	“swfUr l”
02 00 36 68 74 74 70 3a 2f 2f 31 39 32 2e 31 36 38 2e 31 2e 37 30 2f 73 6f 66 74 77 61 72 65 2f 67 6e 61 73 68 2f 74 65 73 74 73 2f 6f 66 6c 61 5f 64 65 6d 6f 2e 73 77 66	“http://192.1 68.1.70/software /gnash/tests/ofl a_demo.swf”
00 05 74 63 55 72 6c	“tcUrl”
02 00 19 72 74 6d 70 3a 2f 2f 6c 6f 63 61 6c 68 6f 73 74 2f 6f 66 6c 61 44 65 6d 6f	“rtmp://localhost/oflaDemo”
00 04 66 70 61 64	“fpad”
01 00	Boolean False
00 0b 61 75 64 69 6f 43 6f 64 65 63 73	“audioCodecs”
00 40 83 38 00 00 00 00 00	(a double)

00 0b 76 69 64 65 6f 43 6f 64 65 63 73	“videoCodecs”
00 40 5f 00 00 00 00 00 00	(a double) (value 124.0)
00 0d 76 69 64 65 6f 46 75 6e 63 74 69 6f 6e	“videoFunction”
00 3f f0 00 00 00 00 00 00	(a double) (value 1.0)
00 07 70 61 67 65 55 72 6c	“page Url”
02 00 27 68 74 74 70 3a 2f 2f c3 78 38 6 2d 75 62 75 6e 74 75 2f 73 6f 66 74 77 61 72 65 2f 67 6e 61 73 68 2f 74 65 73 74 73 2f	“http://x8 6-ubuntu/software/gnash/tests/”
00 00 09	(end of object)

Successful NetConnection.connect() Message

02 00 07 5f 72 65 73 75 6c 74	"_result"
00 3f f0 00 00 00 00 00 00 05	Connection ID (value 1.0)
03	(start object)
00 06 66 6d 73 56 65 72	"fmsVer"
02 00 0e 46 4d 53 2f 33 2c 30 2c 30 2c 31 31 35 37	"FMS/3,0,0,1157"
00 0c 63 61 70 61 62 69 6c 69 74 69 65 73	"capabilities"
00 40 3f 00 00 00 00 00 00	(a double) (value 31.0)
00 00 09	(end of object)
03	(start object)
00 05 6c 65 76 65 6c	"level"
02 00 06 73 74 61 74 75 73	"status"
00 04 63 6f 64 65	"code"
02 00 1d 4e 65 74 43 6f 6e 6e 65 63 74 69 6f 6e 2e 43 6f 6e 6e 65 63 74 2e 53 75 63 63 65 73 73	"NetConnection.Connect.Success"
00 0b 64 65 73 63 72 69 70 74 69 6f 6e	"description"
02 00 15 43 6f 6e 6e 65 63 74 69 6f 6e 20 73 75 63 63 65 65	"Connection succeeded."

64 65 64 2e	
00 0e 6f 62 6a 65 63 74 45 6e 63 6f 64 69 6e 67	"objectEncoding"
00 00 00 00 00 00 00 00 00	(a double) (value 0.0)
00 00 09	(end of object)

I've also noticed FMS 3 in this example sends objectEncoding, but Red5 doesn't, so I assume it's an optional field. Digging around, I found the http://videolectures.net/ site, which appears to not be using FMS 3, but Red5. The result messages from this site differ from the previous example, the main difference being the FMS 3 result mesage has the version object and the objectEncoding also specified.

I also another difference between Red5 and FMS in how the result messages are constructed. It appears that the order of the properities isn't important. Red5 like to specify the “application” as the name of the object, whereras FMS doesn't. Both work, so I assume the object name is optional.

Here's the FMS version:

03 00 00 00 00 00 73 14 00 00 00 00

02 00 07 5f 72 65 73 75 6c 74	"_result"
00 3f f0 00 00 00 00 00 00	Connection ID (value 1.0)
05	Server source ?
03	(start object)
00 05 6c 65 76 65 6c	"level"
02 00 06 73 74 61 74 75 73	"status"
00 04 63 6f 64 65	"code"
02 00 1d 4e 65 74 43 6f 6e 6e 65 63 74 69 6f 6e 2e 43 6f 6e 6e 65 63 74 2e 53 75 63 63 65 73 73	"NetConnection.Connect.Success"
00 0b 64 65 73 63 72 69 70 74 69 6f 6e	"description"
02 00 15 43 6f 6e 6e 65 63 74 69 6f 6e 20 73 75 63 63 65 65 64 65 64 2e	"Connection succeeded."
00 00 09	(end of object)

And here's the Red5 version:

03 00 00 00 00 00 81 14 00 00 00 00

02 00 07 5f 72 65 73 75 6c 74	"_result"
00 3f f0 00 00 00 00 00 00	Connection ID (value 1.0)
05	Server source ?
03	(start object)
00 0b 61 70 70 6c 69 63 61 74 69 6f 6e	"application"
05	
00 05 6c 65 76 65 6c	"level"
02 00 06 73 74 61 74 75 73	"status"
00 0b 64 65 73 63 72 69 70 74 69 6f 6e	"description"
02 00 15 43 6f 6e 6e 65 63 74 69 6f 6e 20 73 75 63 63 65 65 64 65 64 2e	"Connection succeeded."
00 04 63 6f 64 65	"code"
02 00 1d 4e 65 74 43 6f 6e 6e 65 63 74 69 6f 6e 2e 43 6f 6e 6e 65 63 74 2e 53 75 63 63 65 73 73	"NetConnection.Connect.Success"
00 00 09	(end of object)

Bandwidth Checking Message

After a successful connection is made, and the Connection.Succeeded message is returned, the server sends this message, which sets a callback for an optionally used method to do bandwidth checking. This sets it to the default of “undefined”.

03 00 00 00 00 00 15 14 00 00 00 00

02 00 08 6f 6e 42 57 44 6f 6e 65	"onBWDone"
00 00 00 00 00 00 00 00 00	(a double) (value 0.0)
05	

NetConnection.close() Message

03 00 00 00 00 00 12 14 00 00 00 00

02 00 05 63 6c 6f 73 65	"close"
00 00 00 00 00 00 00 00 00	(a double) (value 0.0)
05	

Failed NetConnection.connect() Message

I've also found variations betwen Red5 and FMS in how the connection failed error message is constructed. FMS appears to create the error message as a named object, and uses a 12 byte header. Red5 uses the 8 byte header for thwe same message, and it also contains less properties.

Here's the FMS version:

02 00 07 5f 72 65 73 75 6c 74	"_result"
00 3f f0 00 00 00 00 00 00	Connection ID (value 1.0)
05	
03	(start object)
00 0b 61 70 70 6c 69 63 61 74 69 6f 6e	"application"
05	server ?????
00 05 6c 65 76 65 6c	"level"
02 00 05 65 72 72 6f 72	"error"
00 0b 64 65 73 63 72 69 70 74 69 6f 6e	"description"
02 00 00	(null string)
00 04 63 6f 64 65	"code"
02 00 1c 4e 65 74 43 6f 6e 6e 65 63 74 69 6f 6e 2e 43 6f 6e 6e 65 63 74 2e 46 61 69 6c 65 64	"NetConnection.Connect.Failed"
00 00 09	(end of object)

And here's the Red5 version:

43	RTMP 8 byte header
00 00 00	timestamp
00 00 48	message size in bytes
14	message type, Invoke
02 00 06 5f 65 72 72 6f 72	“error”

00 40 00 00 00 00 00 00 00	Connection ID (value 2.0)
05	
03	(start object)
00 04 63 6f 64 65	“code”
02 00 19 4e 65 74 43 6f 6e 6e 65 63 74 69 6f 6e 2e 43 61 6c 6c 2e 46 61 69 6c 65 64 00 05 6c 65 76 65 6c	“NetConnection.Call.Failed”
02 00 05 65 72 72 6f 72	“error”
00 00 09	(end of object)

NetStream.createStream() Message

43	RTMP 8 byte header
00 00 00	timestamp
00 00 1d	message size in bytes
14	message type
02 00 0c 63 72 65 61 74 65 53 74 72 65 61 6d	“createStream”
00 40 08 00 00 00 00 00 00	Stream ID (value 3.0)
05	unknown

4d	RTMP 8 byte header
00 05 1d	timestamp
00 00 18	message size in bytes
14	message type, Invoke
02 00 0b 63 6c 6f 73 65 3 74 72 65 61 6d	closeStream
00 00 00 00 00 00 00 00 00	Stream ID (value 0.0)
05	unknown

NetStream.Reset() Message

44	RTMP 8 byte header

00 00 00	timestamp
00 00 a8	message size in bytes
14	message type, Invoke
02 00 08 6f 6e 53 74 61 74 75 73	onStatus
00 3f f0 00 00 00 00 00 00	Connection ID (value 1.0)
05	
03	
00 08 63 6c 69 65 6e 74 69 64	clientid
00 3f f0 00 00 00 00 00 00	Connection ID (value 1.0)
00 05 6c 65 76 65 6c	level
02 00 06 73 74 61 74 75 73	status
00 07 64 65 74 61 69 6c 73	details
02 00 0e 4f 46 4c 41 5f 50 52 4f 4d 4f 2e 66 6c 76	OFLA_PROMO.flv
00 0b 64 65 73 63 72 69 70 74 69 6f 6e	description
02 00 25 50 6c 61 79 69 6e 67 20 61 6e 64 20 72 65 73 65 74 74 69 6e 67 20 4f 46 4c 41 5f 50 52 4f 4d 4f 2e 66 6c 76 2e	Playing and resetting OFLA_PROMO.flv.
00 04 63 6f 64 65	code
02 00 14 4e 65 74 53 74 72 65 61 6d 2e 50 6c 61 79 2e 52 65 73 65 74	NetStream.Play.Reset
00 00 09	(end of object)

Result Message

43	RTMP 8 byte header
00 00 00	timestamp
00 00 1d	message size in bytes
14	message type, Invoke
02 00 07 5f 72 65 73 75 6c 74	“_result”
00 40 08 00 00 00 00 00 00	Stream ID (value 3.0)

05	unknown
00 3f f0 00 00 00 00 00 00	Connection ID (value 1.0)