The AMF

The AMF object is used for the sending and receiving of all video/audio data. After the initial handshake, all objects sent use the AMF structure. Once the AMF structure is understood, the conversation between the client and the server can be understood.

Below is a connection request sent by the Flash Player. This is the first AMF object sent by the Flash Player after the initial handshake and is a product of the Action Script call NetConnection.Connect( ).

THE AMF

```
03 00 00 01
00 01 05 14 00 00 00 00   02 00 07 63 6f 6e 6e 65
63 74 00 3f f0 00 00 00   00 00 00 03 00 03 61 70
70 02 00 16 73 61 6d 70   6c 65 5f 76 69 64 65 6f
63 6f 6e 66 65 72 65 6e   63 65 00 08 66 6c 61 73
68 56 65 72 02 00 0c 57   49 4e 20 37 2c 30 2c 31
39 2c 30 00 06 73 77 66   55 72 6c 02 00 78 66 69
6c 65 3a 2f 2f 5c 5c 77   6e 6d 69 68 65 72 72 65
73 30 33 5c 63 24 5c 44   6f 63 75 6d 65 6e 74 73
20 61 6e 64 20 53 65 74   c3 74 69 6e 67 73 5c 6d
69 63 6b 2e 68 65 72 72   65 73 2e 4d 49 43 4b 44
4f 4d 41 49 4e 5c 4d 79   20 44 6f 63 75 6d 65 6e
74 73 5c 46 6c 61 73 68   20 50 72 6f 6a 65 63 74
73 5c 56 69 64 65 6f 52   65 70 72 61 63 74 69 63
69 6e 67 2e 73 77 66 00   05 74 63 55 72 6c 02 00
2a 72 74 6d 70 3a 2f 2f   31 39 32 2e 31 36 38 2e
32 2e 31 34 2f 73 61 6d   70 6c 65 5f 76 69 64 65
6f 63 6f 6e 66 65 72 65   6e c3 63 65 00 00 09
```

Initial Header Byte Bits

```
0000 0011 -> 0x03
```

Using the color coding above, the AMF object is broken out as follows:

- `03`
  - This is the initial header byte of the AMF header.
  - It defines 2 things
    - The size of the AMF header.
      - `00` -- The first 2 significant bits of the initial header byte define the size of the header.

            - `00` = 12 bytes
            - `01` = 8 bytes
            - `10` = 4 bytes
            - `11` = 1 byte
        - The AMF number.
            - `00 0011` = 3 (as in `0x03` from above)
            - A single TCP socket connection between the Flash Player and the server will send multiple AMF objects back and forth between the client and the server. AMF objects are not queued, but are interweaved together so they show up at the other end at the same time.
            - At any given moment, a maximum of 64 AMF objects (`11 1111 -> 0xcf`) can be exchanged between the client and the server.
            - AMF numbers are re-used when the AMF object they are representing has been fully transmitted.
    - **This byte is always present, regardless of header size**
    - *For this AMF, the header size is 12 and the AMF number is 3*
- `00 00 01`
    - These bytes are unknown but are always in the same position and are always 3 bytes long. They can be transferred over when sending the video/audio data (via AMF) to the subscribers of the video/audio stream.
    - **These bytes are sent when the header size is 4 or more**
- `00 01 05`
    - These bytes denote the size of the body of the AMF object. The body of the AMF are the bytes highlighted in brown (🟫) above.
    - **These bytes are sent when the header size is 8 or more.**
    - *The size of this AMF object is 261*
- `14`
    - This byte defines the content type of the AMF object.
        - 0x14 – Function Call
        - 0x09 – Video data
        - 0x08 – Audio data
    - **This byte is sent when the header size is 8 or more.**
    - *This AMF object is a Function Call.*
- `00 00 00 00`
    - These bytes define the source/destination of the AMF.
    - If the destination of the AMF is the Server, these bytes determine the source (which NetStream object) of the AMF object.
    - If the destination of the AMF is the client, these bytes determine which NetStream object should receive the AMF object.
    - Note: 00 00 00 00 is reserved for either the NetConnection object, or the Flash player itself. I suspect it is the NetConnection object.
        - The first NetStream to use this NetConnection object will have the following bytes in this location; 01 00 00 00.
    - **These Bytes are only sent when the header size is 12**

    - *The source of this AMF object is the NetConnection object*
- `02 00 07 63 … 65 00 00 09`
    - AMF Body Bytes.
        - These bytes are sent in 128 byte chunks.
        - Each chunk is either headed with the initial header or a 1 byte header (`c3` from above).
            - Note: `0xc3` follows the same initial header byte rules as the header byte of the entire AMF.
                - `0xc3 -> 1100 0011`
                    - `11` – Header size is 1
                    - `00 0011` – AMF number is 3
            - Note: The final chunk of bytes is less than 128
                - This is because the total body size, 261, does not divide evenly in to 128. There is a remainder of 5 bytes.

- The significance of the header size and its relation to the AMF number.
    - If the header size is 12
        - All header data is reset for the AMF number defined in the AMF. This includes all items from above; size, body type, source/destination.
        - *If the AMF number is 3, the initial header byte is* `0x03`
    - If the header size is 8
        - The source/destination is the same as the last AMF sent with the same AMF number
        - *If AMF number is 3, the initial header byte is* `0x43`
    - If the header size is 4
        - The source/destination is the same as the last AMF sent with the same AMF number
        - The AMF content type of the body of the AMF is the same as the last AMF sent with the same AMF number.
        - The AMF body is the same size as the last AMF sent with the same AMF number.
        - *If AMF number is 3, the initial header byte is* `0x83`
    - If the header size is 1
        - The source/destination is the same as the last AMF sent with the same AMF number
        - The AMF content type of the body of the AMF is the same as the last AMF sent with the same AMF number.
        - The AMF body is the same size as the last AMF sent with the same AMF number.
        - The mystery 3 bytes are the same as the last AMF sent with the same AMF number.
        - *If AMF number is 3, the initial header byte is* `0xc3`

And Example of 2 interweaved AMF objects.

```
03 00 00 01 00 01 05 14  00 00 00 00 02 00 07 63
6f 6e 6e 65 63 74 00 3f  f0 00 00 00 00 00 00 03
00 03 61 70 70 02 00 16  73 61 6d 70 6c 65 5f 76
69 64 65 6f 63 6f 6e 66  65 72 65 6e 63 65 00 08
66 6c 61 73 68 56 65 72  02 00 0c 57 49 4e 20 37
2c 30 2c 31 39 2c 30 00  06 73 77 66 55 72 6c 02
00 78 66 69 6c 65 3a 2f  2f 5c 5c 77 6e 6d 69 68
65 72 72 65 73 30 33 5c  63 24 5c 44 6f 63 75 6d
65 6e 74 73 20 61 6e 64  20 53 65 74 04 00 00 01
00 01 05 14 00 00 00 00  02 00 07 63 6f 6e 6e 65
63 74 00 3f f0 00 00 00  00 00 00 03 00 03 61 70
70 02 00 16 73 61 6d 70  6c 65 5f 76 69 64 65 6f
63 6f 6e 66 65 72 65 6e  63 65 00 08 66 6c 61 73
68 56 65 72 02 00 0c 57  49 4e 20 37 2c 30 2c 31
39 2c 30 00 06 73 77 66  55 72 6c 02 00 78 66 69
6c 65 3a 2f 2f 5c 5c 77  6e 6d 69 68 65 72 72 65
73 30 33 5c 63 24 5c 44  6f 63 75 6d 65 6e 74 73
20 61 6e 64 20 53 65 74  c3 74 69 6e 67 73 5c 6d
69 63 6b 2e 68 65 72 72  65 73 2e 4d 49 43 4b 44
4f 4d 41 49 4e 5c 4d 79  20 44 6f 63 75 6d 65 6e
74 73 5c 46 6c 61 73 68  20 50 72 6f 6a 65 63 74
73 5c 56 69 64 65 6f 52  65 70 72 61 63 74 69 63
69 6e 67 2e 73 77 66 00  05 74 63 55 72 6c 02 00
2a 72 74 6d 70 3a 2f 2f  31 39 32 2e 31 36 38 2e
32 2e 31 34 2f 73 61 6d  70 6c 65 5f 76 69 64 65
6f 63 6f 6e 66 65 72 65  6e c4 74 69 6e 67 73 5c
6d 69 63 6b 2e 68 65 72  72 65 73 2e 4d 49 43 4b
44 4f 4d 41 49 4e 5c 4d  79 20 44 6f 63 75 6d 65
6e 74 73 5c 46 6c 61 73  68 20 50 72 6f 6a 65 63
74 73 5c 56 69 64 65 6f  52 65 70 72 61 63 74 69
63 69 6e 67 2e 73 77 66  00 05 74 63 55 72 6c 02
00 2a 72 74 6d 70 3a 2f  2f 31 39 32 2e 31 36 38
2e 32 2e 31 34 2f 73 61  6d 70 6c 65 5f 76 69 64
65 6f 63 6f 6e 66 65 72  65 6e c3 63 65 00 00 09
c4 63 65 00 00 09
```

Based on the color coding and the rules defined above, it can be seen how AMFs are interweaved. *It is good to note that Video data and Function Calls send data in 128 byte chunks, while Audio data is sent in 64 byte chunks. Other than that, Audio data is sent in identical AMF object structures.*