moq
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2026

C. Jennings
Cisco

7 July 2025

Serialization of MoQ Objects to Files
draft-jennings-moq-file-04

Abstract

This specification provides a way to save the metadata about each MoQ Object in one or more files as well as pointers to other files that contain the contents of the object. Separating of the metadata and payload data allows the payload data to remain in files that are used for other purposes such as serving HLS/DASH video. This format makes it easier to test and develop caching relays and create test data they can serve to clients.

Table of Contents

1. Introduction

This specification defines a way of serializing the MoQ Objects defined in [MoQT] into files. The payload data and the metadata are separated into separate files to allows reuse of existing files with the payload data.

2. JSON Meta Object

The .moq files consist of an array of one or more JSON objects. Each JSON object contains information about the MoQT object as well as pointers to the where the original data can be found.

The following fields are defined for JSON object:

* trackNamespace: Array of strings that have a Base64 encoded version of the data in each tuple of MoQT Track Namespace as defined in [MoQT].

* trackName: string with Base64 encoded version of the MoQT Trackname as defined in [MoQT].

* objectID: integer corresponding to the MoQT Object ID as defined in [MoQT].

* groupID: integer corresponding to the MoQT Group ID as defined in [MoQT].

* subGroupID: integer corresponding to the MoQT Subgroup as defined in [MoQT].

* forwardingPref: String with value of "Subgroup" or "Datagram" to represent the Object Forwarding Preference as defined in [MoQT]. Open Issue: string or use the binary values used in spec?

* objectStatus: Numeric value representing Object Status as defined in [MoQT].

* publisherPriority: integer corresponding to the MoQT Publisher Priority as defined in [MoQT].

* maxCacheDuration: integer corresponding to the MoQT publisher MAX CACHE DURATION Parameter as defined in [MoQT].

* publisherDeliveryTimeout: integer corresponding to the MoQT DELIVERY TIMEOUT Parameter sent by the publisher as defined in [MoQT].

* receiveTime: time original object was created (if known) or time object was received by the relay. This is saved as an integer in milliseconds since the unix epoch which is 00:00:00 UTC on January first, 1970.

* dataFile: string with relative path name to the file that stores the MoQT Object, including header and its payload data.

* dataOffset: number of bytes into file where the objects starts (0 is first byte of file)

* dataLength: number of bytes of data in the object

Any Object Extension Headers, as defined in [MoQT], should also be saved using a field name formed by the string "ext" then the base 10 integer representation of the extension type with a value that is the Base64 encoded version of the extension header data. Open Issue: this will not preserve the order of the extension headers. Is that a problem?

3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This specification uses the terminology defined in [MoQT].

3.1. Base64 Encoding

The Base64 encoding used in this specification is the "Base64 Encoding with URL and Filename Safe Alphabet" as defined in [RFC4648], Section 5. Additionally, the encoding is performed without any padding or extra blank space characters

3.2. File Name Encoding

The filename SHOULD be formed by percent encoding each tuple of the namespace and track. Each tuple of the namespace is separated with a period and the last tuple is separated from the track name with a dash. The percent encoding users a percent symbol followed by two lower case hex digits for any characters outside the range of 0 to 9, a to z, or A to Z.

For example, a names of of (Foo, b+r) with track name of ex1 would be encoded as: Foo.b%2br-ex1

If the filename only has data for a subset of the track, a plus sign followed by number of first group ID represented by the file SHOULD be appended.

4. MoQT Track DataFile

When saving a whole MoQT Track to a file, a common way to do this would be to make one ".dat" file with all the object data and another ".moq" file with all the array of JSON object for each MoQT Object. An implementation can choose to have one file per MoQT group. In such a case, it does so by creating one metadata (".moq") file and one datafile (".dat") containing data for each object in the MoQT group.

5. Playback

Some use cases will want to just load a file into the relay as quickly as possible. Others may decide to rename the track name to a new track name and publish the objects at a rate based on differences of the receiveTime of the JSON objects.

6. Example

TODO More complete example

6.1. Time Object Example

Data file named time1.dat contains:

{"time":17294570764566}

Metadata file contains:

```
[
  {
    "namesSpace": [ "bW9xLXRpbWUuYXJwYQo=" , "dGltZS12MQo=" ],
    "trackName": "bWFjOjcyOjVjOmYwOjdjOmJmOmIw",
    "objectID": 0,
    "groupID": 123,
    "subGroup": 0,
    "publisherPriority": 0,
    "maxCacheDuration": 3600000,
    "publisherDeliveryTimeout": 60000,
    "receiveTime": 1729457464000,
    "dataFile": "time1.dat",
    "dataOffset": 0,
    "dataLength": 25
  }
]
```

7. IANA

TODO file extension registrations.

8. Security Considerations

TODO

9. Acknowledgements

Thank you to Suhas Nandakumar, Tomas Rigaux, Carsten Bormann and Gwendal Simon for their reviews and suggestions.

10. Normative References

[MoQT] Nandakumar, S., Vasiliev, V., Swett, I., and A. Frindell, "Media over QUIC Transport", Work in Progress, Internet-Draft, draft-ietf-moq-transport-12, 23 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-moq-transport-12>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

Author’s Address

Cullen Jennings
Cisco
Canada
Email: fluffy@iii.ca