moq
Internet-Draft
Intended status: Informational
Expires: 23 September 2026

L. Curley
22 March 2026

MoQ Largest Group Extension
draft-lcurley-moq-largest-group-00

Abstract

This document defines a Largest Group subscription filter type for MoQ Transport [moqt]. A subscriber uses this filter to request delivery starting from the first object of the publisher's largest (most recent) group, ensuring a complete group is received.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Media Over QUIC Working Group mailing list (moq@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/moq/.

Source for this draft and an issue tracker can be found at https://github.com/kixelated/moq-drafts.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 23 September 2026.

Copyright Notice

Table of Contents

1. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Introduction

[moqt] Section 5.1.2 defines four subscription filter types that control where delivery begins:

* *Next Group Start (0x1)*: Starts at {Largest.Group + 1, 0}, skipping the remainder of the current group entirely.

* *Largest Object (0x2)*: Despite the name, starts at {Largest.Group, Largest.Object + 1} — the object _after_ the largest, not the largest object itself.

* *AbsoluteStart (0x3)*: Starts at an explicit location specified by the subscriber.

* *AbsoluteRange (0x4)*: Starts and ends at explicit locations specified by the subscriber.

There is no filter that starts from the _beginning_ of the current group. Next Group Start skips the current group entirely, potentially adding latency equal to the group duration. Largest Object starts mid-group, delivering objects that may depend on earlier objects in the same group.

Objects within a group are typically delta encoded (ex. video GOPs), so arbitrary objects in the middle of a group are undecodable without prior objects. A subscriber using Next Group Start avoids this problem but must wait for the next group to begin, unnecessarily increasing join latency.

2.1. Joining Fetch Workaround

[moqt] does provide a workaround: a subscriber can issue a separate "joining" FETCH request alongside a SUBSCRIBE to retrieve the beginning of the current group. However, this approach has several drawbacks:

* *Complexity*: Libraries emulate Largest Group by coordinating a FETCH and SUBSCRIBE, splitting the group across multiple streams. This requires merging the results into a single coherent group and handling various edge cases, such as one of the two requests failing independently.

* *Head-of-line blocking*: If the group contains multiple sub-groups, the FETCH delivers them sequentially over a single stream, introducing head-of-line blocking that negates the benefits of sub-group parallelism.

* *Priority*: Everything should be delivered in dependency order to improve startup time and avoid potential flow control deadlocks. This requires prioritizing the FETCH higher than the SUBSCRIBE, which may be non-obvious or unsupported.

This extension avoids these issues by defining a Largest Group filter that starts delivery from the first object of the publisher’s most recent group within the SUBSCRIBE itself, ensuring a complete group is delivered over the normal subscription path. Additionally, the first group of a subscription behaves the same as any other group.

3. Setup Negotiation

The Largest Group extension is negotiated during the SETUP exchange as defined in [moqt] Section 9.4.

Both endpoints indicate support by including the following Setup Option:

```
LARGEST_GROUP Setup Option {
  Option Key (vi64) = TBD1
  Option Value Length (vi64) = 0
}
```

If both endpoints include this option, the Largest Group filter is available for the session. If a peer receives a SUBSCRIBE containing the Largest Group filter without having negotiated the extension, it MUST close the session with a PROTOCOL_VIOLATION.

4. Largest Group Filter

This document defines a new subscription filter type for use in the Subscription Filter field of SUBSCRIBE messages as defined in [moqt] Section 5.1.2.

4.1. Filter Type

```
Subscription Filter {
  Filter Type (vi64) = 0x20
}
```

The Largest Group filter uses Filter Type value 0x20. No additional fields (Start Location or End Group Delta) are present in the Subscription Filter.

4.2. Semantics

When the publisher receives a SUBSCRIBE with the Largest Group filter, it computes the start location as:

```
{Largest.Group, 0}
```

Where Largest.Group is the group sequence number of the largest object known to the publisher at the time the SUBSCRIBE is processed. Delivery begins from the first object (object 0) of that group.

The subscription is open-ended: the publisher continues delivering subsequent groups until the subscription is cancelled or the session ends. This is consistent with the behavior of Next Group Start and Largest Object filters.

The largest group is delivered using normal SUBSCRIBE semantics. If earlier objects in the group have been evicted from the cache, the publisher MAY attempt to repopulate the cache or simply drop the group as it would for any other subscription.

5. Security Considerations

This extension introduces no new security considerations beyond those described in [moqt]. The publisher already tracks group state; this filter uses existing state to compute the start location.

6. IANA Considerations

This document requests the following registrations:

6.1. MoQ Setup Option Types

This document registers the following entry in the "MoQ Setup Option Types" registry established by [moqt]:

Value	Name	Reference
TBD1	LARGEST_GROUP	This Document

Table 1

6.2. MoQ Subscription Filter Types

This document registers the following entry in the "MoQ Subscription Filter Types" registry established by [moqt]:

Value	Name	Reference
0x20	LARGEST_GROUP	This Document

Table 2

7. 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-17, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-moq-transport-17>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Acknowledgments

This document was drafted with the assistance of Claude, an AI assistant by Anthropic.

Author’s Address

Luke Curley
Email: kixelated@gmail.com