



# Extensibility

Boulder, Feb 2026  
Slides by Ian Swett

# Open Extensibility Issues ([list](#))

[#1184](#) TV vs TLV: Our Parameters know no length

[#1407](#) Unknown Setup Parameters, Extension Negotiation

[#855](#), [#1268](#) Object Extension Headers

Global space or Application Specific range?

[#1231](#), [#1175](#) Are Error Codes best effort or mandatory?

Handling undefined error codes

## Editorial

[#1181](#) Naming: Extension Headers or just Headers?

## Greasing: Fix after others

[#416](#) Extensibility and Greasing (PR [#1460](#) for SETUP)

## TLV to TV: Our Parameters know no length [#1184](#)

Message Parameters today use even/odd to indicate type

We require them to be understood and hop-by-hop, so you aren't supposed to skip ones you don't know about.

Switching format to TV ensures you know about them and saves a few bytes

Also allows us to optimally allocate the 1-byte space

PR [#1462](#)

# Unknown Setup Parameters Options and Extension Negotiation [#1407](#)

Editorial PR [#1461](#) Renames Setup Parameters to Setup Options, and clarifies Setup Options are separate from Message Parameters

Setup ~~Parameters~~ Options used to negotiate extensions

PR #TODO Specifies how we expect Message Parameters, Message types, Extension Headers, Error Codes, and new Stream types to be negotiated during Setup.

Also discusses how pipelining does and doesn't work with extensibility.

## #855, #1268 Extension Allocations - Parameters

Is the Extension Header Type space global?

What range(s) do application params and headers use?

Proposal: For *Message Parameters*, space is not that critical, so a QUIC style ([RFC 9000](#)) IANA registry should work well.

SHOULD randomly select from appropriate range

SHOULD select sequential values to improve  
delta-encoding efficiency

## #855, #1268 Extension Header Allocations

Track and Object *Extension Headers* are more frequently used than Message Parameters

Goals: Reduce chances of collision and wire overhead  
Compatibility with Filters

Option 1: Use QUIC style approach like Parameters  
Unambiguous what Extension Header is present  
Can consume 1-3 extra bytes per Track or Object

Option 2: Reserve 8 1-byte & 2048 2-byte codepoints for Applications #1473  
Different Applications Can use the same codepoints

## #1175, #1231 Error Code Handling - Session Errors

If a peer closes the *Session* with an unknown error,  
nothing you can do

That being said, we do say: "MUST treat as a connection error of type X"

Proposal: Stick with the current approach, clarify the draft

# #1175, #1231 Error Code Handling - Request Errors

#1339 Added a Request Retry Interval

Request Errors don't specify responses today

Extensions could negotiate new Request Errors

An unknown Request Error closing a session needs caution

Options:

1. Specify the Error to use, but permit others - Best Effort, similar to QUIC and HTTP/3
  - a. Specify a default behavior if error code is unknown
2. Only allow negotiated Request Errors
3. Something else?

# #1181 Naming Round! Extension Headers?

Currently have 2 types of *Extension Headers*:

Track Extension Headers

Object Extension Headers

No one loves *Extension Headers*, but what's better?

Some are defined in the core draft, not extensions

- Headers: Too similar to HTTP?
- MoQT Headers: Track MoQT Headers & Object MoQT...
- Properties: Too similar to Message Parameters?
- Attributes: ?
- Extensions: ?
- Other Ideas?