19 Nov 2026

IETF 127

draft-ietf-mlcodec-opus-extension



Draft Status

- No changes since -06

Second WGLC

- Only comments on the list were in support of publishing
- But Jean-Marc ran into an issue bringing the repacketizer up-to-date to handle extensions...

Repacketizer

- The repacketizer splits an Opus packet into individual frames and re-combines them into new packets with different amounts of frames
- Extensions are encoded per-packet, but semantically belong to individual frames
- Repacketizer must parse the extensions for each input packet and generate a new encoding for each output packet

Repacketizer Resource Limits

- The RTE mechanism is compression, and like any good compression, can generate a *lot* of output with relatively little input
- A naive parsed extension representation might look like

Field	**Size (bytes)**
ID	1
Frame	1
Payload (Pointer)	8 (on 64-bit system)
Length	4
Total	14 (16 with alignment)

Repacketizer Resource Limits (2)

- A repeated zero-byte extension allows a single input byte to expand to 48 16-byte extension descriptors, for a **multiplication factor of 768**
- 20 kB of extension data (a reasonable size for the Scalable Quality extension, on par with lossless) could expand to **15 MB of descriptors**
- To avoid the use of malloc on embedded systems that might not have libc, libopus was trying to put this on the stack...

Options (1)

- Let implementations impose some reasonable cap on the total number of extensions
  - What is reasonable and future-proof?
- We have a compact representation of extensions: the wire format
  - We can just iterate over that
  - But the extensions do not appear in the order required, so we will have to iterate over some multiple times
  - Worst case O(N*nb_frames*nb_frames), where N is the input size (up to 300 ms with an AI-generated implementation attempt)

Options (2)

- Special-case zero-byte extensions in the implementation
  - Collapse zero-byte extensions to a single descriptor that spans multiple frames
  - Requires re-ordering iterator output to maintain relative order within each frame
  - Space expansion reduced from 768 to 16 (or better with more aggressive struct optimization), but still needs a limit
  - O(N*nb_frames)
  - Implementation adds a fair amount of complexity (a simple matter of typing…)
    - Draft patch at https://gitlab.xiph.org/xiph/opus/-/merge_requests/206

Options (3)

- Ignore multiple instances of a zero-byte extension
  - Special-case semantics for *all* extension uses
  - Loses some possible expressivity
  - Not possible to enforce in the wire format (without redesigning it)
- Discard *all* unknown extensions in repacketizer
  - Requires libopus upgrades to handle any new extension
  - Will still have to deal with this some day
- Other proposals?

Summary

- Options
  - Impose some reasonable cap on the total number of extensions (future-proofing concerns)
  - Iterate over existing packed extension format (slow)
  - Special-case zero-byte extensions in implementation (complex)
  - Ignore multiple zero-byte extensions in format (broader impact)
  - Discard all unknown extensions in repacketizer (can-kicking)
  - Others?

Opus Extension Format

