Draft Status

- No change since Prague
  - 02 submitted to refresh expiry
Opus Extension Format

- TOC Byte (code 3)
- Padding Bit
- Opus Frame Length(s)
- Padding (extensions go here)
- Frame Count Byte (variable)
- Compressed Opus Frames
- Extensions for Frame 0
- Length for ID=32...127, L=1 (variable)
- ID #1
- L Flag
- Extension Payload
- Separator (ID=1)
- Extensions for Frame 1
- Separator (ID=1)
- Frame Increment (optional)
- Etc.
Overhead
Single-frame: 3 bytes (minimum)
Overhead

Multi-frame: 2 bytes each (minimum)

TOC Byte (code 3)
Padding Bit
Opus Frame Length(s)
Padding (extensions go here)
Compressed Opus Frames

Frame Count Byte (variable)
Padding Length

Extensions for Frame 0

ID #1
Length for ID=32...127, L=1 (variable) ID #2...

L Flag
Extension Payload

Separator (ID=1)

Extensions for Frame 1

Frame Increment (optional)

Separators (ID=1)

Etc.
Example:
60 ms packet with 3×20 ms frames

- Short (1-byte) extension on each frame
  - 3×2 bytes of overhead
  - 3×1 bytes of payload
- Total: 9 bytes
Proposition: Repeat These Extensions (RTE)

- Additional mandatory-to-implement extension (ID=2)
- When encountered in current frame C
  - For each frame F after frame C in the same packet
    - For each extension E already seen in frame C (after any previous RTE extension)
      - Decode an extension payload of type E for frame F
        - Short extension: Use same L flag as the original
        - Long extension: Use L=1 (explicit length), unless RTE extension had L=0 and this is the last repeated extension
  - If RTE extension had L=1, continue decoding extensions for frame C
  - If RTE extension had L=0 and did not end with a long extension, continue decoding extensions for frame C+1
    - If frame C was already the last frame, zero pad to the end of the packet
Example Revisited:  
60 ms packet with 3×20 ms frames

- Short (1-byte) extension on each frame
  - 1 byte for padding length
  - 1 byte for extension ID
  - 1 byte for RTE extension
  - 3×1 bytes of payload

- Total: 6 bytes
Summary

● Reasons to do this
  - Can reduce overhead even with just 1 extension appearing in 2 frames
  - Savings scale with the number of frames and repeated extensions
  - Applies to any extension: no extra IDs to register or SDP signaling
  - Integrates well with non-repeated extensions (e.g., DRED)
  - Doing it later would be a breaking change to extension parsing

● Reasons not to do this
  - Additional implementation complexity (entirely optional for encoder)
  - Extensions for a frame no longer guaranteed to be physically contiguous
Extension ID numbering:
Strawman Proposal

- Split Extension ID space into “Short” and “Long” extensions

<table>
<thead>
<tr>
<th>Ext. Byte (B)</th>
<th>ID(s)</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0...1</td>
<td>0</td>
<td>(B &amp; 1) → 0 = rest, 1 = coded</td>
</tr>
<tr>
<td>2...3</td>
<td>1</td>
<td>(B &amp; 1)</td>
</tr>
<tr>
<td>4...5</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>6...7</td>
<td>N/A</td>
<td>Reserved</td>
</tr>
<tr>
<td>8...63</td>
<td>a0...a55</td>
<td>(B &amp; 3)</td>
</tr>
<tr>
<td>64...255</td>
<td>b0...b95</td>
<td>(B &amp; 1) → 0 = rest, 1 = coded</td>
</tr>
</tbody>
</table>
Questions?

- How do we get more reviews and feedback?