ICNRG
Internet-Draft
Updates: 8569, 8609 (if approved)
Intended status: Experimental
Expires: 11 August 2026

M.E. Mosko

H. Asaeda
NICT

7 February 2026

CCNx Content Object Chunking
draft-irtf-icnrg-ccnxchunking-04

Abstract

This document specifies a chunking protocol for dividing a user payload into CCNx Content Objects. It defines a name segment type to identify each sequential chunk number and a Content Object field to identify the last available chunk number. This includes specification for the naming convention to use for the chunked payload and a field added to a Content Object to represent the last chunk of an object. This document updates RFC8569 and RFC8609.

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 11 August 2026.

Copyright Notice

Table of Contents

1. Introduction

CCNx Content Objects [RFC8569] are sized to amortize cryptographic operations over user data while simultaneously staying a reasonable size for transport over today's networks. This means a Content Object is usually within common UDP or jumbo Ethernet size. If a publisher has a larger amount of data to associate with a single Name, the data may be chunked with this chunking protocol. This protocol uses state in the Name and in an optional field within the Content Object. A chunked object may also have an external metadata that describes the original pre-chunked object.

For example, a video file may be several gigabytes of data. To publish the video, one would divide it up into transport-sized Content Objects. Each Content Object would have a common name prefix plus a chunk number.

CCNx uses two types of messages: Interests and Content Objects [RFC8569]. An Interest carries the hierarchically structured Name of a Content Object which serves as a request to return that object. A Content Object contains the Name, the object's Payload, and the cryptographic information used to bind the Name to the payload.

This document introduces three new protocol elements to CCNx by updating [RFC8609].

ChunkNumber A new CCNx Name Segment TLV for conveying chunk

numbers.

EndChunkNumber A new CCNx Message Type TLV for conveying the last chunk number of the content.

ChunkSize A new CCNx Message Type TLV for specifying a constant payload size.

Experiments employing ICN protocols like CCNx all require a consistent way to represent objects larger than can be carried as a single protocol message. This specification provides such capabilities for simple cases where the flexibility of a Manifest approach like that in FLIC [I-D.irtf-icnrg-flic] are not strictly needed. Experiments will shed light on how to best optimize the fetching of large objects using CCNx Interest/Data exchanges.

2. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. CCNx Chunking

3.1. Protocol Specification

Chunking, as used in this specification, means serializing user data into one or more chunks, each encapsulated in a CCNx Content Object. A chunk is a contiguous byte range within the user data. One segment in the Name of each Content Object represents the chunk number. A field in the Content Object that a chunk is the last chunk of the object Chunks are denoted by a serial counter, beginning at 0 and incrementing by 1 for each contiguous chunk. Any Interest issued with a chunk number higher than the last chunk number MUST be dropped with no response.

This document introduces the new CCNx Name Segment TLV, ChunkNumber, for conveying chunk numbers. The ChunkNumber is the serial order of the chunks. It MUST begin at 0 and MUST be incremented by 1. The value of the ChunkNumber TLV is a network byte order compact encoding of the chunk number.

This document also introduces the new CCNx Message Type TLV, EndChunkNumber. The EndChunkNumber indicates the last chunk number of the chunked user data. It MUST be included in the Content Object which is the last chunk of user data, where the ChunkNumber TLV value

and the EndChunkNumber TLV value MUST be the same. The value of the EndChunkNumber value is the same format as the ChunkNumber name segment. For example, if 3000 bytes of user data is split with a 1200 byte block size, there will be 3 chunks: 0, 1, and 2. The EndChunkNumber is 2.

The EndChunkNumber field MAY be present in any of the object's chunks, such as to assist in pipelining Interests. It MAY be placed repeatedly across multiple chunks to maximize the fetching flexibility of a consumer.

The value of the EndChunkNumber MAY be increased by subsequent EndChunkNumber fields. However, it SHOULD NOT be decreased. If a publisher wishes to terminate creation of chunks before reaching the last chunk, it should publish empty Content Objects to fill out to the maximum EndChunkNumber ever published. These padding chunks MUST contain the real last chunk number in the EndChunkNumber with an empty Payload TLV. This mechanism for early termination is to accommodate circumstances where a data creation by the producer ended before expected.

It may occur that a publisher never includes an EndChunkNumber, and thus has never published the last chunk. This may happen due to an error or network conditions that do not allow finding the last chunk. If a consumer times out trying to retrieve chunks, it SHOULD report an error to the user and terminate. This is similar to a TCP client not hearing a FIN.

If the user data fits within one Content Object and the publisher uses the Chunking protocol, the publisher names the content with "Chunk=0" and includes "EndChunkNumber=0" (assuming the Payload is in the first chunk).

In addition, this document introduces the new CCNx Message Type TLV, ChunkSize, for specifying a constant payload size. Chunking MAY use a fixed block size, where only the last chunk MAY use a smaller block size. This is required to allow a reader to seek to a specific byte offset once it knows the block size. If the first (Chunk 0) object has the field ChunkSize, then the series of chunked objects uses a fixed payload size. ChunkSize MUST only appear in Chunk 0, if it is used. The first chunk of user data (a chunk with a Payload TLV) may not be chunk 0.

Any mismatch between the value of the length field of the payload TLV and the the ChunkSize TLV is considered an error of chunk creation by the producer. Such mismatched Data messages returned from a matching Interest MUST be discarded by the consumer.

Because of the possible use of a fixed block size, the inclusion of certain cryptographic fields in the same Content Objects as user data would throw off the ability to seek. Therefore, it is RECOMMENDED that all required cryptographic data, such as public keys or key name links, be included in the leading chunks before the first byte of user data. User data SHOULD then run continuously. If the publisher included ChunkSize in Chunk 0, then all Payload sizes must be of ChunkSize except the last, which may be smaller.

Data produced using the chunking protocol may also be represented via a Manifest (e.g., FLIC [I-D.irtf-icnrg-flic]). An advantage of using a Manifest is that all cryptographic data is in the manifest, which then links to each chunk via a full hash name. This means that the chunked user data can devote all available Content Object bytes after the name to the user data.

To summarize:

1. Every chunk MUST have a ChunkNumber name segment, beginning at 0 and incrementing by 1.

2. The leading chunks MAY have missing or empty Payload TLVs and convey only cryptographic or other information.

3. The last chunk MUST have an EndChunkNumber TLV and the value MUST be equal to the ChunkNumber TLV value.

4. Content Objects before the last chunk MAY have an EndChunkNumber TLV with the expected last chunk number. These hints MAY be updated in subsequent Content Objects but SHOULD NOT decrease.

5. If the last chunk has a ChunkNumber less than a previously published EndChunkNumber, the publisher SHOULD pad out the chunks with empty Content Objects that have the real last chunk number in the EndChunkNumber.

6. An optional ChunkSize TLV MAY be in, and only in, Chunk 0. If it is present, it indicates that all Payloads are the same number of bytes, except the last chunk which may be smaller.

3.2. Cryptographic Material

Chunk 0 MAY include the public key or key name link used to verify the chunked data. It is RECOMMENDED to use the same key for the whole set of chunked data. If a publisher uses multiple keys, then the public key or key name link for all keys SHOULD be in the leading chunks before any user data. Each subsequent chunk only needs to include the KeyId and signature.

As noted above, using a Manifest eliminates the need for cryptographic material in the user-data Content Objects.

3.3. Examples

Here are some examples of chunked Names using the Labeled Content Identifier URI scheme in human readable form (ccnx:).

In this example, the content producer publishes a JPG that takes 4 Chunks. The EndChunkNumber is missing in the first content object (Chunk 0), but is known and included when Chunk 1 is published. It is omitted in Chunk 2, then appears in Chunk 3, where it is mandatory.

```
ccnx:/Name=example.com/Name=picture.jpg/Chunk=0  --
ccnx:/Name=example.com/Name=picture.jpg/Chunk=1  EndChunkNumber=3
ccnx:/Name=example.com/Name=picture.jpg/Chunk=2  --
ccnx:/Name=example.com/Name=picture.jpg/Chunk=3  EndChunkNumber=3
```

In this example, the publisher is publishing an audio stream that ends before expected so the publisher fills empty Content Objects out to the maximum ChunkNumber, stating the correct EndChunkNumber. Chunks 4, 5, and 6 do not contain any new user data.

```
ccnx:/Name=example.com/Name=talk.wav/Chunk=0  --
ccnx:/Name=example.com/Name=talk.wav/Chunk=1  EndChunkNumber=6
ccnx:/Name=example.com/Name=talk.wav/Chunk=2  --
ccnx:/Name=example.com/Name=talk.wav/Chunk=3  EndChunkNumber=3
ccnx:/Name=example.com/Name=talk.wav/Chunk=4  EndChunkNumber=3
ccnx:/Name=example.com/Name=talk.wav/Chunk=5  EndChunkNumber=3
ccnx:/Name=example.com/Name=talk.wav/Chunk=6  EndChunkNumber=3
```

4. TLV Types

This section specifies the TLV types used by CCNx chunking. TBA by IANA (see Section 6).

4.1. ChunkNumber TLV

CCNx chunking defines one new CCNx Name Segment type in the IANA Registry: ChunkNumber.

Type	Abbrev	Name	Description
TBA by IANA	T_CHUNK	Chunk Number (Section 4.1)	The Chunk number of the associated Content Object. It is an unsigned integer in network byte order without leading zeros. The value of zero is represented as the single byte %x00.

Table 1: ChunkNumber

In ccnx: URI form, it is denoted as "Chunk", like the example shown in Section 3.3.

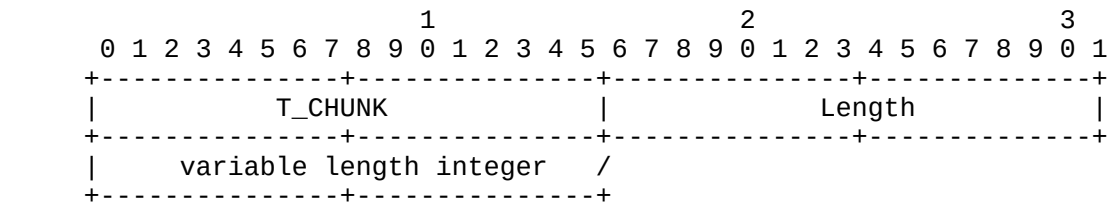

Figure 1: ChunkNumber TLV

4.2. EndChunkNumber and ChunkSize TLVs

CCNx chunking defines two new CCNx Message types in the IANA Registry: EndChunkNumber and ChunkSize.

Type	Abbrev	Name	Description
TBA by IANA	T_ENDCHUNK	EndChunkNumber (Section 4.2)	The value in the EndChunkNumber TLV indicates the last chunk number. It is an unsigned integer in network byte order without leading zeros. The value of zero is represented as the single byte %x00.
TBA by IANA	T_CHUNK_SIZE	ChunkSize (Section 4.2)	The fixed payload size of user data up to the last chunk. It MUST be a positive integer.

Table 2: EndChunkNumber and ChunkSize

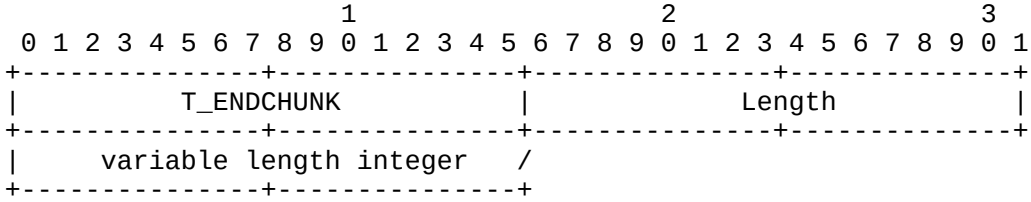

Figure 2: EndChunkNumber TLV

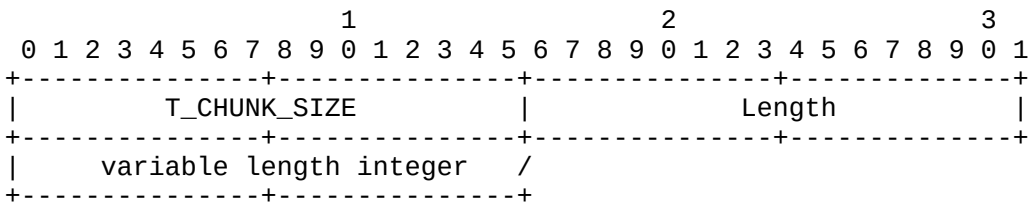

Figure 3: ChunkSize TLV

5. Acknowledgements

The authors would like to thank Ken Calvert and David Oran for their valuable comments and suggestions on this document.

6. IANA Considerations

As per [RFC8126], this section makes an assignment in one existing registry in the "Content-Centric Networking (CCNx)" registry group. The registration procedure is "RFC Required".

6.1. CCNx Name Segment Type Registry

This document defines one message type, T_CHUNK, whose suggested value is %x0005.

6.2. CCNx Message Type Registry

This document defines two message types, T_ENDCHUNK and T_CHUNK_SIZE, whose suggested values are %x0008 and %0x0009, respectively.

7. Security Considerations

The addition of chunking for large application objects does not affect the existing methods used in CCNx for authenticating user data as it employs CCNx Content Objects identically to how they are secured for non chunked data. Therefore, users of this chunking protocol may employ any of the existing signing and encrypting methods without additional considerations.

8. References

8.1. Normative References

[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/info/rfc2119>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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/info/rfc8174>.

[RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <https://www.rfc-editor.org/info/rfc8569>.

[RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <https://www.rfc-editor.org/info/rfc8609>.

8.2. Informative References

[I-D.irtf-icnrg-flic] Tschudin, C., Wood, C. A., Mosko, M., and D. R. Oran, "File-Like ICN Collections (FLIC)", Work in Progress, Internet-Draft, draft-irtf-icnrg-flic-07, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-flic-07>.

Authors' Addresses

Marc Mosko
Kensington, California 94707
United States of America
Email: marc@mosko.org

Hitoshi Asaeda
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi, Tokyo
184-8795
Japan
Email: asaeda@nict.go.jp