Network Working Group
Internet Draft
Intended status: Standards Track
Expires: May 2013

V. Roca
INRIA
B. Adamson
Naval Research Laboratory
November 30, 2012

FCAST: Scalable Object Delivery for the ALC and NORM Protocols

Status of this Memo

CONFORMANCE UNDEFINED.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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”.

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire in May 2013.

Abstract

This document introduces the FCAST object (e.g., file) delivery application on top of the ALC and NORM reliable multicast protocols. FCAST is a highly scalable application that provides a reliable object delivery service.

1. Introduction

This document introduces the FCAST reliable and scalable object (e.g., file) delivery application. Two variants of FCAST exist:

- FCAST/ALC that relies on the Asynchronous Layer Coding (ALC) [RFC5775] and the Layered Coding Transport (LCT) [RFC5651] reliable multicast transport protocol, and
- FCAST/NORM that relies on the NACK-Oriented Reliable Multicast (NORM) [RFC5740] reliable multicast transport protocol.

Hereafter, the term FCAST denotes either FCAST/ALC or FCAST/NORM. FCAST is not a new protocol specification per se. Instead it is a set of data format specifications and instructions on how to use ALC and NORM to implement a file-casting service.

A design goal behind FCAST is to define a streamlined solution, in order to enable lightweight implementations of the protocol stack, and limit the operational processing and storage requirements. A consequence of this choice is that FCAST cannot be considered as a versatile application, capable of addressing all the possible use-cases. On the contrary, FCAST has some intrinsic limitations. From this point of view it differs from FLUTE [RFC6726] which favors flexibility at the expense of some additional complexity.

A good example of the design choices meant to favor simplicity is the way FCAST manages the object meta-data: by default, the meta-data and the object content are sent together, in a compound object. This solution has many advantages in terms of simplicity as will be described later on. However this solution has an intrinsic limitation since it does not enable a receiver to decide in advance, before beginning the reception of the compound object, whether the object is of interest or not, based on the information that may be provided in the meta-data. Therefore this document discusses additional techniques that may be used to mitigate this limitation. When use-cases require that each receiver download the whole set of objects sent in the session (e.g., with mirroring tools), this limitation is not considered a problem.

FCAST is expected to work in many different environments and is designed to be flexible. The service provided by FCAST can differ according to the exact conditions FCAST is used, like the presence or not of reception feedbacks. For example, environments exist where no feedback is possible with FCAST/ALC, which guarantees a maximum scalability but at the same time may reduce transmission reliability. Section 6 discusses such operational considerations in detail. Finally, Section 5 provides the guidance for compliant implementation of the specification and identifies those features that are optional.

1.1 Requirements Notation

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 [RFC2119].

1.2 Definitions, Notations and Abbreviations

This document uses the following terms:

FCAST/ALC	: Denotes the FCAST application running on top of the ALC/LCT reliable transport protocol
FCAST/NORM	: Denotes the FCAST application running on top of the NORM reliable transport protocol
FCAST	: Denotes either FCAST/ALC or FCAST/NORM
Compound Object	: An ALC or NORM transport object composed of the Compound Object Header
Carousel	: The process of sending Compound Objects implemented by a FCAST sender
Carousel Instance	: Fixed set of registered Compound Objects that are sent by the carousel during a certain number of cycles; Whenever Compound Objects need to be added or removed, a new Carousel Instance is defined.
Carousel Instance Descriptor (CID)	: Special object that lists the Compound Objects comprising a given Carousel Instance

Carousel Instance IDentifier (CIID)	: Numeric value that identifies a Carousel Instance
Carousel Cycle	: A transmission round within which all the Compound Objects registered in the Carousel Instance are transmitted a certain number of times; By default, Compound Objects are transmitted once per cycle, but higher values are possible, that might differ on a per-object basis.
Transport Object Identifier (TOI)	: Numeric identifier associated to a specific object by the underlying transport protocol. In the case of ALC, this corresponds to the TOI described in [RFC5651]; In the case of NORM, this corresponds to the NormTransportId described in [RFC5740].
FEC Object Transmission Information (FEC OTI)	: FEC information associated with an object that is essential for the FEC decoder to decode a specific object.

2. FCAST Data Formats

This section details the various data formats used by FCAST.

2.1 Compound Object Format

In an FCAST session, Compound Objects are constructed by prepending the Compound Object Header (which contains the meta-data of the object) before the original object data content (see Section 3.2). Figure 1 illustrates the associated Compound Object header format. All multi-byte fields are in network (Big Endian) byte order.

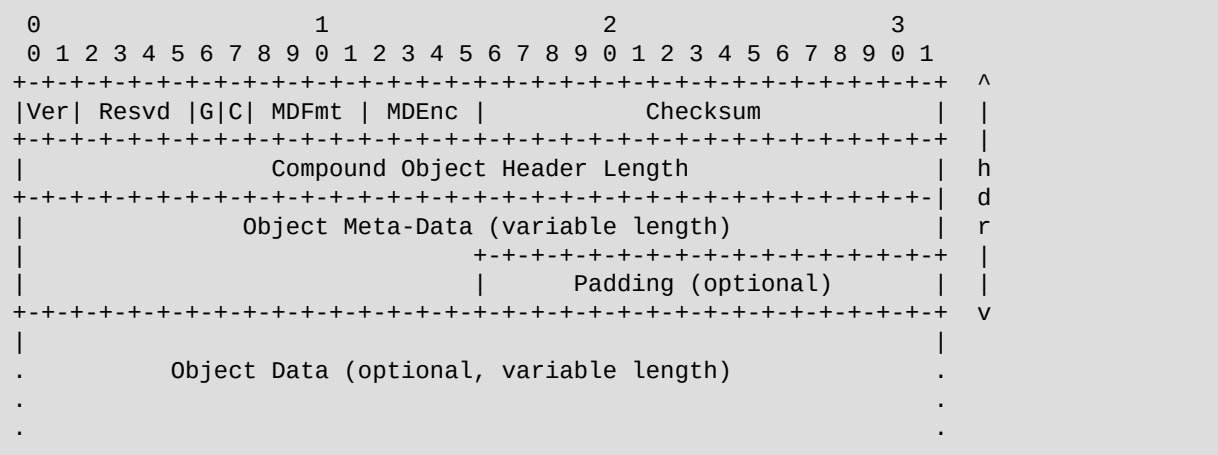

Figure 1: Compound Object Format.

The Compound Object Header fields are:

Field	Description
Version	2-bit field that MUST be set to 0 in this specification and indicates the protocol version number.
Reserved	4-bit field that MUST be set to 0 in this specification and is reserved for future use. Receivers MUST ignore this field.
G	1-bit field that, when set to 1, indicates that the checksum encompasses the whole Compound Object (Global checksum). When set to 0, this field indicates that the checksum encompasses only the Compound Object header.
C	1-bit field that, when set to 1, indicates the object is a Carousel Instance Descriptor (CID). When set to 0, this field indicates that the transported object is a standard object.
Meta-Data Format (MDFmt)	4-bit field that defines the format of the object meta-data (see Section 7). An HTTP/1.1 metainformation format [RFC2616] MUST be supported and is associated to value 0. Other formats (e.g., XML) MAY be defined in the future.
Meta-Data Encoding (MDEnc)	4-bit field that defines the optional encoding of the Object Meta-Data field (see Section 7). By default, a plain text encoding is used and is associated to value 0. GZIP encoding MUST also be supported and is associated to value 1. Other encodings MAY be defined in the future.
Checksum	16-bit field that contains the checksum computed over either the whole Compound Object (when G is set to 1), or over the Compound Object header (when G is set to 0), using the Internet checksum algorithm specified in [RFC1071]. More precisely, the checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words to be considered. If a segment contains an odd number of octets to be checksummed, the last octet is padded on the right with zeros to form a 16-bit word for checksum purposes (this pad is not transmitted). While computing the checksum, the checksum field itself MUST be set to zero.
Compound Object Header Length	32-bit field indicating total length (in bytes) of all fields of the Compound Object Header, except the optional padding. A header length field set to value 8 means that there is no meta-data included. When this size is not multiple to 32-bits words and when the Compound Object

Field	Description
	Header is followed by a non null Compound Object Data, padding MUST be added. It should be noted that the meta-data field maximum size is equal to (2^32 - 8) bytes.
Object Meta-Data	Variable length field that contains the meta-data associated to the object. The format and encoding of this field are defined respectively by the MDFmt and MDEnc fields. With the default HTTP/1.1 format and plain text encoding, the Meta-Data is NULL-terminated plain text that follows the "TYPE" ":" "VALUE" "<CR-LF>" format used in HTTP/1.1 for metainformation [RFC2616]. The various meta-data items can appear in any order. The associated string, when non empty, MUST be NULL-terminated. When no meta-data is communicated, this field MUST be empty and the Compound Object Header Length MUST be equal to 8.
Padding	Optional, variable length field of zero-value bytes to align the start of the Object Data to 32-bit boundary. Padding is only used when the Compound Object Header Length value, in bytes, is not multiple of 4 and when the Compound Object Header is followed by non null Compound Object Data.

The Compound Object Header is then followed by the Object Data, i.e., the original object possibly encoded by FCAST. Note that the length of this content is the transported object length (e.g., as specified by the FEC OTI) minus the Compound Object Header Length and optional padding if any.

2.2 Carousel Instance Descriptor Format

In an FCAST session, a Carousel Instance Descriptor (CID) MAY be sent in order to carry the list of Compound Objects that are part of a given Carousel Instance (see Section 3.5). The format of the CID, that is sent as a special Compound Object, is given in Figure 2. Being a special case of Compound Object, this format is in line with the format described in Section 2.1.

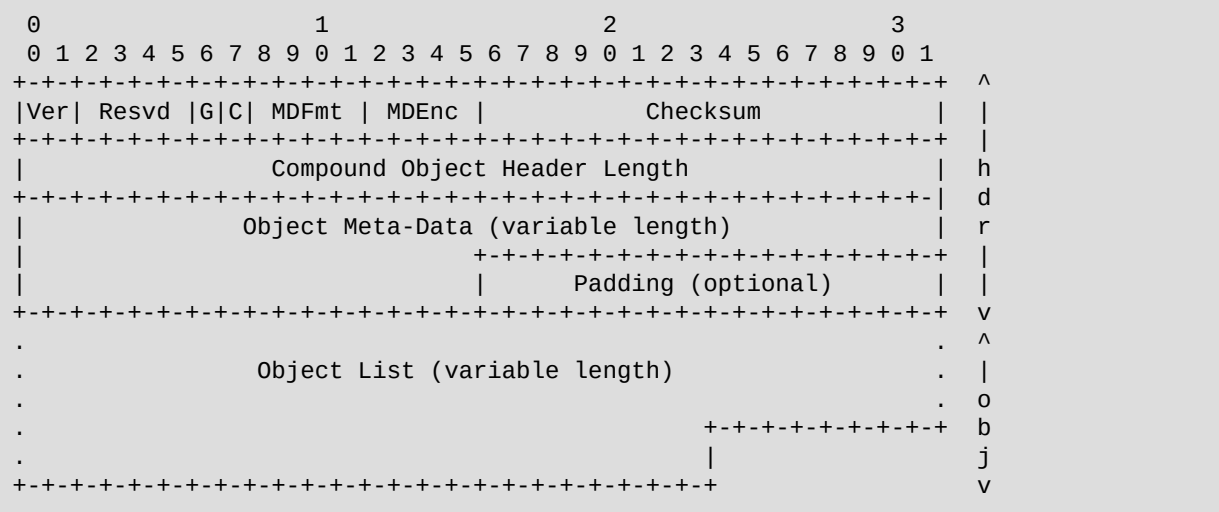

Figure 2: Carousel Instance Descriptor Format.

Because the CID is transmitted as a special Compound Object, the following CID-specific meta-data entries are defined:

- Fcast-CID-Complete: when set to '1', it indicates that no new object in addition to the ones whose TOI are specified in this CID, or the ones that have been specified in the previous CID(s), will be sent in the future. Otherwise it MUST be set to 0. This entry is optional. If absent, a receiver MUST conclude that the session is complete.
- Fcast-CID-ID: this entry contains the Carousel Instance IDentifier, or CIID. It starts from 0 and is incremented by 1 for each new carousel instance. This entry is optional if the FCAST session consists of a single, complete, carousel instance. In all other cases, this entry MUST be defined. In particular, the CIID is used by the TOI

equivalence mechanism thanks to which any object is uniquely identified, even if the TOI is updated (e.g., after re-enqueuing the object with NORM). The Fcast-CID-ID value can also be useful to detect possible gaps in the Carousel Instances, for instance caused by long disconnection periods. Finally, it can also be useful to avoid problems when TOI wrapping to 0 takes place to differentiate the various incarnations of the TOIs if need be.

The motivation for making the Fcast-CID-Complete and Fcast-CID-ID entries optional is to simplify the simple case of a session consisting of a single, complete, carousel instance, with an Object List given in plain text, without any content encoding. In that case, the CID does not need to contain any meta-data entry.

The following standard meta-data entry types are also used (Section 3.3):

- Content-Length: it specifies the size of the object list, before any content encoding (if any).
- Content-Encoding: it specifies the optional encoding of the object list, performed by FCAST. For instance:

```
Content-Encoding: gzip
```

indicates that the Object List field has been encoded with GZIP [RFC1952]. If there is no Content-Encoding entry, the receiver MUST assume that the Object List field is plain text (default). The support of GZIP encoding, in addition to the plain text form, is REQUIRED. The Content-Encoding entry MAY be used to indicate other encoding types to support non-standard FCAST implementation or future extended specifications.

An empty Object List is valid and indicates that the current carousel instance does not include any objects (Section 3.5). This can be specified by using the following meta-data entry:

```
Content-Length: 0
```

or simply by leaving the Object List empty. In both cases, padding MUST NOT be used and consequently the transported object length (e.g., as specified by the FEC OTI) minus the Compound Object Header Length equals zero.

The non-encoded (i.e., plain text) Object List, when non empty, is a NULL-terminated ASCII string. It can contain two things:

- a list of TOI values, and
- a list of TOI equivalences;

A list of TOIs included in the current carousel instance is specified as an ASCII string containing comma-delimited individual TOI values and/or TOI intervals. Inidividual TOIs consist of a single integer value while TOI intervals are a hyphen-delimited pair of TOI values to indicate a inclusive range of TOI values (e.g., "1,2,4-6" would indicate the list of TOI values of 1,2,4,5, and 6). For a TOI Interval indicated by ""TOI_a-TOI_b", the 'TOI_a' value MUST be strictly inferior to the 'TOI_b' value. If a TOI wrapping to 0 occurs in an interval, then two TOI intervals MUST be specified, TOI_a-MAX_TOI and 0-TOI_b.

This string can also contain the TOI equivalences, if any. The format is a comma-separated list of equivalence TOI value pairs with a delimiting equals size '=' to indicate the equivalence assignment (e.g., " newTOI "=" 1stTOI "/" 1stCIID "). Each equivalence indicates that the new TOI, for the current Carousel Instance, is equivalent to (i.e., refers to the same object as) the provided identifier, 1stTOI, for the Carousel Instance of ID 1stCIID. In the case of the NORM protocol where NormTransportId values need to monotonically increase for NACK-based protocol operation, this allows an object from a prior Carousel Instance to be reincluded in a subsequent Carousel Instance with the receiver set informed of the equivalence so that unnecessary retransmission requests can be avoided.

The ABNF [RFC5234] specification is the following:

```
cid-list   =  *(list-elem *( "," list-elem))
list-elem    =  toi-elem / toieq-elem
toi-elem     =  toi-value / toi-interval
toi-value    =  1*DIGIT
toi-interval =  toi-value "-" toi-value
                ; additionally, the first toi-value MUST be
                ; strictly inferior to the second toi-value
toieq-elem   =  "(" toi-value "=" toi-value "/" ciid-value ")"
ciid-value   = 1*DIGIT
DIGIT        =  %x30-39
                ; a digit between 0 and 9, inclusive
```

For readability purposes and to simplify processing, the TOI values in the list MUST be given in increasing order handling wrap of the TOI space appropriately. TOI equivalence elements MUST be grouped together at the end of the list in increasing newTOI order. Specifying a TOI equivalence for a given newTOI relieves the sender from specifying newTOI explicitly in the TOI list. A receiver MUST be able to handle situations where the same TOI appears both in the TOI value and TOI equivalence lists. Finally, a given TOI value or TOI equivalence item MUST NOT be included multiple times in either list.

For instance, the following object list specifies that the current Carousel Instance is composed of 8 objects, and that TOIs 100 to 104 are equivalent to the TOIs 10 to 14 of Carousel Instance ID 2 and refer to the same objects:

```
97,98,99,(100=10/2),(101=11/2),(102=12/2),(103=13/2),(104=14/2)
```

or equivalently:

```
97-104,(100=10/2),(101=11/2),(102=12/2),(103=13/2),(104=14/2)
```

3. FCAST Principles

This section details the principles of FCAST.

3.1 FCAST Content Delivery Service

The basic goal of FCAST is to transmit objects to a group of receivers in a reliable way, where the receiver set may be restricted to a single receiver or may include possibly a very large number of receivers. FCAST supports two forms of operation:

1. FCAST/ALC, where the FCAST application works on top of the ALC/LCT reliable multicast transport protocol, without any feedback from the receivers, and
2. FCAST/NORM, where the FCAST application works on top of the NORM reliable multicast transport protocol, that requires positive/negative acknowledgements from the receivers.

This specification is designed such that both forms of operation share as much commonality as possible. Section 6 discusses some operational aspects and the content delivery service that is provided by FCAST for a given use-case.

3.2 Meta-Data Transmission

FCAST carries meta-data elements by prepending them to the object they refer to. As a result, a Compound Object is created that is composed of a header followed by the original object data. This header is itself composed of the meta-data as well as several fields, for instance to indicate the boundaries between the various parts of this Compound Object (Figure 3).

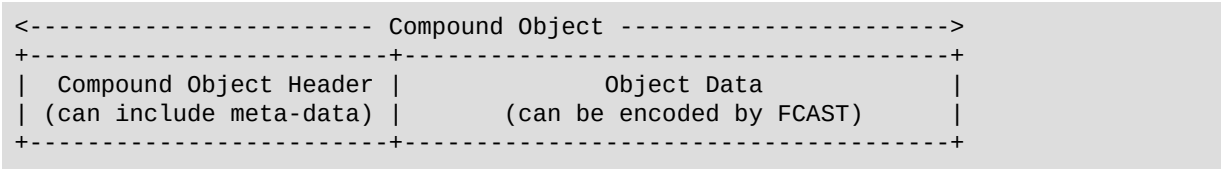

Figure 3: Compound Object composition.

Attaching the meta-data to the object is an efficient solution, since it guaranties that meta-data are received along with the associated object, and it allows the transport of the meta-data to benefit from any transport-layer erasure protection of the Compound Object (e.g., using FEC encoding and/or NACK-based repair). However a limit of this scheme is that a client does not know the meta-data of an object before beginning its reception, and in case of erasures affecting the meta-data, not until the object decoding is completed. The details of course depend upon the transport protocol and the FEC code used.

Appendix B describes extensions that provide additional means to carry meta-data, e.g., to communicate meta-data ahead of time.

3.3 Meta-Data Content

A compliant FCAST implementation MUST support at least the following meta-data types defined in [RFC2616]:

- Content-Location: the URI of the object, which gives the name and location of the object;
- Content-Type: a string that contains the MIME type of the object;
- Content-Length: an unsigned 64-bit integer that contains the size of the initial object, before any content encoding (if any) and without considering the Compound Object header. Note that the use of certain FEC schemes MAY further limit the maximum value of the object;
- Content-Encoding: a string that contains the optional encoding of the object performed by FCAST. If there is no Content-Encoding entry, the receiver MUST assume that the object is not encoded (default). The support of GZIP encoding, or any other solution, remains optional.
- Content-MD5: a binary content coded as "base64" that contains the MD5 message digest of the object in order to check its integrity. This digest is meant to protect from transmission and processing errors, not from deliberate

attacks by an intelligent attacker (see Section 4). This digest only protects the object, not the header, and therefore not the meta-data. A separate checksum is provided to that purpose (Section 2.1);

The following additional REQUIRED meta-data types are used for dealing with very large objects (e.g., that largely exceed the working memory of a receiver). When this happens, the meta-data associated to each sub-object MUST include the following entries:

- Fcast-Obj-Slice-Nb: an unsigned 32-bit integer that contains the total number of slices. A value greater than 1 indicates that this object is the result of a split of the original object;
- Fcast-Obj-Slice-Idx: an unsigned 32-bit integer that contains the slice index (in the {0 .. SliceNb - 1} interval);
- Fcast-Obj-Slice-Offset: an unsigned 64-bit integer that contains the offset at which this slice starts within the original object;

Future standards actions can extend the set of meta-data types supported by FCAST.

3.4 Carousel Transmission

A set of FCAST Compound Objects scheduled for transmission are considered a logical "Carousel". A given "Carousel Instance" is comprised of a fixed set of Compound Objects. Whenever the FCAST application needs to add new Compound Objects to or remove old Compound Objects from the transmission set, a new Carousel Instance is defined since the set of Compound Objects changes. Because of the native object multiplexing capability of both ALC and NORM, sender and receiver(s) are both capable to multiplex and demultiplex FCAST Compound Objects.

For a given Carousel Instance, one or more transmission cycles are possible. During each cycle, all of the Compound Objects comprising the Carousel are sent. By default, each object is transmitted once per cycle. However, in order to allow different levels of priority, some objects MAY be transmitted more often that others during a cycle, and/or benefit from higher FEC protection than others. For example, this can be the case for the CID objects (Section 3.5) where extra protection can benefit overall carousel integrity. For some FCAST usage (e.g., a unidirectional "push" mode), a Carousel Instance may be sent in a single transmission cycle. In other cases it may be conveyed in a large number of transmission cycles (e.g., in "on-demand" mode, where objects are made available for download during a long period of time).

3.5 Carousel Instance Descriptor Special Object

The FCAST sender can transmit an OPTIONAL Carousel Instance Descriptor (CID). The CID carries the list of the Compound Objects that are part of a given Carousel Instance, by specifying their respective Transmission Object Identifiers (TOI). However the CID does not describe the objects themselves (i.e., there is no meta-data). Additionally, the CID MAY include an "FCAST-CID-Complete" meta-data entry set to '1' to indicate that no further modification to the enclosed list will be done in the future. Finally, the CID MAY include a Carousel Instance ID (CIID) that identifies the Carousel Instance it pertains to. These aspects are discussed in Section 2.2.

There is no reserved TOI value for the CID Compound Object itself, since this special object is regarded by ALC/LCT or NORM as a standard object. On the contrary, the nature of this object (CID) is indicated by means of a specific Compound Object header field (the "C" flag from Figure 1) so that it can be recognized and processed by the FCAST application as needed. A direct consequence is the following: since a receiver does not know in advance which TOI will be used for the following CID (in case of a dynamic session), it MUST NOT filter out packets that are not in the current CID's TOI list. Said differently, the goal of CID is not to setup ALC or NORM packet filters (this mechanism would not be secure in any case).

The use of a CID remains OPTIONAL. If it is not used, then the clients progressively learn what files are part of the carousel instance by receiving ALC or NORM packets with new TOIs. However using a CID has several benefits:

- When an "FCAST-CID-Complete" meta-data entry set to '1' is included, the receivers know when they can leave the session, i.e., when they have received all the objects that are part of the last carousel instance of this delivery session;

- In case of a session with a dynamic set of objects, the sender can reliably inform the receivers that some objects have been removed from the carousel with the CID. This solution is more robust than the "Close Object flag (B)" of ALC/LCT since a client with an intermittent connectivity might loose all the packets containing this B flag. And while NORM provides a robust object cancellation mechanism in the form of its NORM_CMD(SQUELCH) message in response to receiver NACK repair requests, the use of the CID provides an additional means for receivers to learn of objects for which it is futile to request repair;
- The TOI equivalence (Section 3.6) can be signaled with the CID. This is often preferable to the alternative solution where the equivalence is identified by examining the object meta-data, especially in case of erasures.

During idle periods, when the carousel instance does not contain any object, a CID with an empty TOI list MAY be transmitted. In that case, a new carousel instance ID MUST be used to differentiate this (empty) carousel instance from the other ones. This mechanism can be useful to inform the receivers that:

- all the previously sent objects have been removed from the carousel. It therefore improves the FCAST robustness even during "idle" period;
- the session is still active even if there is currently no content being sent. Said differently, it can be used as a heartbeat mechanism. If an "FCAST-CID-Complete" meta-data entry is not included (or if set to '0'), it informs the receivers the carousel instance may be modified and that new objects could be sent in the future;

3.6 Compound Object Identification

The FCAST Compound Objects are directly associated with the object-based transport service that the ALC and NORM protocols provide. In each of these protocols, the messages containing transport object content are labeled with a numeric transport object identifier (i.e., the ALC TOI and the NORM NormTransportId). For the purposes of this document, this identifier in either case (ALC or NORM) is referred to as the TOI.

There are several differences between ALC and NORM:

- the ALC use of TOI is rather flexible, since several TOI field sizes are possible (from 16 to 112 bits), since this size can be changed at any time, on a per-packet basis, and since the TOI management is totally free as long as each object is associated to a unique TOI (if no wraparound happened);
- the NORM use of TOI is more directive, since the TOI field is 16 bit long and since TOIs MUST be managed sequentially;

In both NORM and ALC, it is possible that the transport identification space may eventually wrap for long-lived sessions (especially with NORM where this phenomenon is expected to happen more frequently). This can possibly introduce some ambiguity in FCAST object identification if a sender retains some older objects in newer Carousel Instances with updated object sets. To avoid ambiguity the active TOIs (i.e., the TOIs corresponding to objects being transmitted) can only occupy half of the TOI sequence space. If an old object, whose TOI has fallen outside the current window, needs to be transmitted again, a new TOI must be used for it. In case of NORM, this constraint limits to 32768 the maximum number of objects that can be part of any carousel instance. In order to allow receivers to properly combine the transport packets with a newly-assigned TOI to those of associated to the previously-assigned TOI, a mechanism is required to equate the objects with the new and the old TOIs.

The preferred mechanism consists in signaling, within the CID, that the newly assigned TOI, for the current Carousel Instance, is equivalent to the TOI used within a previous Carousel Instance. By convention, the reference tuple for any object is the {TOI; CI ID} tuple used for its first transmission within a Carousel Instance. This tuple MUST be used whenever a TOI equivalence is provided.

An alternative solution, when meta-data can be processed rapidly (e.g., by using NORM-INFO messages), consists for the receiver in identifying that both objects are the same, after examining the meta-data. The receiver can then take appropriate measures.

3.7 FCAST Sender Behavior

This section provides an informative description of expected FCAST sender behavior. The following operations can take place at a sender:

1. The user (or another application) selects a set of objects (e.g., files) to deliver and submits them, along with their meta-data, to the FCAST application;
2. For each object, FCAST creates the Compound Object and registers this latter in the carousel instance;
3. The user then informs FCAST that all the objects of the set have been submitted. If the user knows that no new object will be submitted in the future (i.e., if the session's content is now complete), the user informs FCAST. Finally, the user specifies how many transmission cycles are desired (this number may be infinite);
4. At this point, the FCAST application knows the full list of Compound Objects that are part of the Carousel Instance and can create a CID if desired, possibly with the complete flag set;
5. The FCAST application can now define a transmission schedule of these Compound Objects, including the optional CID. This schedule defines in which order the packets of the various Compound Objects should be sent. This document does not specify any scheme. This is left to the developer within the provisions of the underlying ALC or NORM protocol used and the knowledge of the target use-case.
6. The FCAST application then starts the carousel transmission, for the number of cycles specified. Transmissions take place until:
   - the desired number of transmission cycles has been reached, or
   - the user wants to prematurely stop the transmissions, or
   - the user wants to add one or several new objects to the carousel, or on the contrary wants to remove old objects from the carousel. In that case a new carousel instance must be created.
7. If the session is not finished, then continue at Step 1 above;

3.8 FCAST Receiver Behavior

This section provides an informative description of expected FCAST receiver behavior. The following operations can take place at a receiver:

1. The receiver joins the session and collects incoming packets;
2. If the header portion of a Compound Object is entirely received (which may happen before receiving the entire object with some ALC/NORM configurations), or if the meta-data is sent by means of another mechanism prior to the object, the receiver processes the meta-data and chooses to continue to receive the object content or not;
3. When a Compound Object has been entirely received, the receiver processes the header, retrieves the object meta-data, perhaps decodes the meta-data, and processes the object accordingly;
4. When a CID is received, which is indicated by the 'C' flag set in the Compound Object header, the receiver decodes the CID, and retrieves the list of objects that are part of the current carousel instance. This list can be used to remove objects sent in a previous carousel instance that might not have been totally decoded and that are no longer part of the current carousel instance;
5. When a CID is received, the receiver also retrieves the list of TOI equivalences, if any, and takes appropriate measures, for instance by informing the transport layer;
6. When a receiver receives a CID with an "FCAST-CID-Complete" meta-data entry set to '1', and has successfully received all the objects of the current carousel instance, it can safely exit from the current FCAST session;
7. Otherwise continue at Step 2 above.

4. Security Considerations

4.1 Problem Statement

A content delivery system is potentially subject to attacks. Attacks may target:

- the network (to compromise the routing infrastructure, e.g., by creating congestion),
- the Content Delivery Protocol (CDP) (e.g., to compromise the normal behavior of FCAST), or
- the content itself (e.g., to corrupt the objects being transmitted).

These attacks can be launched either:

- against the data flow itself (e.g., by sending forged packets),
- against the session control parameters (e.g., by corrupting the session description, the CID, the object meta-data, or the ALC/LCT control parameters), that are sent either in-band or out-of-band, or
- against some associated building blocks (e.g., the congestion control component).

In the following sections we provide more details on these possible attacks and sketch some possible counter-measures. We finally provide recommendations in Section 4.5.

4.2 Attacks Against the Data Flow

Let us consider attacks against the data flow first. At least, the following types of attacks exist:

- attacks that are meant to give access to a confidential object (e.g., in case of a non-free content) and
- attacks that try to corrupt the object being transmitted (e.g., to inject malicious code within an object, or to prevent a receiver from using an object, which is a kind of Denial of Service (DoS)).

4.2.1 Access to Confidential Objects

Access control to the object being transmitted is typically provided by means of encryption. This encryption can be done over the whole object (e.g., by the content provider, before submitting the object to FCAST), or be done on a packet per packet basis (e.g., when IPsec/ESP is used [RFC4303], see Section 4.5). If confidentiality is a concern, it is RECOMMENDED that one of these solutions be used.

4.2.2 Object Corruption

Protection against corruptions (e.g., if an attacker sends forged packets) is achieved by means of a content integrity verification/sender authentication scheme. This service can be provided at the object level, but in that case a receiver has no way to identify which symbol(s) is(are) corrupted if the object is detected as corrupted. This service can also be provided at the packet level. In this case, after removing all corrupted packets, the file may be in some cases recovered. Several techniques can provide this content integrity/sender authentication service:

- at the object level, the object can be digitally signed, for instance by using RSASSA-PKCS1-v1_5 [RFC3447]. This signature enables a receiver to check the object integrity, once this latter has been fully decoded. Even if digital signatures are computationally expensive, this calculation occurs only once per object, which is usually acceptable;
- at the packet level, each packet can be digitally signed [RFC6584]. A major limitation is the high computational and transmission overheads that this solution requires. To avoid this problem, the signature may span a set of packets (instead of a single one) in order to amortize the signature calculation. But if a single packets is missing, the integrity of the whole set cannot be checked;
- at the packet level, a Group Message Authentication Code (MAC) [RFC2104][RFC6584] scheme can be used, for instance by using HMAC-SHA-256 with a secret key shared by all the group members, senders and receivers. This technique creates a cryptographically secured digest of a packet that is sent along with the packet. The Group MAC scheme does not create prohibitive processing load nor transmission overhead, but it has a major limitation: it only provides a group authentication/integrity service since all group members share the same secret group key, which means that each member can send a forged packet. It is therefore

restricted to situations where group members are fully trusted (or in association with another technique as a pre-check);

- at the packet level, Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [RFC4082][RFC5776] is an attractive solution that is robust to losses, provides a true authentication/integrity service, and does not create any prohibitive processing load or transmission overhead. Yet checking a packet requires a small delay (a second or more) after its reception;
- at the packet level, IPsec/ESP [RFC4303] can be used to check the integrity and authenticate the sender of all the packets being exchanged in a session (see Section 4.5).

Techniques relying on public key cryptography (digital signatures and TESLA during the bootstrap process, when used) require that public keys be securely associated to the entities. This can be achieved by a Public Key Infrastructure (PKI), or by a PGP Web of Trust, or by pre-distributing securely the public keys of each group member.

Techniques relying on symmetric key cryptography (Group MAC) require that a secret key be shared by all group members. This can be achieved by means of a group key management protocol, or simply by pre-distributing securely the secret key (but this manual solution has many limitations).

It is up to the developer and deployer, who know the security requirements and features of the target application area, to define which solution is the most appropriate. In any case, whenever there is any concern of the threat of file corruption, it is RECOMMENDED that at least one of these techniques be used.

4.3 Attacks Against the Session Control Parameters and Associated Building Blocks

Let us now consider attacks against the session control parameters and the associated building blocks. The attacker has at least the following opportunities to launch an attack:

- the attack can target the session description,
- the attack can target the FCAST CID,
- the attack can target the meta-data of an object,
- the attack can target the ALC/LCT parameters, carried within the LCT header or
- the attack can target the FCAST associated building blocks, for instance the multiple rate congestion control protocol.

The consequences of these attacks are potentially serious, since they can compromise the behavior of content delivery system or even compromise the network itself.

4.3.1 Attacks Against the Session Description

An FCAST receiver may potentially obtain an incorrect Session Description for the session. The consequence of this is that legitimate receivers with the wrong Session Description are unable to correctly receive the session content, or that receivers inadvertently try to receive at a much higher rate than they are capable of, thereby possibly disrupting other traffic in the network.

To avoid these problems, it is RECOMMENDED that measures be taken to prevent receivers from accepting incorrect Session Descriptions. One such measure is the sender authentication to ensure that receivers only accept legitimate Session Descriptions from authorized senders. How these measures are achieved is outside the scope of this document since this session description is usually carried out-of-band.

4.3.2 Attacks Against the FCAST CID

Corrupting the FCAST CID is one way to create a Denial of Service attack. For example, the attacker can insert an "FCAST-CID-Complete" meta-data entry to make the receivers believe that no further modification will be done.

It is therefore RECOMMENDED that measures be taken to guarantee the integrity and to check the sender's identity of the CID. To that purpose, one of the counter-measures mentioned above (Section 4.2.2) SHOULD be

used. These measures will either be applied on a packet level, or globally over the whole CID object. When there is no packet level integrity verification scheme, it is RECOMMENDED to digitally sign the CID.

4.3.3 Attacks Against the Object Meta-Data

Corrupting the object meta-data is another way to create a Denial of Service attack. For example, the attacker changes the MD5 sum associated to a file. This possibly leads a receiver to reject the files received, no matter whether the files have been correctly received or not. When the meta-data are appended to the object, corrupting the meta-data means that the Compound Object will be corrupted.

It is therefore RECOMMENDED that measures be taken to guarantee the integrity and to check the sender's identity of the Compound Object. To that purpose, one of the counter-measures mentioned above (Section 4.2.2) SHOULD be used. These measures will either be applied on a packet level, or globally over the whole Compound Object. When there is no packet level integrity verification scheme, it is RECOMMENDED to digitally sign the Compound Object.

4.3.4 Attacks Against the ALC/LCT and NORM Parameters

By corrupting the ALC/LCT header (or header extensions) one can execute attacks on the underlying ALC/LCT implementation. For example, sending forged ALC packets with the Close Session flag (A) set to one can lead the receiver to prematurely close the session. Similarly, sending forged ALC packets with the Close Object flag (B) set to one can lead the receiver to prematurely give up the reception of an object. The same comments can be made for NORM.

It is therefore RECOMMENDED that measures be taken to guarantee the integrity and to check the sender's identity of each ALC or NORM packet received. To that purpose, one of the counter-measures mentioned above (Section 4.2.2) SHOULD be used.

4.3.5 Attacks Against the Associated Building Blocks

Let us first focus on the congestion control building block that may be used in an ALC or NORM session. A receiver with an incorrect or corrupted implementation of the multiple rate congestion control building block may affect the health of the network in the path between the sender and the receiver. That may also affect the reception rates of other receivers who joined the session.

When congestion control is applied with FCAST, it is therefore RECOMMENDED that receivers be required to identify themselves as legitimate before they receive the Session Description needed to join the session. If authenticating a receiver does not prevent this latter to launch an attack, it will enable the network operator to identify him and to take counter-measures. This authentication can be made either toward the network operator or the session sender (or a representative of the sender) in case of NORM. The details of how it is done are outside the scope of this document.

When congestion control is applied with FCAST, it is also RECOMMENDED that a packet level authentication scheme be used, as explained in Section 4.2.2. Some of them, like TESLA, only provide a delayed authentication service, whereas congestion control requires a rapid reaction. It is therefore RECOMMENDED [RFC5775] that a receiver using TESLA quickly reduces its subscription level when the receiver believes that a congestion did occur, even if the packet has not yet been authenticated. Therefore TESLA will not prevent DoS attacks where an attacker makes the receiver believe that a congestion occurred. This is an issue for the receiver, but this will not compromise the network. Other authentication methods that do not feature this delayed authentication could be preferred, or a group MAC scheme could be used in parallel to TESLA to prevent attacks launched from outside of the group.

4.4 Other Security Considerations

Lastly, we note that the security considerations that apply to, and are described in, ALC [RFC5775], LCT [RFC5651], NORM [RFC5740] and FEC [RFC5052] also apply to FCAST as FCAST builds on those specifications. In addition, any security considerations that apply to any congestion control building block used in conjunction

with FCAST also applies to FCAST. Finally, the security discussion of [I-D.ietf-rmt-sec-discussion] also applies here.

4.5 Minimum Security Recommendations

We now introduce a mandatory to implement but not necessarily to use security configuration, in the sense of [RFC3365]. Since FCAST/ALC relies on ALC/LCT, it inherits the "baseline secure ALC operation" of [RFC5775]. Similarly, since FCAST/NORM relies on NORM, it inherits the "baseline secure NORM operation" of [RFC5740]. More precisely, in both cases security is achieved by means of IPsec/ESP in transport mode. [RFC4303] explains that ESP can be used to potentially provide confidentiality, data origin authentication, content integrity, anti-replay and (limited) traffic flow confidentiality. [RFC5775] specifies that the data origin authentication, content integrity and anti-replay services SHALL be used, and that the confidentiality service is RECOMMENDED. If a short lived session MAY rely on manual keying, it is also RECOMMENDED that an automated key management scheme be used, especially in case of long lived sessions.

Therefore, the RECOMMENDED solution for FCAST provides per-packet security, with data origin authentication, integrity verification and anti-replay. This is sufficient to prevent most of the in-band attacks listed above. If confidentiality is required, a per-packet encryption SHOULD also be used.

5. Requirements for Compliant Implementations

This section lists the features that any compliant FCAST/ALC or FCAST/NORM implementation MUST support, and those that remain OPTIONAL, e.g., in order to enable some optimizations for a given use-case, at a receiver.

5.1 Requirements Related to the Object Meta-Data

Meta-data transmission mechanisms:

Feature	Status
meta-data transmission using FCAST's in-band mechanism	An FCAST sender MUST send meta-data with the in-band mechanism provided by FCAST, i.e., within the Compound Object header. All the FCAST receivers MUST be able to process meta-data sent with this FCAST in-band mechanism.
meta-data transmission using other mechanisms	In addition to the FCAST in-band transmission of meta-data, an FCAST sender MAY send a subset or all of the meta-data using another mechanism. Supporting this mechanism in a compliant FCAST receiver is OPTIONAL, and its use is OPTIONAL too. An FCAST receiver MAY support this mechanism and take advantage of the meta-data sent in this way. If it is not the case, the FCAST receiver will anyway receive and process meta-data sent in-bound. See Annex Appendix B.

Meta-data format and encoding:

Feature	Status
Meta-Data Format (MDFmt field)	All FCAST implementations MUST support an HTTP/1.1 metainformation format [RFC2616]. Other formats (e.g., XML) MAY be defined in the future.
Meta-Data Encoding (MDEnc field)	All FCAST implementations MUST support both a plain text and a GZIP encoding [RFC1952] of the Object Meta-Data field. Other encodings MAY be defined in the future.

Meta-data items (Section 3.3):

Feature	Status
Content-Location	MUST be supported
Content-Type	MUST be supported
Content-Length	MUST be supported
Content-Encoding	MUST be supported
Content-MD5	MUST be supported
Fcast-Obj-Slice-Nb	MUST be supported
Fcast-Obj-Slice-Idx	MUST be supported
Fcast-Obj-Slice-Offset	MUST be supported

5.2 Requirements Related to the Carousel Instance Descriptor (CID) Mechanism

Any compliant FCAST implementation MUST support the CID mechanism, in order to list the Compound Objects that are part of a given Carousel Instance. However its use is OPTIONAL.

6. Operational Considerations

FCAST is compatible with any congestion control protocol designed for ALC/LCT or NORM. However, depending on the use-case, the data flow generated by the FCAST application might not be constant, but instead be bursty in nature. Similarly, depending on the use-case, an FCAST session might be very short. Whether and how this will impact the congestion control protocol is out of the scope of the present document.

FCAST is compatible with any security mechanism designed for ALC/LCT or NORM. The use of a security scheme is strongly RECOMMENDED (see Section 4).

FCAST is compatible with any FEC scheme designed for ALC/LCT or NORM. Whether FEC is used or not, and the kind of FEC scheme used, is to some extent transparent to FCAST.

FCAST is compatible with both IPv4 and IPv6. Nothing in the FCAST specification has any implication on the source or destination IP address type.

The delivery service provided by FCAST might be fully reliable, or only partially reliable depending upon:

- the way ALC or NORM is used (e.g., whether FEC encoding and/or NACK-based repair requests are used or not),
- the way the FCAST carousel is used (e.g., whether the objects are made available for a long time span or not), and
- the way in which FCAST itself is employed (e.g., whether there is a session control application that might automatically extend an existing FCAST session until all receivers have received the transmitted content).

The receiver set can be restricted to a single receiver or possibly possibly a very large number of receivers. While the choice of the underlying transport protocol (i.e., ALC or NORM) and its parameters may limit the practical receiver group size, nothing in FCAST itself limits it. For instance, if FCAST/ALC is used on top of purely unidirectional transport channels, with no feedback information at all, which is the default mode of operation, then the scalability is maximum since neither FCAST, nor ALC, UDP or IP generates any feedback message. On the contrary, the FCAST/NORM scalability is typically limited by NORM scalability itself. For example, NORM can be configured to operate using proactive FEC without feedback similar to ALC, with receivers configured to provide NACK and optionally ACK feedback, or a hybrid combination of these. Similarly, if FCAST is used along with a session control application that collects reception information from the receivers, then this session control application may limit the scalability of the global object delivery system. This situation can of course be mitigated by using a hierarchy of feedback message aggregators or servers. The details of this are out of the scope of the present document.

The content of a carousel instance MAY be described by means of an OPTIONAL Carousel Instance Descriptor (CID) (Section 3.5). The decisions of whether a CID should be used or not, how often and when a CID should be sent, are left to the sender and depend on many parameters, including the target use case and the session dynamics. For instance it may be appropriate to send a CID at the beginning of each new carousel instance, and then periodically. These operational aspects are out of the scope of the present document.

7. IANA Considerations

This specification requires IANA to create two new registries.

7.1 Creation of the FCAST Object Meta-Data Format Registry

This document requires IANA to create a new registry, "FCAST Object Meta-Data Format" (MDFmt), with a reference to this document. The registry entries consist of a numeric value from 0 to 15, inclusive (i.e., they are 4-bit positive integers) that define the format of the object meta-data (see Section 2.1).

The initial value for this registry registry is defined below. Future assignments are to be made through Expert Review with Specification Required [RFC5226].

Value	Format Name	Format Reference	Reference
0 (default)	HTTP/1.1 metainformation format	[RFC2616], Section 7.1	This specification

7.2 Creation of the FCAST Object Meta-Data Encoding Registry

This document requires IANA to create a new registry, "FCAST Object Meta-Data Encoding" (MDEnc), with a reference to this document. The registry entries consist of a numeric value from 0 to 15, inclusive (i.e., they are 4-bit positive integers) that define the optional encoding of the Object Meta-Data field (see Section 2.1).

The initial values for this registry registry are defined below. Future assignments are to be made through Expert Review [RFC5226].

Value	Encoding Name	Encoding Reference	Reference
0 (default)	Plain Text	This specification	This specification
1	GZIP	[RFC1952]	This specification

7.3 Creation of the FCAST Object Meta-Data Types Registry

This document requires IANA to create a new registry, "FCAST Object Meta-Data Types" (MDType), with a reference to this document. The registry entries consist of additional text meta-data type identifiers and descriptions for meta-data item types that are specific to FCAST operation and not previously defined in [RFC1952]. The initial values are those described in Section 3.3 of this specification. This table summarizes those initial registry entries. Future assignments are to be made through Expert Review [RFC5226].

Meta-Data Type	Description	Reference
Fcast-Obj-Slice-Nb	Unsigned 32-bit integer that contains the total number of slices. A value greater than 1 indicates that this object is the result of a split of the original object	This specification
Fcast-Obj-Slice-Idx	Unsigned 32-bit integer that contains the slice index (in the {0 .. SliceNb - 1} interval)	This specification
Fcast-Obj-Slice-Offset	Unsigned 64-bit integer that contains the byte offset at which this slice starts within the original object	This specification

8. Acknowledgments

The authors are grateful to the authors of [ALC-00] for specifying the first version of FCAST/ALC. The authors are also grateful to David Harrington, Gorry Fairhurst and Lorenzo Vicisano for their valuable comments.

9. References

9.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5651] Luby, M., Watson, M., and L. Vicisano, "Layered Coding Transport (LCT) Building Block", RFC 5651, October 2009.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, November 2009.

[RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, April 2010.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

9.2 Informative References

[ALC-00] Luby, M., Gemmell, G., Vicisano, L., Crowcroft, J., and B. Lueckenhoff, "Asynchronous Layered Coding: a Scalable Reliable Multicast Protocol", March 2000.

[RFC6726] Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport", RFC 6726, November 2012.

[I-D.ietf-rmt-sec-discussion] Adamson, B., Roca, V., and H. Asaeda, "Security and Reliable Multicast Transport Protocols: Discussions and Guidelines", Internet-Draft draft-ietf-rmt-sec-discussion-06 (work in progress), March 2011.

[RFC3365] Schiller, J., "Strong Security Requirements for Internet Engineering Task Force Standard Protocols", BCP 61, RFC 3365, August 2002.

[RFC1071] Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum", RFC 1071, September 1988.

[RFC1952] Deutsch, P., Gailly, J-L., Adler, M., Deutsch, L.P., and G. Randers-Pehrson, "GZIP file format specification version 4.3", RFC 1952, May 1996.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J.D., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, August 2007.

[RFC5510] Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes", RFC 5510, April 2009.

[RFC5776] Roca, V., Francillon, A., and S. Faurite, "Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 5776, April 2010.

[RFC6584] Roca, V., "Simple Authentication Schemes for the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 6584, April 2012.

Authors' Addresses

Vincent Roca
INRIA
655, av. de l'Europe
Inovallee; Montbonnot
ST ISMIER cedex, 38334
France
EMail: vincent.roca@inria.fr
URI: http://planete.inrialpes.fr/people/roca/

Brian Adamson
Naval Research Laboratory
Washington, DC, 20375
USA
EMail: adamson@itd.nrl.navy.mil
URI: http://cs.itd.nrl.navy.mil

A. FCAST Examples

A.1 Introduction

This appendix provides informative examples of FCAST compound object header and carousel instance descriptor formats.

A.2 Regular Compound Object Example

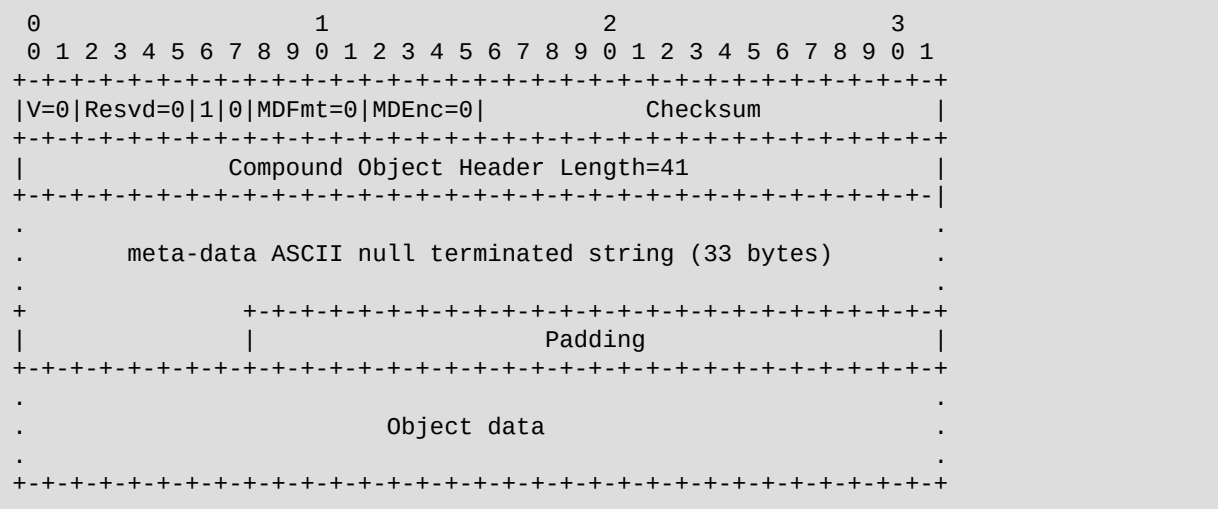

Figure 4: Compound Object Example.

Figure 4 shows a regular Compound Object where the meta-data ASCII string, in HTTP/1.1 meta-information format (MDFmt=0) contains:

```
Content-Location: example.txt <CR-LF>
```

This string is 33 bytes long, including the NULL-termination character. There is no GZIP encoding of the meta-data (MDEnc=0) and there is no Content-Length information either since this length can easily be calculated by the receiver as the FEC OTI transfer length minus the header length. Finally, the checksum encompasses the whole Compound Object (G=1).

A.3 Carousel Instance Descriptor Example

Figure 5 shows an example CID object, in the case of a static FCAST session, i.e., a session where the set of objects is set once and for all. There is no meta-data in this example since Fcast-CID-Complete and Fcast-CID-ID are both implicit.

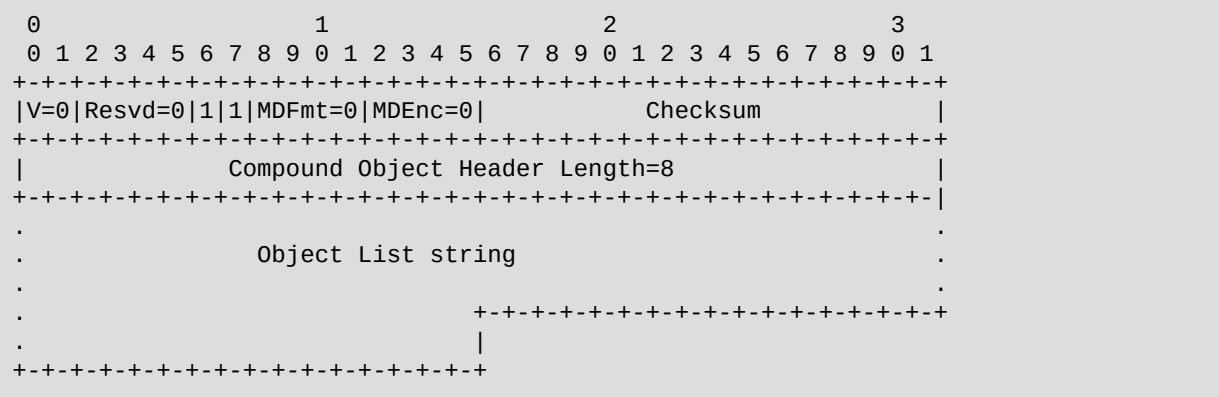

Figure 5: Example of CID, in case of a static session.

The object list contains the following 26 byte long string, including the NULL-termination character:

```
1,2,3,100-104,200-203,299
```

There are therefore a total of 3+5+4+1 = 13 objects in the carousel instance, and therefore in the FCAST session. There is no meta-data associated to this CID. The session being static and composed of a single Carousel Instance, the sender did not feel the necessity to carry a Carousel Instance ID meta-data.

B. Additional Meta-Data Transmission Mechanisms

B.1 Supporting Additional Mechanisms

In certain use-cases, FCAST can take advantage of another in-band (e.g., via NORM_INFO messages Appendix B.2) or out-of-band signaling mechanism. This section provides an overview of how other signaling mechanism could be employed and a normative specification for how FCAST information is embedded when NORM_INFO messages are used for FCAST compound message headers. Such additional signaling schemes can be used to carry the whole meta-data, or a subset of it, ahead of time, before the associated compound object. Therefore a receiver could be able to decide in advance, before beginning the reception of the compound object, whether the object is of interest or not, based on the information retrieved by this way, which mitigates FCAST limitations. While out-of-band techniques are out of the scope of this document, we explain below how this can be achieved in case of FCAST/NORM.

Supporting additional mechanisms is OPTIONAL in FCAST implementations. In any case, an FCAST sender MUST continue to send all the required meta-data in the compound object, even if the whole meta-data, or a subset of it, is sent by another mechanism (Section 5). Additionally, when meta-data is sent several times, there MUST NOT be any contradiction in the information provided by the different mechanisms. In case a mismatch is detected, the meta-data contained in the Compound Object MUST be used as the definitive source.

When meta-data elements are communicated out-of-band, in advance of data transmission, the following piece of information can be useful:

- TOI: a positive integer that contains the Transmission Object Identifier (TOI) of the object, in order to enable a receiver to easily associate the meta-data to the object. The valid range for TOI values is discussed in Section 3.6;

B.2 Using NORM_INFO Messages with FCAST/NORM

The NORM_INFO message of NORM can convey "out-of-band" content with respect to a given transport object. With FCAST, this message MAY be used as an additional mechanism to transmit meta-data. In that case, the NORM_INFO message carries a new Compound Object that contains all the meta-data of the original object, or a subset of it. The NORM_INFO Compound Object MUST NOT contain any Object Data field (i.e., it is only composed of the header), it MUST feature a non global checksum, and it MUST NOT include any padding field. Finally, note that the availability of NORM_INFO for a given object is signaled through the use of a dedicated flag in the NORM_DATA message header. Along with NORM's NACK-based repair request signaling, it allows a receiver to quickly (and independently) request an object's NORM_INFO content. However, a limitation here is that the NORM_INFO Compound Object header MUST fit within the byte size limit defined by the NORM sender's configured "segment size" (typically a little less than the network MTU);

B.2.1 Example

In case of FCAST/NORM, the FCAST Compound Object meta-data (or a subset of it) can be carried as part of a NORM_INFO message, as a new Compound Object that does not contain any Compound Object Data. In the following informative example we assume that the whole meta-data is carried in such a message for a certain Compound Object. Figure 6 shows an example NORM_INFO message that contains the FCAST Compound Object Header and meta-data as its payload. In this example, the first 16 bytes are the NORM_INFO base header, the next 12 bytes are a NORM EXT_FTI header extension containing the FEC Object Transport Information for the associated object, and the remaining bytes are the FCAST Compound Object Header and meta-data. Note that "padding" MUST NOT be used and that the FCAST checksum only encompasses the Compound Object Header (G=0).

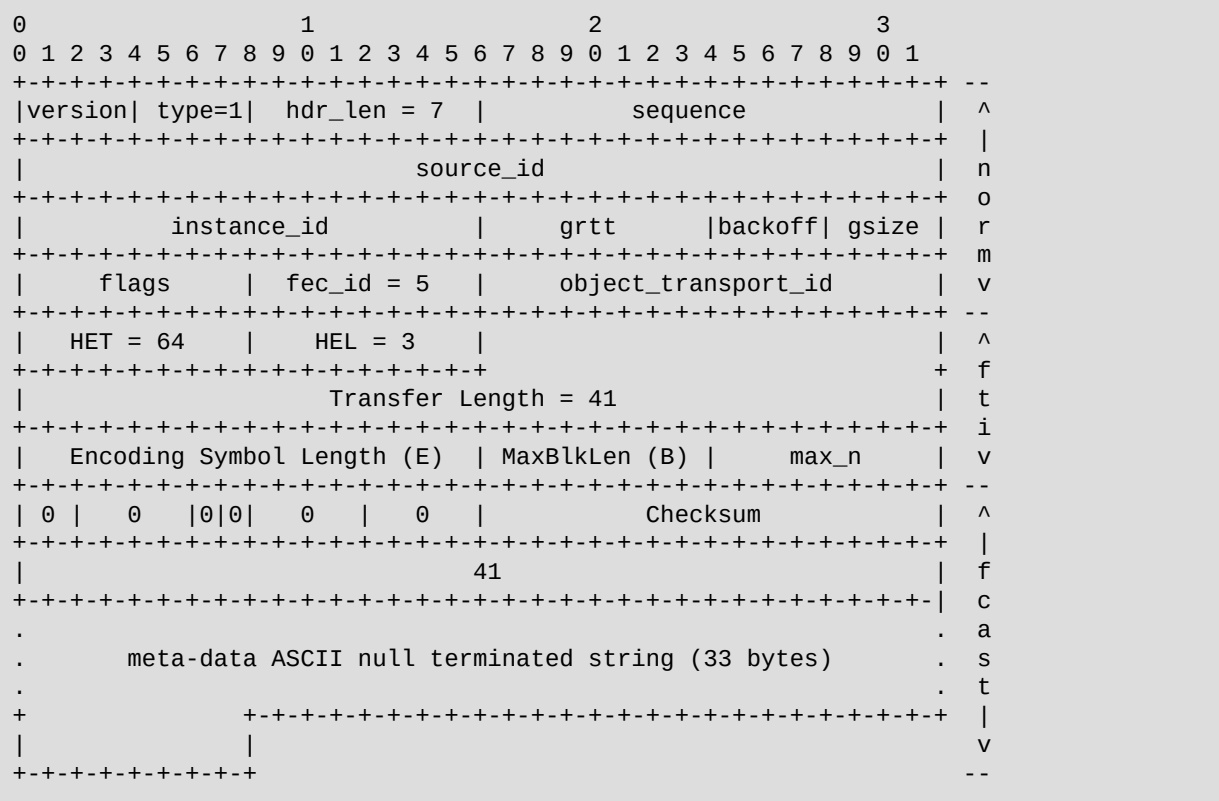

Figure 6: NORM_INFO containing an EXT_FTI header extension and an FCAST Compound Object Header

The NORM_INFO message shown in Figure 6 contains the EXT_FTI header extension to carry the FEC OTI. In this example, the FEC OTI format is that of the Reed-Solomon FEC coding scheme for fec_id = 5 as described in [RFC5510]. Other alternatives for providing the FEC OTI would have been to either include it directly in the meta-data of the FCAST Compound Header, or to include an EXT_FTI header extension to all NORM_DATA packets (or a subset of them). Note that the NORM "Transfer_Length" is the total length of the associated FCAST Compound Object, i.e., 41 bytes.

The FCAST Compound Object in this example does contain the same meta-data and is formatted as in the example of Figure 4. With the combination of the FEC_OTI and the FCAST meta-data, the NORM protocol and FCAST application have all of the information needed to reliably receive and process the associated object. Indeed, the NORM protocol provides rapid (NORM_INFO has precedence over the associated object content), reliable delivery of the NORM_INFO message and its payload, the FCAST Compound Object.

Full Copyright Statement

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org[1].

[1] mailto:ietf-ipr@ietf.org