[MS-RTVPF]:

RTP Payload Format for RT Video Streams Extensions

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Preliminary Documentation. This particular Open Specifications document provides documentation for past and current releases and/or for the pre-release version of this technology. This document provides final documentation for past and current releases and preliminary documentation, as applicable and specifically noted in this document, for the pre-release version. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. Because this documentation might change between the pre-release version and the final

version of this technology, there are risks in relying on this preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1	New	Initial version
4/25/2008	0.2	Minor	Updated based on feedback
6/27/2008	1.0	Major	Updated and revised the technical content.
8/15/2008	1.01	Minor	Revised and edited the technical content.
12/12/2008	2.0	Major	Updated and revised the technical content.
2/13/2009	2.01	Minor	Updated the technical content.
3/13/2009	2.02	Minor	Updated the technical content.
7/13/2009	2.03	Major	Revised and edited the technical content
8/28/2009	2.04	Editorial	Revised and edited the technical content
11/6/2009	2.05	Editorial	Revised and edited the technical content
2/19/2010	2.06	Editorial	Revised and edited the technical content
3/31/2010	2.07	Major	Updated and revised the technical content
4/30/2010	2.08	Editorial	Revised and edited the technical content
6/7/2010	2.09	Minor	Updated the technical content
6/29/2010	2.10	Editorial	Changed language and formatting in the technical content.
7/23/2010	2.10	None	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	3.0	Major	Significantly changed the technical content.
11/15/2010	3.0	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	3.0	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	3.0	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	3.0	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	4.0	Major	Significantly changed the technical content.
4/11/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	4.0.1	Editorial	Changed language and formatting in the technical content.
2/11/2013	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
7/30/2013	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.
3/30/2015	5.0	Major	Significantly changed the technical content.
9/4/2015	5.0	None	No changes to the meaning, language, or formatting of the technical content.
7/15/2016	5.0	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2016	5.0	None	No changes to the meaning, language, or formatting of the technical content.
4/27/2018	6.0	Major	Significantly changed the technical content.
7/24/2018	7.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The RTP Payload Format for RTVideo Streams Extensions [MS-RTVPF] protocol is a proprietary protocol describing the payload format for carrying real-time video streams in the payload of the Real-Time Transport Protocol (RTP). It is used to transmit and receive real-time video streams in two-party peer-to-peer calls and in multi-party conference calls.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

B-frame: A bidirectional video frame that references both the previous frame and the next frame.

big-endian: Multiple-byte values that are byte-ordered with the most significant byte stored in the memory location with the lowest address.

cached frame: A video frame that is cached for later use by an encoder and a decoder. A cached frame acts as a reference frame for the next **Super P-frame (SP-frame)**. I-frames and SP-frames typically are cached frames.

endpoint: A device that is connected to a computer network.

entry point header: A header field whose values specify the horizontal and vertical dimensions of a video frame. See also **sequence header**.

forward error correction (FEC): A process in which a sender uses redundancy to enable a receiver to recover from packet loss.

GOP: A group of pictures that starts with one **I-frame** and ends with the next I-frame, excluding the next I-frame, as described in [SMPTE-VC-1].

I-frame: A video frame that is encoded as a single image, such that it can be decoded without any dependencies on previous frames. Also referred to as Intra-Coded frame, Intra frame, and key frame.

maximum transmission unit (MTU): The size, in bytes, of the largest packet that a given layer of a communications protocol can pass onward.

network byte order: The order in which the bytes of a multiple-byte number are transmitted on a network, most significant byte first (in **big-endian** storage). This may or may not match the order in which numbers are normally stored in memory for a particular processor.

P-frame: A predicative video frame that references a previous frame. Also referred to as inter-coded frame or inter-frame.

Real-Time Transport Protocol (RTP): A network transport protocol that provides end-to-end transport functions that are suitable for applications that transmit real-time data, such as audio and video, as described in [RFC3550].

RTP packet: A data packet consisting of the fixed RTP header, a possibly empty list of contributing sources, and the payload data. Some underlying protocols may require an encapsulation of the RTP packet to be defined. Typically one packet of the underlying protocol contains a single RTP packet, but several RTP packets can be contained if permitted by the encapsulation method. See [RFC3550] section 3.

RTP payload: The data transported by **RTP** in a packet, for example audio samples or compressed video data. For more information, see [RFC3550] section 3.

RTVC1: A Microsoft proprietary implementation of the VC1 codec for real-time transmission purposes, as described in [SMPTE-VC-1]. Microsoft extensions to VC1 are based on **cached frame** and **SP-frame**, as described in [MS-RTVPF].

RTVideo: A video stream that carries an **RTVC1** bit stream.

RTVideo FEC metadata packet: A packet that is generated by using the forward error correction (FEC) algorithm to provide redundancy. It is packetized in the RTVideo FEC Real-Time Transport Protocol (RTP) payload format.

RTVideo frame: A video frame that is encoded by using an **RTVC1** codec.

sequence header: A set of encoding and display parameters that are placed before a group of pictures, as described in [SMPTE-VC-1]. See also **entry point header**.

Super P-frame (SP-frame): A special P-frame that uses the previous **cached frame** instead of the previous **P-frame** or **I-frame** as a reference frame.

video data packet: A video data block that encapsulates a complete video frame or a fragment of a video frame. It contains the video payload header and the video payload.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[MS-RTP] Microsoft Corporation, "Real-time Transport Protocol (RTP) Extensions".

[MS-SDPEXT] Microsoft Corporation, "Session Description Protocol (SDP) Version 2.0 Extensions".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[SMPTE-VC-1] Society of Motion Picture and Television Engineers, "VC-1 Compressed Video Bitstream Format and Decoding Process", SMPTE 421M-2006, 2006, http://www.smpte.org/standards

Note There is a charge to download this specification.

1.2.2 Informative References

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003, http://www.ietf.org/rfc/rfc3550.txt

1.3 Overview

This protocol specifies a payload format to transport an **RTVC1** bitstream using the **Real-Time Transport Protocol (RTP)**.

This protocol accepts an RTVC1-encoded video frame. It fragments the video frame into one or more packets enumerated in a data packet list containing **RTVideo.** Each RTVideo **video data packet** contains an RTVideo payload header and a video payload. The RTVideo video data packet list optionally has one or more (up to 31) **RTVideo FEC metadata packets** appended to the end of the list. Each RTVideo FEC metadata packet contains an RTVideo **forward error correction (FEC)** payload header and FEC metadata.

1.4 Relationship to Other Protocols

This protocol carries the **RTVC1** bitstream, described in [SMPTE-VC-1], as a payload, and in turn is carried as a payload in **RTP**, as described in [MS-RTP].

1.5 Prerequisites/Preconditions

This protocol specifies only the payload format for **RTVideo** video streams. This protocol requires the establishment of an **RTP** stream, a mechanism to obtain RTVideo video frames for it to packetize, and a mechanism to render RTVideo video frames that it has depacketized.

Higher layers are required to provide **RTVideo frames** with the following added information about each frame:

I-frame Flag: Specifies whether the frame is an I-frame.

SP-frame Flag: Specifies whether the frame is a Super P-frame (SP-frame).

cached frame Flag: Specifies whether the frame is a cached frame.

sequence header: This is required for each I-frame. It is not needed for other frame types.

Higher layers are required to provide video frames in referencing order. Frames being referenced are required to be provided earlier than frames referring to them.

Higher layers are also required to respect the following assumptions:

- An I-frame does not have any reference frame.
- I-frames and SP-frames are cached frames as well.
- An SP-frame refers to the previous cached frame.
- A **P-frame** refers to the previous P-frame, SP-frame, or I-frame.
- A **B-frame** refers to the previous P-frame, SP-frame, or I-frame.

1.6 Applicability Statement

This protocol is only applicable for transporting video frames encoded using the **RTVC1** codec.

1.7 Versioning and Capability Negotiation

This protocol has the following versioning constraints:

- **Supported Transports:** This protocol uses **RTP** as its transport as discussed in section 2.1.

- **Protocol Versions:** This protocol supports **FEC** Version 0 and FEC Version 1 as discussed in section 2.2.

The RTP Payload Format for RTVideo Streams Extensions protocol does not have any capability negotiation constraints.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol is a payload for the [MS-RTP] transport protocol and therefore relies on **RTP** for providing means to transport its payload over the network.

2.2 Message Syntax

This protocol defines four **RTP payload** formats:

RTVideo Basic RTP Payload Format:

The RTVideo Basic RTP Payload Format provides a basic scheme to packetize and transport **RTVideo frames** between a sender and receiver. It provides enough information to allow the receiver to reconstruct the video frames.

RTVideo Extended RTP Payload Format:

The RTVideo Extended RTP Payload Format extends the RTVideo Basic RTP Payload Format with extra fields. A video frame counter and the video frame counter of its reference frame are added. This provides a way for the receiver to actively drop frames received but not decodable because of packet loss. The receiver can consider a video frame as lost if one or more packets in the video frame are lost. It can then optionally drop the video frame that references the lost video frame, because the **RTVideo** decoder cannot decode the video frame.

RTVideo Extended 2 RTP Payload Format:

The RTVideo Extended 2 RTP Payload Format extends the RTVideo Extended RTP Payload Format. It carries four extra bytes.

RTVideo FEC RTP Payload Format:

The RTVideo FEC RTP Payload Format is a special case of the RTVideo Extended 2 RTP Payload Format that provides a way to protect against a frame loss contributed to a packet loss. When this protocol is applied, one or more (up to 31) **FEC** metadata packets are added to the end of the video packet list for a video frame.

The FEC metadata packets carry metadata calculated over the packet list using an FEC algorithm. If one of the packets in the packet list is lost, this lost packet can be reconstructed using the rest of the packets and the FEC metadata packet.

In this document all the fields in the payload format headers are in **big-endian** byte order, also called **network byte order**.

2.2.1 RTP Header Usage

The syntax of the **RTP** header is specified in [MS-RTP] section 2.2.1. The fields of the fixed RTP header have their usual meaning with the following additional notes:

Marker (M): This bit MUST be set to "1" if the **RTP packet** contains the last packet for the video frame. Otherwise, it MUST be set to "0". This last packet can be the last fragment of the video frame or an **FEC** metadata packet generated by the FEC algorithm.

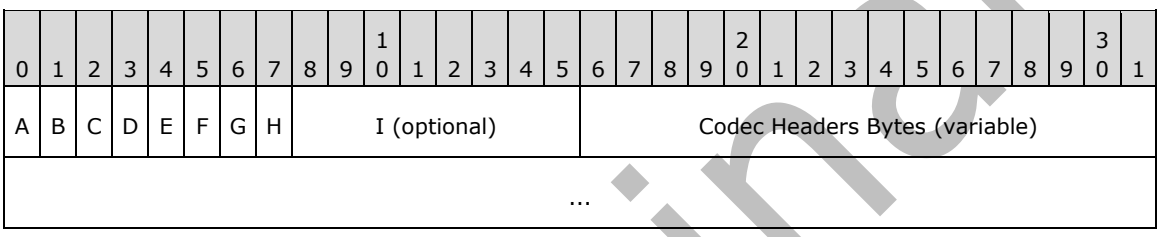
Timestamp: The syntax of this field is defined in [RFC3550], section 5.1. The sampling clock frequency MUST be 90000 Hz.

For the RTVideo FEC RTP Payload Format (section 2.2.5), the following guidance is provided for a server forwarding video packets, in case of packet loss on the link from the sender to the server:

- If a server does not receive all video packets in a video frame but there are enough FEC metadata packets to reconstruct the lost video packets, it MUST still forward all video packets.
- For any video packet that the server does not receive, it MUST forward an empty packet in replacement of the lost packet. This empty packet MUST have a valid RTP header for **RTVideo**, MUST NOT have any payload data and MUST use the same RTP sequence number space as the FEC and RTVideo packets.<1>
- Such an empty RTP packet MUST be sent for every packet lost on the uplink.

2.2.2 RTVideo Basic RTP Payload Format

The size of the RTVideo Basic Payload Format header varies. The minimum size is one byte without the codec headers present. If the codec headers are present, the maximum size is 65 bytes.



A - M (1 bit): Payload format mode. This field MUST be set to zero in the RTVideo Basic RTP Payload Format mode. The field is set to one in other **RTP payload** formats, as specified in sections 2.2.3, 2.2.4, and 2.2.5.

B - C (1 bit): **Cached frame** flag. A value of one specifies a cached frame. A value of zero specifies the frame is not a cached frame.

C - SP (1 bit): **Super P-frame (SP-frame)** flag. A value of one specifies an SP-frame. A value of zero specifies the frame is not an SP-frame.

D - L (1 bit): Last packet flag. Indicates whether this packet is the last packet of the video frame, excluding **FEC** metadata packets. A value of one specifies the last packet. A value of zero specifies it is not the last packet.

E - O (1 bit): MUST be set to one.

F - I (1 bit): **I-frame** flag. Indicates whether the frame is an I-frame. A value of one indicates the frame is an I-frame. A value of zero indicates that the frame is not an I-frame, but rather a SP-frame, **P-frame**, or **B-frame**.

G - S (1 bit): Codec headers presence flag. Indicates presence of the codec headers. A value of one indicates the **Codec Headers Length** field is present. A value of zero indicates the **Codec Headers Length** field is not present.

H - F (1 bit): First packet flag. Indicates whether the packet is the first packet of the video frame, excluding FEC metadata packets. A value of one indicates the packet is the first packet. A value of zero indicates it is not the first packet.

I - Codec Headers Length (1 byte, optional): The size of **Codec Headers Bytes** field. Only present when the **S** bit is one. The value of this field MUST be less than or equal to 63.

Codec Headers Bytes (variable): Codec headers (binding byte, **sequence header** and **entry point header**). Only present when the **S** bit is one and the **Codec Headers Length** is greater than zero. The size is indicated by the **Codec Headers Length** field.

Codec headers include the binding byte, the sequence header, and the entry point header. The sequence header and the entry point header are specified in [SMPTE-VC-1].

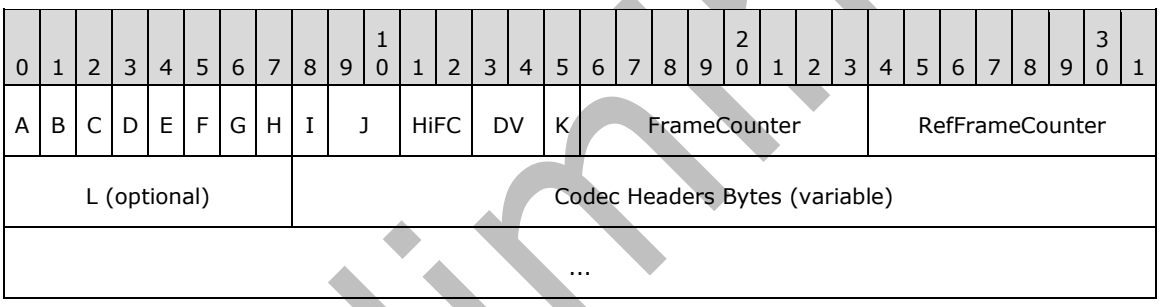
The binding byte is the first byte of the **codecs headers bytes**. It indicates whether B-frames are present in the bitstream. Only two values are defined: 0x25 and 0x27. A binding byte of 0x25 indicates B-frames are present. A binding byte of 0x27 indicates B-frames are not present.

The entry point header SHOULD be carried at the beginning of the video payload data. When present, the entry point header carried in the video payload data supersedes the entry point header carried in the **Codec Headers Bytes** field of the payload format header.<2>

2.2.3 RTVideo Extended RTP Payload Format

The size of the RTVideo Extended RTP Payload Format header varies. The minimum size is 4 bytes without the codec headers present. With codec headers present, the maximum size is 68 bytes.

The frame counters described in the following paragraphs are meaningful only within a video **GOP**. The counter starts at zero for the first frame in a GOP and increments by one for every succeeding frame. The frame counter is reset to zero at the beginning of the next GOP.



A - M (1 bit): Payload format mode. It MUST be set to one in the RTVideo Extended RTP Payload Format.

B - C (1 bit): **Cached frame** flag. A value of one specifies a cached frame. A value of zero specifies the frame is not a cached frame. The decoder on the receiver side MUST cache the cached frame because the next **SP-frame** references it.

C - SP (1 bit): Super P-frame (SP-frame) flag. A value of one specifies an SP-frame. A value of zero specifies the frame is not an SP-frame.

D - L (1 bit): Last packet flag. Indicates whether this packet is the last packet of the video frame, excluding **FEC** metadata packets. A value of one specifies the last packet. A value of zero specifies it is not the last packet.

E - O (1 bit): MUST be set to one.

F - I (1 bit): **I-frame** flag. Indicates whether the frame is an I-frame. A value of one indicates the frame is an I-frame. A value of zero indicates it is an SP-frame, **P-frame**, or **B-frame**.

G - S (1 bit): Codec headers presence flag. Indicates presence of the Codec Headers. A value of one indicates the **Codec Headers Length** field is present. A value of zero indicates the **Codec Headers Length** field is not present.

H - F (1 bit): First packet flag. Indicates whether the packet is the first packet of the video frame, excluding FEC metadata packets. A value of one indicates the packet is the first packet. A value of zero indicates it is not the first packet.

I - M2 (1 bit): Payload format mode2 flag. This field MUST be set to zero in the RTVideo Extended RTP Payload Format. The field is set to one in other **RTP payload** formats as specified in sections 2.2.4 and 2.2.5.

J - HiRFC (2 bits): The **HiRFC** field and **RefFrameCounter** field together specify the video frame counter (10 bits) for the reference frame or the reference frames, if the current frame is a B-frame, of this frame.

If the video frame is an I-frame, P-frame, or SP-frame, the whole 10-bit field specifies a frame counter.

The **HiRFC** field specifies the 2 high bits of the counter. The **RefFrameCounter** specifies the 8 low bits.

If the video frame is a B-frame, the **HiRFC** field MUST be set to zero. The 4 high bits of the **RefFrameCounter** specify one reference frame counter delta, **RefFrameCounterDelta1**, and the 4 low bits of the **RefFrameCounter** specify another reference frame counter delta, **RefFrameCounterDelta2**. These two reference counter delta values correspond to the frame counters, referred to as **RefFrameCounter1** and **RefFrameCounter2**, of the two reference frames for the B-frame, respectively.

The two reference frame counters are calculated by subtracting the frame counter delta from the frame counter for the B-frame. If the B-frame references a single frame only, the two reference counters MUST be the same.

HiFC (2 bits): The **HiFC** field and the **FrameCounter** field together specify the video frame counter (10 bits) for the video frame. The **HiFC** field specifies the 2 high bits. **FrameCounter** specifies the 8 low bits.

DV (2 bits): The **Data version** field. The value ranges from zero to three. This field SHOULD be ignored when receiving an RTVideo Extended RTP Payload Format packet and MUST be set to zero when sending an RTVideo Extended RTP Payload Format packet.

K - E (1 bit): The **ExtraData** field. In this document, the FEC metadata is considered to be extra data. A value of one specifies the packet is an FEC metadata packet. A value of zero specifies it is an **RTVideo video data packet**. There is no extra data defined in the RTVideo Extended RTP Payload Format header. For this reason, this field MUST be set to zero in the RTVideo Extended RTP Payload Format header.

FrameCounter (1 byte): The **HiFC** field and the **FrameCounter** field together specify the video frame counter (10 bits) for the video frame. The **HiFC** field specifies the 2 high bits. **FrameCounter** specifies the 8 low bits.

RefFrameCounter (1 byte): The **HiRFC** field and **RefFrameCounter** field together specify the video frame counter (10 bits) for the reference frame or the reference frames, if the current frame is a B-frame, of this frame.

If the video frame is an I-frame, P-frame, or SP-frame, the whole 10-bit field specifies a frame counter.

The **HiRFC** field specifies the 2 high bits of the counter. The **RefFrameCounter** specifies the 8 low bits.

If the video frame is a B-frame, the **HiRFC** field MUST be set to zero. The 4 high bits of the **RefFrameCounter** specify one reference frame counter delta, **RefFrameCounterDelta1**, and the 4 low bits of the **RefFrameCounter** specify another reference frame counter delta,

RefFrameCounterDelta2. These two reference counter delta values correspond to the frame counters, referred to as **RefFrameCounter1** and **RefFrameCounter2**, of the two reference frames for the B-frame, respectively.

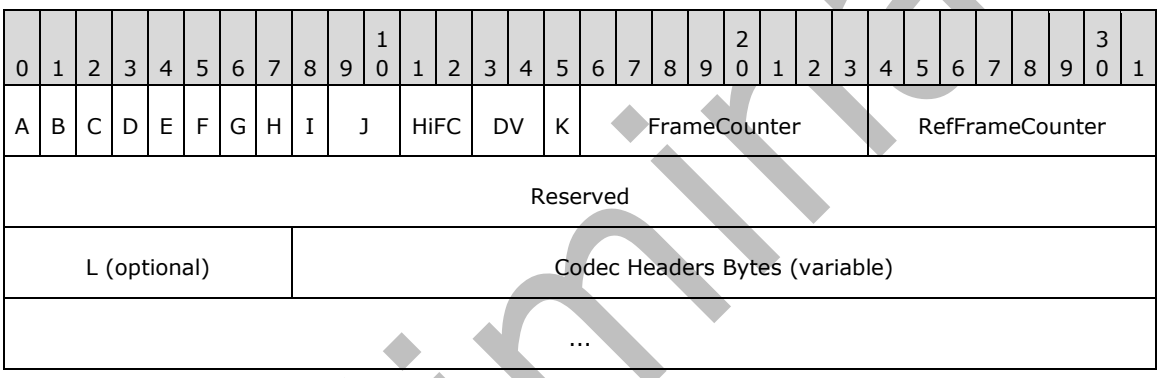
The two reference frame counters are calculated by subtracting the frame counter delta from the frame counter for the B-frame. If the B-frame references a single frame only, the two reference counters MUST be the same.

L - Codec Headers Length (1 byte, optional): The size of **Codec Headers Bytes** field. Only present when the **S** bit is one. The value of this field MUST be less than or equal to 63.

Codec Headers Bytes (variable): Codec headers (binding byte, **sequence header** and **entry point header**). Only present when the **S** bit is one and the **Codec Headers Length** is greater than zero. The size is indicated by the **Codec Headers Length** field.

2.2.4 RTVideo Extended 2 RTP Payload Format

The size of the RTVideo Extended 2 RTP Payload Format header varies. The minimum size is 8 bytes without codec headers present. With codec headers present, the maximum size is 72 bytes.



A - M (1 bit): Payload format mode. It MUST be set to one in the RTVideo Extended 2 RTP Payload Format.

B - C (1 bit): **Cached frame** flag. A value of one specifies a cached frame. A value of zero specifies the frame is not a cached frame. The decoder on the receiver side MUST cache the cached frame because the next **SP-frame** references it.

C - SP (1 bit): Super P-frame (SP-frame) flag. A value of one specifies an SP-frame. A value of zero specifies the frame is not an SP-frame.

D - L (1 bit): Last packet flag. Indicates whether this packet is the last packet of the video frame, excluding **FEC** metadata packets. A value of one specifies the last packet. A value of zero specifies that it is not the last packet.

E - O (1 bit): MUST be set to one.

F - I (1 bit): **I-frame** flag. Indicates whether the frame is an I-frame. A value of one indicates the frame is an I-frame. A value of zero indicates that the frame is not an I-frame, but rather a SP-frame, **P-frame**, or **B-frame**.

G - S (1 bit): Codec headers presence flag. Indicates presence of the Codec Headers. A value of one indicates the **Codec Headers Length** field is present. A value of zero indicates that the **Codec Headers Length** field is not present.

H - F (1 bit): First packet flag. Indicates whether the packet is the first packet of the video frame, excluding FEC metadata packets. A value of one indicates the packet is the first packet. A value of zero indicates it is not the first packet.

I - M2 (1 bit): Payload format mode2 flag. This field MUST be set to one in the RTVideo Extended 2 RTP Payload Format.

J - HiRFC (2 bits): The **HiRFC** field and **RefFrameCounter** field together specify the video frame counter (10 bits) for the reference frame or the reference frames, if the current frame is a B-frame of this frame.

If the video frame is an I-frame, P-frame, or SP-frame, the whole 10-bit field specifies a frame counter.

The **HiRFC** field specifies the 2 high bits of the counter. The **RefFrameCounter** specifies the 8 low bits.

If the video frame is a B-frame, the **HiRFC** field MUST be set to zero. The 4 high bits of the **RefFrameCounter** specify one reference frame counter delta, **RefFrameCounterDelta1**, and the 4 low bits of the **RefFrameCounter** specify another reference frame counter delta, **RefFrameCounterDelta2**. These two reference counter delta values correspond to the frame counters, referred to as **RefFrameCounter1** and **RefFrameCounter2**, of the two reference frames for the B-frame, respectively.

The two reference frame counters are calculated by subtracting the frame counter delta from the frame counter for the B-frame. If the B-frame references a single frame only, the two reference counters MUST be the same.

HiFC (2 bits): The **HiFC** field and the **FrameCounter** field together specify the video frame counter (10 bits) for the video frame. The **HiFC** field specifies the 2 high bits. **FrameCounter** specifies the 8 low bits.

DV (2 bits): The **Data version** field. The value ranges from zero to three. This field SHOULD be ignored when receiving an RTVideo Extended 2 RTP Payload Format packet, and MUST be set to zero when sending an RTVideo Extended 2 RTP Payload Format packet.

K - E (1 bit): The **ExtraData** field. In this document, the FEC metadata is considered to be extra data. A value of one specifies the packet is an FEC metadata packet. A value of zero specifies it is an **RTVideo video data packet**. There is no extra data defined in the RTVideo Extended 2 RTP Payload Format header. For this reason, this field MUST be set to zero in the RTVideo Extended 2 RTP Payload Format header.

FrameCounter (1 byte): The **HiFC** field and the **FrameCounter** field together specify the video frame counter (10 bits) for the video frame. The **HiFC** field specifies the 2 high bits. **FrameCounter** specifies the 8 low bits.

RefFrameCounter (1 byte): The **HiRFC** field and **RefFrameCounter** field together specify the video frame counter (10 bits) for the reference frame or the reference frames, if the current frame is a B-frame, of this frame.

If the video frame is an I-frame, P-frame, or SP-frame, the whole 10-bit field specifies a frame counter.

The **HiRFC** field specifies the 2 high bits of the counter. The **RefFrameCounter** specifies the 8 low bits.

If the video frame is a B-frame, the **HiRFC** field MUST be set to zero. The 4 high bits of the **RefFrameCounter** specify one reference frame counter delta, **RefFrameCounterDelta1**, and the 4 low bits of the **RefFrameCounter** specify another reference frame counter delta, **RefFrameCounterDelta2**. These two reference counter delta values correspond to the frame

counters, referred to as **RefFrameCounter1** and **RefFrameCounter2**, of the two reference frames for the B-frame, respectively.

The two reference frame counters are calculated by subtracting the frame counter delta from the frame counter for the B-frame. If the B-frame references a single frame only, the two reference counters MUST be the same.

Reserved (4 bytes): Reserved field. This field MUST be set to zero when the **E** field is set to zero. This field is redefined when the **E** field is set to one, as specified in the RTVideo FEC RTP Payload Format header (section 2.2.5).

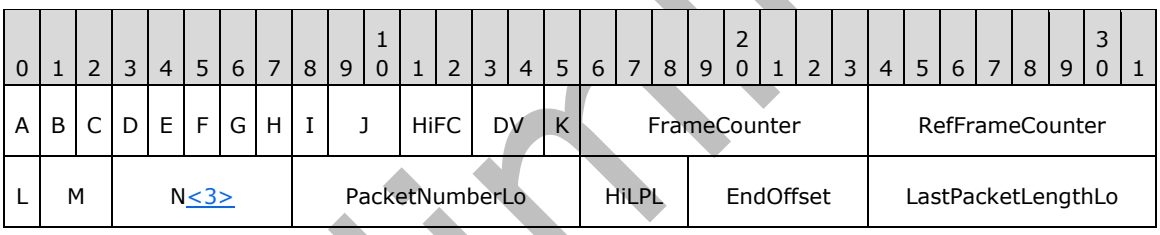
L - Codec Headers Length (1 byte, optional): The size of **Codec Headers Bytes** field. Only present when the **S** bit is one. The value of this field MUST be less than or equal to 63.

Codec Headers Bytes (variable): Codec headers (binding byte, **sequence header** and **entry point header**). Only present when the **S** bit is one and the **Codec Headers Length** is greater than zero. The size is indicated by the **Codec Headers Length** field.

2.2.5 RTVideo FEC RTP Payload Format

The RTVideo FEC RTP Payload Format header can be considered a special case of the RTVideo Extended 2 RTP Payload Format (section 2.2.4), where the **S** field MUST be set to zero and the **E** field MUST be set to one.

This RTVideo FEC Payload Format header has a fixed size of 8 bytes.



A - M (1 bit): Payload format mode. This field MUST be set to one in the RTVideo FEC RTP Payload Format.

B – C (1 bit): The **Cached frame** flag. A value of one specifies a cached frame. A value of zero specifies the frame is not a cached frame. The decoder on the receiver side MUST cache the cached frame because the next **SP-frame** references it.

C – SP (1 bit): Super P-frame (SP-frame) flag. A value of one specifies an SP-frame. A value of zero specifies the frame is not an SP-frame.

D – L (1 bit): Last packet flag. Indicates whether this packet is the last packet of the video frame, excluding **FEC** metadata packets. A value of one specifies the last packet. A value of zero specifies it is not the last packet.

E – O (1 bit): MUST be set to one.

F – I (1 bit): The **I-frame** flag. Indicates whether the frame is an I-frame. A value of one indicates the frame is an I-frame. A value of zero indicates that the frame is not an I-frame, but rather a SP-frame, **P-frame**, or **B-frame**.

G - S (1 bit): Codec headers presence flag. Indicates presence of the Codec Headers. A value of one indicates the **Codec Headers Length** field is present. A value of zero indicates the **Codec Headers Length** field is not present. The **S** field MUST be set to zero in the RTVideo FEC RTP Payload Format. This means the codec headers size and the codec headers data fields MUST NOT be present in the RTVideo FEC RTP Payload Format.

H – F (1 bit): First packet flag. Indicates whether the packet is the first packet of the video frame, excluding FEC metadata packets. A value of one indicates the packet is the first packet. A value of zero indicates it is not the first packet.

I - M2 (1 bit): Payload format mode2 flag. This field MUST be set to one in the in the RTVideo FEC RTP Payload Format.

HiRFC (2 bits): MUST be set to zero. The **HiRFC** field and **RefFrameCounter** field together specify the video frame counter (10 bits) for the reference frame or the reference frames, if the current frame is a B-frame, of this frame.

If the video frame is an I-frame, P-frame, or SP-frame, the whole 10-bit field specifies a frame counter.

The **HiRFC** field specifies the 2 high bits of the counter. The **RefFrameCounter** specifies the 8 low bits.

If the video frame is a B-frame, the **HiRFC** field MUST be set to zero. The 4 high bits of the **RefFrameCounter** specify one reference frame counter delta, **RefFrameCounterDelta1**, and the 4 low bits of the **RefFrameCounter** specify another reference frame counter delta, **RefFrameCounterDelta2**. These two reference counter delta values correspond to the frame counters, referred to as **RefFrameCounter1** and **RefFrameCounter2**, of the two reference frames for the B-frame, respectively.

The two reference frame counters are calculated by subtracting the frame counter delta from the frame counter for the B-frame. If the B-frame references a single frame only, the two reference counters MUST be the same.

HiFC (2 bits): MUST be set to zero. The **HiFC** field and the **FrameCounter** field together specify the video frame counter (10 bits) for the video frame. The **HiFC** field specifies the 2 high bits. **FrameCounter** specifies the 8 low bits.

DV (2 bits): The FEC version number. The value ranges from zero to three. The version number SHOULD<4> be 00 or 01 in the RTVideo FEC RTP Payload Format. Currently, it is set to zero for the first FEC packet and equal to 01 for subsequent FEC packets of the same frame.

K - E (1 bit): The **ExtraData** field. In this document, the FEC metadata is considered to be extra data. A value of one specifies the packet is an FEC metadata packet. This field MUST be set to one in the RTVideo FEC RTP Payload Format.

FrameCounter (1 byte): The **HiFC** field and the **FrameCounter** field together specify the video frame counter (10 bits) for the video frame. The **HiFC** field specifies the 2 high bits. **FrameCounter** specifies the 8 low bits.

RefFrameCounter (1 byte): MUST be set to zero.

L - M3 (1 bits): Payload format mode3 flag. This field MUST be set to zero in the RTVideo FEC Payload Format.

M - HiPN (2 bits): The **HiPN** field and **PacketNumberLo** field specify the number of video packets (10 bits) the video frame is fragmented into. The **HiPN** field specifies the 2 high bits. The **PacketNumberLo** field specifies the 8 low bits.

N - Reserved/FECPacketsNumber (5 bits): The semantics of these bits depends on the value of **DV**. If the value of **DV** is not "01", this is a field reserved for future use and MUST be set to zero by the sender and MUST be ignored by the receiver. If the value of **DV** is "01", this field represents a **FECPacketsNumber** field specifying the total number of contiguous FEC packets, generated by the FEC algorithm, associated with the packets conveying the video frame. The maximum number of FEC packets is limited to 31. If the value of **DV** is "01", **FECPacketsNumber** MUST NOT<5> be set to zero.

PacketNumberLo (8 bits): The **HiPN** field and **PacketNumberLo** field specify the number of video packets (10 bits) the video frame is fragmented into. The **HiPN** field specifies the 2 high bits. The **PacketNumberLo** field specifies the 8 low bits.

HiLPL (3 bits): The **HiLPL** field and **LastPacketLengthLo** field specify the size (11 bits) of the last **video data packet**. Both the video payload header size and the video payload size are counted. The **HiLPL** field specifies the three high bits. The **LastPacketLengthLo** field specifies the eight low bits.

EndOffset (5 bits): Indicates the FEC metadata packet distance from the last video data packet, minus one. For example, a video frame is fragmented into five video data packets. One FEC metadata packet is added after the five video data packets. These six packets are indexed starting at zero. The index of the first data packet is zero. The index of the last data packet is four. The index of the FEC metadata packet is 5. The **EndOffset** field is set to 5-4-1, or zero.

LastPacketLengthLo (8 bits): The **HiLPL** field and **LastPacketLengthLo** field specify the size (11 bits) of the last video data packet. Both the video payload header size and the video payload size are counted. The **HiLPL** field specifies the three high bits. The **LastPacketLengthLo** field specifies the eight low bits.

3 Protocol Details

3.1 Sender Details

This section covers the role of the sender of **RTVideo frames**.

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Send a RTVideo Frame

Whenever higher layers send an **RTVideo frame**, the video frame MUST be fragmented if it does not fit in one **video data packet** and the appropriate **RTP payload** format MUST be used to packetize the video fragments.

3.1.5 Message Processing Events and Sequencing Rules

The **RTVideo** sender specifies the packetization process. When the sender receives a video frame from higher layers, the RTVideo sender fragments the video frame into multiple data fragments and adds the **RTP payload** format header before each packet.

An RTP payload format MUST be selected to send a video frame, as specified in section 3.1.5.1.

3.1.5.1 Choice of RTP Payload Format

The **RTVideo** receiver might not be able to decode a video frame if the reference frame is missing, or the receiver might manage to decode the frame, but with undesirable video artifacts.

The RTVideo Extended RTP Payload Format (section 2.2.3) provides a way to reduce video artifacts by detecting and dropping the video frames for which reference frames are missing. This not only reduces the undesirable artifacts, it also reduces CPU usage. The RTVideo Extended RTP Payload Format SHOULD be used.

If protection against packet loss is desired, and the application can afford the cost of extra bandwidth, the RTVideo FEC RTP Payload Format (section 2.2.5) can be used together with the RTVideo Extended RTP Payload Format.

When minimizing resource use is desired or if reducing the video artifacts is desired, the RTVideo Basic RTP Payload Format (section 2.2.2) can be used.

The RTVideo Extended 2 RTP Payload Format (section 2.2.4) is reserved for future extensions. It MUST NOT be used.

3.1.5.2 Fragmenting Video Frames

3.1.5.2.1 Maximum Video Fragment Size

The video frame MUST be fragmented in such a way that the size of a fragment plus the overhead of all layers does not exceed the **maximum transmission unit (MTU)**.

The overhead MUST include at least the:

- **RTP payload** format header.
- **RTP** header.
- Transport protocol header. For example, UDP header (8 bytes) or TCP header (20 bytes).
- IP protocol header.

Upper protocol layers can have different limits on the payload size. To prevent payload size from exceeding any limit on any upper layer, the video fragment size MUST be smaller than 1200 bytes.

3.1.5.2.2 Additional Requirement for FEC

The RTVideo FEC RTP Payload Format (section 2.2.5) requires the video frame MUST be fragmented into packets of the same size with the exception of the last packet. The size of the last packet can be smaller in case the video frame cannot be evenly fragmented.

3.1.5.3 Understanding the Sequence Header

The **sequence header** carries enough information for the **RTVideo** receiver to decode a group of video frames. Because of this, this protocol requires that the sequence header MUST be present for the first **video data packet** of an RTVideo **I-frame**.

In addition to the semantics described in the SMPTE 421M standard [SMPTE-VC-1], the semantics of the **MAX_CODED_WIDTH** and **MAX_CODED_HEIGHT** fields in the sequence header is extended to represent the original picture aspect ratio of the video frames. Once a video frame has been received and decoded using the **CODED_WIDTH** and **CODED_HEIGHT** information in the **entry point header**, the display process in the receiving **endpoint** SHOULD<6> use the **MAX_CODED_WIDTH** and **MAX_CODED_HEIGHT** field values to reconstruct a video frame with horizontal and vertical dimensions agreeing with the original picture aspect ratio.

The sequence header present in the **Codec Header Bytes** field also contains the **DISP_HORIZ_SIZE** and **DISP_VERT_SIZE** fields. The receiving endpoint SHOULD<7> ignore these fields and continue to use the **MAX_CODED_WIDTH** and **MAX_CODED_HEIGHT** field values to determine the horizontal and vertical dimensions of the displayed video frame, as described in the preceding paragraph.

3.1.5.4 Forward Error Correction (FEC) Algorithm

The **FEC** implementation is capable of detecting and correcting errors in the video payload header, as well as the video payload data. It achieves this by constructing an FEC metadata packet(s), using the video payload header and video payload data, and sending it alongside video packets. Note that only one FEC packet SHOULD be constructed if the value of **DV** is "00".

Assuming that a video frame is fragmented and packetized into N data packets, the **RTP payload** header size plus the payload data size MUST be the same for all data packets, except for the last data packet.

Each **video data packet** is considered an FEC data block. The size of all the FEC data blocks MUST be the same. If the size of the video frame is not the multiple of the FEC data block size, zeros MUST be padded to the end to make the total size a multiple of the FEC data block size. The padding zeros are only used for calculating the FEC metadata packet and reconstructing the lost video data packet, and

are not sent over the network. The actual size of the last video data packet without padding might be smaller than the FEC data block size.

The actual size of the last video packet without padding is set in the **HiLPL** field and the **LastPacketLengthLo** field of the FEC RTP payload header. When the last video packet is reconstructed, the receiver MUST strip out the padding zeros using the **HiLPL** and the **LastPacketLengthLo** fields.

The FEC data block size MUST be smaller than the **MTU** size.

The process to calculate the FEC metadata packet of FEC Version 0 (**DV**="00")<8>.

1. For each video data packet and for each byte in the correspondent FEC data block, do a bitwise XOR with the correspondent byte in the FEC result buffer and save the result into the result buffer.
2. After completing the calculation for all the data packets, the FEC result buffer is the resulting FEC metadata.

The reconstruction of a lost video data packet is done by performing a byte-by-byte XOR operation on all the received video data packets and the FEC metadata packet. The result is the lost video data packet. If this data packet is the last data packet, only the first N bytes (N is specified by **HiLPL** field and **LastPacketLengthLo** field in the FEC RTP payload header) are actual data. The rest of the buffer is the padding and MUST be thrown away.

For FEC Version 1 (**DV** = "01"), the algorithm allows multiple FEC packets to protect the data packets of a video frame. All FEC packets are appended and transmitted immediately after the data packets pertaining to the video frame they protect. The first FEC packet is compatible with the default RTVideo FEC RTP Payload used when **DV** ="00". In this case, the FEC packet is a simple XOR packet of all the data packets pertaining to the video frame it protects, as described in the previous algorithm. The generation and use of subsequent FEC packets is dependent on the implementation on the clients.<9>

3.1.5.4.1 RTP Header Usage for FEC packets

The **FEC** metadata packet is also encapsulated into an **RTP packet**. The RTP packet MUST use the same numbering space as the rest of the **video data packets** for the **RTP** sequence number field. Refer to [RFC3550], section 5.1 for more details about the RTP sequence number field.

3.1.5.4.2 FEC Metadata Packet Usage

The **FEC** metadata packet MUST follow the **video data packets**.

3.1.5.5 SP-Frame and Cached Frame mechanisms

Both encoder and decoder can periodically cache decoded video frames. The cached video frame is stored in a dedicated memory location in addition to the current reference decoded I and P frames. The caching mechanism is done in a synchronized fashion, meaning that the video encoder and the video decoder MUST have a copy of the same decoded video frame in their cache whenever the encoder has finished encoding a video frame and the decoder is about to decode the same video frame.

The decoder on the receiver side MUST cache the **cached frame** because the next **SP-frame** references it.

The cache frame is signaled in the packetized video bitstream by means of the **C** field in the Packet Payload Format, as described in sections 2.2.2, 2.2.3 and 2.2.4. The encoder and decoder use only one cached frame at a time, so the data of the previous cached frame can be discarded whenever a new frame is cached.

When the encoder receives a packet loss event reported by the decoder, it can choose to encode the next **P-frame** using the cached frame as reference, as opposed to the previous P-frame or the

previous **I-frame**. In this case, the P-frame is called a Super P-frame (SP-frame) because it is predicted from the latest cached frame and not from the previous P-frame or the previous I-frame. The presence of an SP-frame in the packetized video bitstream is signaled by the **SP** field in the Packet Payload Format, as described in sections 2.2.2, 2.2.3 and 2.2.4. Upon receiving an SP-frame, the decoder decodes the video frame using the cached frame reference.

3.1.5.6 Other Requirements

The following rules MUST be followed when packetizing an **RTVideo frame**:

- The **RTP** sequence numbers MUST be continuous across video frames.
- The first packet of a video frame MUST have the **F** (first packet flag) bit set to "1".
- The last data packet of a video frame MUST have the **L** (last packet flag) bit set to "1".
- The last packet of a video frame (either the data packet or the **FEC** metadata packet) MUST have the RTP M bit set to "1".
- The **PacketNumber**, **HiPN**, **LastPacketLength**, **HiLPL**, **DV** and **EndOffset** fields for all FEC metadata packets (up to 31) MUST be set correctly. For each FEC packet, if the value of **DV** is "01", the **FECPacketsNumber** SHOULD<10> be set correctly.
- The **sequence header** MUST be present for the first packet of an **I-frame**. In other words, the **S** bit MUST be set to "1".
- The **HiFC**, **FrameCounter**, **HiRFC**, and **RefFrameCounter** fields MUST be set to the correct frame counter and the reference frame counter for the data packet and MUST be set to "0" for the FEC metadata packet.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Receiver Details

This section covers the role of the receiver of **RTVideo** packets.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Receive a Video Packet

Whenever a higher-layer component receives a video packet, the video packet MUST be parsed as specified in section 3.2.4.2. Validation SHOULD be done as specified in section 3.2.5.

If the video packet is the last data packet and all the data packets are received, the video frame can be constructed by concatenating all video payloads in the video packets.

If **video data packet** loss is detected and an **FEC** metadata packet is present, the lost data packet can be reconstructed by using the FEC algorithm.

If this video packet belongs to a new frame and the previous video frame cannot be constructed through the previous steps, all video packets of the previous video frame SHOULD be dropped.

3.2.4.2 Parsing RTVideo Packets

The M, M2, M3, E, and DV fields can be used to determine the **RTP payload** format type of a RTVideo packet.

Field values	RTP payload format type
M=0	Basic RTP Payload Format
M=1, M2=0	Extended RTP Payload Format
M=1, M2=1, E=0	Extended 2 RTP Payload Format (non **FEC**)
M=1, M2=1, M3=0, E=1, DV=00 or 01<11>	FEC RTP Payload Format

3.2.5 Message Processing Events and Sequencing Rules

When receiving each video packet, validation is done considering the following factors:

- The **RTP** sequence numbers MUST be continuous.
- The first packet of a video frame MUST have the **F** (first packet flag) bit set to "1".
- The last data packet MUST have the **L** (last packet flag) bit set to "1".
- The last packet of the frame (either the data packet or the **FEC** metadata packet) MUST have the RTP **M** bit set to "1".
- If FEC metadata packet(s) are present, the **PacketNumber**, **HiPN**, **LastPacketLength**, **HiLPL**, and **EndOffset** fields MUST be set correctly. For each FEC packet, if the value of **DV** is "01", the **FECPacketsNumber** MUST be set correctly.
- If the **I** bit is set to "1" (**I-frame**) and the **F** bit is set to "1" (the first packet), the codec headers MUST be present; in other words, the **S** bit MUST be set to "1".

If one or more **video data packets** for the video frame are not received and the lost video data packet cannot be reconstructed by FEC or FEC is not used, all the video packets of the video frame MUST be dropped.

If the RTVideo Extended RTP Payload Format (section 2.2.2) is used, the **HiFC**, **FrameCounter**, **HiRFC**, and **RefFrameCounter** fields can be used to reduce video artifact.

If the received video frame's encoded resolution is higher than the resolution negotiated as part of the capability negotiation phase of call setup, as specified in [MS-SDPEXT] section 3.1.5.24, this frame MUST NOT be forwarded to the receiver. The encoded resolution of the received video frame is available in the **entry point header** accompanying the I-frame's packets.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Basic RTP Payload Format Examples

4.1.1 I-Frame

The frame is packetized as four data packets.

4.1.1.1 First Packet

Payload header bytes in **network byte order**:

0x4F, 0x16, 0x25, 0x00, 0x00, 0x01, 0x0F, 0xC2 , 0x86, 0x0A, 0xF0, 0x8F, 0x88, 0x80, 0x00, 0x00, 0x01, 0x0E, 0x48, 0x04, 0x2B, 0xC2, 0x3C, 0x80

The payload header contains fields of the following values:

M=0, C=1, SP=0, L=0, O=1, I=1, S=1, F=1

Codec Headers Length=0x16

Codec Headers Bytes=0x25, 0x00 …

4.1.1.2 Second Packet

Payload header bytes in **network byte order**:

0x4C

The payload header contains fields of the following values:

M=0, C=1, SP=0, L=0, O=1, I=1, S=0, F=0

4.1.1.3 Last Packet

Payload header bytes in **network byte order**:

0x5C

The payload header contains fields of the following values:

M=0, C=1, SP=0, L=1, O=1, I=1, S=0, F=0

4.1.2 SP-Frame

The frame is packetized as 4 data packets.

4.1.2.1 First Packet

Payload header bytes in **network byte order**:

0x69

The payload header contains fields of the following values:

M=0, C=1, SP=1, L=0, O=1, I=0, S=0, F=1

4.1.2.2 Second Packet

Payload header bytes in **network byte order**:

0x68

The payload header contains fields of the following values:

M=0, C=1, SP=1, L=0, O=1, I=0, S=0, F=0

4.1.2.3 Last Packet

Payload header bytes in **network byte order**:

0x78

The payload header contains fields of the following values:

M=0, C=1, SP=1, L=1, O=1, I=0, S=0, F=0

4.1.3 P-Frame or B-Frame

The frame is packetized as a single data packet.

4.1.3.1 First Packet/LastPacket

Payload header bytes in **network byte order**:

0x19

The payload header contains fields of the following values:

M=0, C=0, SP=0, L=1, O=1, I=0, S=0, F=1

4.2 Extended RTP Payload Format Examples

4.2.1 I-Frame

The frame is packetized as three or more data packets.

4.2.1.1 First Packet

Payload header bytes in **network byte order**:

0xCF, 0x00, 0x00, 0x00, 0x16, 0x25, 0x00, 0x00, 0x01, 0x0F, 0xC2, 0x86, 0x0A, 0xF0, 0x8F, 0x88, 0x80, 0x00, 0x00, 0x01, 0x0E, 0x48, 0x04, 0x2B

The payload header contains fields of the following values:

M=1, C=1, SP=0, L=0, O=1, I=1, S=1, F=1; M2=0, HiFEC=0, HiFC=0, DV=0, E=0,

FrameCounter=0x0

RefFrameCounter=0x0

Codec Headers Length=0x16

Codec Header Bytes=0x27, 0x00 …

4.2.1.2 Second Packet

Payload header bytes in **network byte order**:

0xCC, 0x00, 0x00, 0x00

The payload header contains fields of the following values:

M=1, C=1, SP=0, L=0, O=1, I=1, S=0, F=0; M2=0, HiFEC=0, HiFC=0, DV=0, E=0,

FrameCounter=0x0

RefFrameCounter=0x0

4.2.1.3 Last Packet

Payload header bytes in **network byte order**:

0xDC, 0x00, 0x00, 0x00

The payload header contains fields of the following values:

M=1, C=1, SP=0, L=1, O=1, I=1, S=0, F=0; M2=0, HiFEC=0, HiFC=0, DV=0, E=0,

FrameCounter=0x0

RefFrameCounter=0x0

4.2.2 P-Frame

The frame is packetized as a single data packet.

4.2.2.1 First Packet/Last Packet

Payload header bytes in **network byte order**:

0x99, 0x00, 0x01, 0x00

The payload header contains fields of the following values:

M=1, C=0, SP=0, L=1, O=1, I=0, S=0, F=1; M2=0, HiFEC=0, HiFC=0, DV=0, E=0,

FrameCounter=0x1

RefFrameCounter=0x0

4.2.3 SP-Frame

The frame is packetized as three or more data packets.

4.2.3.1 First Packet

Payload header bytes in **network byte order**:

0xE9, 0x00, 0x0F, 0x00

The payload header contains fields of the following values:

M=1, C=1, SP=1, L=0, O=1, I=0, S=0, F=1; M2=0, HiFEC=0, HiFC=0, DV=0, E=0,

FrameCounter=0x0F

RefFrameCounter=0x0

4.2.3.2 Second Packet

Payload header bytes in **network byte order**:

0xE8, 0x00, 0x0F, 0x00

The payload header contains fields of the following values:

M=1, C=1, SP=1, L=0, O=1, I=0, S=0, F=0; M2=0, HiFEC=0, HiFC=0, DV=0, E=0,

FrameCounter=0x0F

RefFrameCounter=0x0

4.2.3.3 Last Packet

Payload header bytes in **network byte order**:

0xF8, 0x00, 0x0F, 0x00

The payload header contains fields of the following values:

M=1, C=1, SP=1, L=1, O=1, I=0, S=0, F=0; M2=0, HiFEC=0, HiFC=0, DV=0, E=0,

FrameCounter=0xF

RefFrameCounter=0x0

4.2.4 B-Frame

The frame is packetized as a single data packet.

4.2.4.1 First Packet/Last Packet

Payload header bytes in **network byte order**:

0x99, 0x00, 0x01, 0x11

The payload header contains fields of the following values:

M=1, C=0, SP=0, L=1, O=1, I=0, S=0, F=1; M2=0, HiFEC=0, HiFC=0, DV=0, E=0,

FrameCounter=0x1

RefFrameCounter=0x11, or RefFrameCounterDelta1=0x1, RefFrameCounterDelta2=0x1

RefFrameCounter1=0x0, RefFrameCounter2=0x0

4.3 FEC RTP Payload Format Examples

4.3.1 I-Frame

The frame is packetized as four data packets and one **FEC** metadata packet.

4.3.1.1 FEC Metadata Packet (FEC Version 0)

Payload header bytes in **network byte order**:

0xCC, 0x 81, 0x00, 0x00, 0x00, 0x04, 0x60, 0x84

The payload header contains fields of the following values:

M=1, C=1, SP=0, L=0, O=1, I=1, S=0, F=0; M2=1, HiFEC=0, HiFC=0, DV=0, E=1

FrameCounter=0x0

RefFrameCounter=0x0

M3=0, HiPN=0, Reserved=0

PacketNumberNo=0x4

HiLPL=3, EndOffset=0

LastPacketLengthLo=0x84

4.3.1.2 FEC Metadata Packet (FEC Version 1)

Payload header bytes in **network byte order**:

0xCC, 0x 83, 0x00, 0x00, 0x03, 0x04, 0x60, 0x84

The payload header contains fields of the following values:

M=1, C=1, SP=0, L=0, O=1, I=1, S=0, F=0; M2=1, HiFEC=0, HiFC=0, DV=1, E=1

FrameCounter=0x0

RefFrameCounter=0x0

M3=0, HiPN=0, FEC PacketsNumber = 3

PacketNumberNo=0x4

HiLPL=3, EndOffset=0

LastPacketLengthLo=0x84

4.3.2 SP-Frame

The frame is packetized as three data packets and one **FEC** metadata packet.

4.3.2.1 FEC Metadata Packet

Payload header bytes in **network byte order**:

0xE8, 0x81, 0x10, 0x00, 0x00, 0x03, 0x60, 0xDF

The payload header contains fields of the following values:

M=1, C=1, SP=1, L=0, O=1, I=0, S=0, F=0; M2=1, HiFEC=0, HiFC=0, DV=0, E=1

FrameCounter=0x10

RefFrameCounter=0x0

M3=0, HiPN=0, Reserved=0

PacketNumberNo=0x3

HiLPL=3, EndOffset=0

LastPacketLengthLo=0xDF

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- Microsoft Office Communications Server 2007
- Microsoft Office Communications Server 2007 R2
- Microsoft Office Communicator 2007
- Microsoft Office Communicator 2007 R2
- Microsoft Lync Server 2010
- Microsoft Lync 2010
- Microsoft Lync Server 2013
- Microsoft Lync Client 2013/Skype for Business
- Microsoft Skype for Business 2016
- Microsoft Skype for Business Server 2015
- Microsoft Skype for Business 2019 Preview
- Microsoft Skype for Business Server 2019 Preview

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

<1> Section 2.2.1: Office Communications Server 2007, Office Communications Server 2007 R2: This behavior is not supported. If a server does not receive a video packet, it drops all packets of the sequence of packets representing the entire video frame.

<2> Section 2.2.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported. The **entry point header** is only present in the Codec Headers Bytes field and not in the video payload.

<3> Section 2.2.5: Office Communications Server 2007, Office Communicator 2007: **Reserved/FECPacketsNumber** (5 bits): Reserved field for future use. It is set to zero. The sender does not send more than one **FEC** metadata packets for each **RTVideo frame**.

<4> Section 2.2.5: Office Communications Server 2007, Office Communicator 2007: **DV** (2 bits): FEC version number. The value ranges from zero (0) to 3. The version number is zero (0) in the RTVideo FEC RTP Payload Format (section 2.2.5).

<5> Section 2.2.5: Office Communications Server 2007, Office Communicator 2007: **Reserved/FECPacketsNumber** (5 bits): Reserved field for future use. It is set to zero. The sender does not send more than one FEC metadata packets for each RTVideo frame

<6> Section 3.1.5.3: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

Receivers do not reconstruct a video frame with the aspect ratio of the **MAX_CODED_WIDTH** and **MAX_CODED_HEIGHT** specified in the **sequence header**.

<7> Section 3.1.5.3: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported. The sequence header in the **Codec Header Bytes** field does not contain the **DISP_HORIZ_SIZE** and **DISP_VERT_SIZE** fields.

<8> Section 3.1.5.4: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported. For all other products, the size of the buffer is the FEC data block size.

<9> Section 3.1.5.4: Office Communications Server 2007, Office Communicator 2007: **DV** (2 bits): FEC version number. The value ranges from zero (0) to 3. The version number is zero (0) in the RTVideo FEC RTP Payload Format (section 2.2.5). **Reserved/FECPacketsNumber** (5 bits): Reserved field for future use. It is set to zero. The sender does not send more than one FEC metadata packet for each RTVideo frame.

<10> Section 3.1.5.6: Office Communications Server 2007, Office Communicator 2007:

The value of the **DV** field is zero ("0") and the value of the Reserved/**FECPacketsNumber** field is set to zero ("0"). The sender does not send more than one FEC metadata packets for each RTVideo frame.

<11> Section 3.2.4.2: Office Communications Server 2007, Office Communicator 2007: **DV** (2 bits): FEC version number. The value ranges from zero (0) to 3. The version number is zero (0) in the RTVideo FEC RTP Payload Format (section 2.2.5).

7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
6 Appendix A: Product Behavior	Updated list of supported products.	major

8 Index