International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.271
(05/2006)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Coding of moving video

---

Video back-channel messages for conveyance of status information and requests from a video receiver to a video sender

ITU-T Recommendation H.271

ITU-T

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	**H.260–H.279**
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.349
Directory services architecture for audiovisual and multimedia services	H.350–H.359
Quality of service architecture for audiovisual and multimedia services	H.360–H.369
Supplementary services for multimedia	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
Mobility interworking procedures	H.550–H.559
Mobile multimedia collaboration inter-working procedures	H.560–H.569
BROADBAND AND TRIPLE-PLAY MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation H.271

Video back-channel messages for conveyance of status information and requests from a video receiver to a video sender

Summary

This Recommendation specifies the format of back-channel messages for conveyance of status information and requests from a video receiver to a video sender.

The message syntax has been designed in a generic way to make it applicable for use with the majority of the existing international video coding standards. The application of the generic messages to ITU-T Recs H.261, H.263, and H.264 | ISO/IEC 14496-10 is specified.

Source

ITU-T Recommendation H.271 was approved on 29 May 2006 by ITU-T Study Group 16 (2005-2008) under the ITU-T Recommendation A.8 procedure.

Keywords

Back-channel message, receiver feedback, reference picture selection, video.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Introduction

0.1 Purpose

For some applications, the transmission of additional data (in-band or out-of-band) is useful for improving video quality of service. This Recommendation specifies data payloads for use with a variety of video coding technologies. The data payloads are specified in a generic way to make them applicable to prevalent existing video coding standards. The application of the generic messages to ITU-T Recs H.261, H.263, and H.264 is specified.

0.2 Overview

The following information can be signalled from a video receiver to a video sender using the back-channel message(s) defined in this Recommendation:

- Status reports:
  - One or more pictures that are without detected bitstream error mismatch;
  - Picture-level and/or macroblock-level losses;
  - Information of important header information.
- Update requests:
  - A "reset" request indicating that the sender should completely refresh the video bitstream as if no prior bitstream data had been received.

The application of the back-channel messages to ITU-T Recs H.261, H.263, and H.264 is specified.

ITU-T Recommendation H.271

Video back-channel messages for conveyance of status information and requests from a video receiver to a video sender

1 Scope

This Recommendation specifies back-channel messages for block-based video coding.

2 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

– ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at p × 64 kbit/s*.

– ITU-T Recommendation H.263 (2005), *Video coding for low bit rate communication*.

– ITU-T Recommendation H.264 (2005), *Advanced video coding for generic audiovisual services*.

  ISO/IEC 14496-10:2005, *Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding*.

3 Definitions

This Recommendation defines the following terms:

3.1 back channel: A means to convey back-channel messages from a receiver of a video bitstream to a sender of a video bitstream.

3.2 back-channel message: A message that is generated by a receiver of a video *bitstream* that conveys receiver status information or requests.

3.3 bitstream: A sequence of bits that forms the representation of coded *pictures* and associated data forming one or more coded video sequences.

3.4 bitstream error: A corrupted or incomplete *bitstream*.

3.5 bitstream error mismatch: A difference between the values or quantity of decoded *pictures* caused by one or more *bitstream errors* versus the values or quantity of decoded *pictures* generated by the decoding process with a *bitstream* without *bitstream errors*.

3.6 block: An MxN (M-column by N-row) array of luma samples and the associated chroma samples.

3.7 detected bitstream error: A *bitstream error* that is detected by the receiver.

3.8 detected bitstream error mismatch: A *bitstream error mismatch* that may be present because of *bitstream errors*.

3.9 decoding order: The order in which *syntax elements* are processed by the decoding process specified by a video coding technology.

3.10 macroblock: A 16x16 *block* of luma samples and two corresponding blocks of chroma samples.

3.11 parameter set: A syntax structure that contains a number of *syntax elements* that may be used in the decoding process for one or more *pictures*.

3.12 picture: A collective term for a field or a frame of video that is coded as a distinct unit by a video coding technology.

3.13 reference picture: A *picture* that contains samples that may be used for inter-picture prediction in the decoding process of subsequent *pictures* in the video bitstream.

3.14 reserved: The term reserved, when used in the clauses specifying some values of a particular syntax element, are for future use by ITU-T. These values shall not be used in back-channel messages conforming to this Recommendation, but may be used in future extensions of this Recommendation by ITU-T.

3.15 syntax element: An element of data represented in the *bitstream* or in a back-channel message.

4 Abbreviations

This Recommendation uses the following abbreviations:

CRC Cyclic Redundancy Code

LSB Least Significant Bit

MSB Most Significant Bit

5 Conventions

Throughout this Recommendation, statements appearing with the preamble "NOTE –" are informative and are not an integral part of this Recommendation.

NOTE – The mathematical operators used in this Recommendation are similar to those used in the C programming language. Numbering and counting conventions generally begin from 0.

5.1 Arithmetic operators

The following arithmetic operators are defined as follows.

+ Addition

– Subtraction (as a two-argument operator) or negation (as a unary prefix operator)

* Multiplication

/ Integer division with truncation of the result toward zero. For example, 7/4 and (–7) / (–4) are truncated to 1 and (–7) / 4 and 7 / (–4) are truncated to –1.

x % y Modulus. Remainder of x divided by y, defined only for integers x and y with $x \geq 0$ and $y > 0$.

When order of precedence is not indicated explicitly by use of parenthesis, the following rules apply:

– multiplication and division operations are considered to take place before addition and subtraction;

– multiplication and division operations in sequence are evaluated sequentially from left to right;

– addition and subtraction operations in sequence are evaluated sequentially from left to right.

5.2 Logical operators

The following logical operators are defined as follows:

x && y Boolean logical "and" of x and y

x || y Boolean logical "or" of x and y

! Boolean logical "not"

x ? y : z If x is TRUE or not equal to 0, evaluates to the value of y; otherwise, evaluates to the value of z

5.3 Relational operators

The following relational operators are defined as follows:

> Greater than

>= Greater than or equal to

< Less than

<= Less than or equal to

== Equal to

!= Not equal to

5.4 Bit-wise operators

The following bit-wise operators are defined as follows:

& Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

| Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

^ Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

x >> y Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for positive integer values of y. Bits shifted into the MSBs as a result of the right shift have a value equal to the MSB of x prior to the shift operation.

x << y Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for positive integer values of y. Bits shifted into the LSBs as a result of the left shift have a value equal to 0.

5.5 Assignment operators

The following arithmetic operators are defined as follows:

= Assignment operator.

++ Increment, i.e., x++ is equivalent to x = x + 1, when used in an array index, evaluates to the value of the variable prior to the increment operation.

– – Decrement, i.e., x– – is equivalent to x = x – 1; when used in an array index, evaluates to the value of the variable prior to the decrement operation.

+= Increment by amount specified, i.e., x += 3 is equivalent to x = x + 3, and x += (–3) is equivalent to x = x + (–3).

– = Decrement by amount specified, i.e., x – = 3 is equivalent to x = x – 3, and x – = (–3) is equivalent to x = x – (–3).

5.6 Variables, syntax elements, and tables

Syntax elements in the bitstream are represented in **bold** type. Each syntax element is described by its name (all lower case letters with underscore characters), its one or two syntax categories, and one or two descriptors for its method of coded representation. The decoding process behaves according to the value of the syntax element and to the values of previously decoded syntax elements. When a value of a syntax element is used in the syntax tables or the text, it appears in regular (i.e., not bold) type.

In some cases, the syntax tables may use the values of other variables derived from syntax elements values. Such variables appear in the syntax tables, or text, named by a mixture of lower case and upper case letters and without any underscore characters. Variables starting with an upper case letter are derived for the decoding of the current syntax structure and all depending syntax structures. Variables starting with an upper case letter may be used in the decoding process for later syntax structures mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the subclause in which they are derived.

In some cases, "mnemonic" names for syntax element values or variable values are used interchangeably with their numerical values. Sometimes "mnemonic" names are used without any associated numerical values. The association of values and names is specified in the text. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an upper case letter and may contain more upper case letters.

NOTE – The syntax is described in a manner that closely follows the C-language syntactic constructs.

Functions are described by their names, which are constructed as syntax element names, with left and right round parentheses including zero or more variable names (for definition) or values (for usage), separated by commas (if more than one variable).

Binary notation is indicated by enclosing the string of bit values by single quote marks. For example, '01000001' represents an eight-bit string having only its second and its last bits equal to 1.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", may be used instead of binary notation when the number of bits is an integer multiple of 4. For example, 0x41 represents an eight-bit string having only its second and its last bits equal to 1.

Numerical values not enclosed in single quotes and not prefixed by "0x" are decimal values.

A value equal to 0 represents a FALSE condition in a test statement. The value TRUE is represented by any other value different than zero.

5.7 Text description of logical operations

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0 )
  statement 0
else if ( condition 1 )
  statement 1
…
else /* informative remark on remaining condition */
  statement n
```

may be described in the following manner:

... as follows / ... the following applies.

– If condition 0, statement 0

– Otherwise, if condition 1, statement 1

– …

– Otherwise (informative remark on remaining condition), statement n

Each "If...Otherwise, if...Otherwise, ..." statement in the text is introduced with "... as follows" or "... the following applies" immediately followed by "If ... ". The last condition of the "If...Otherwise, if...Otherwise, ..." is always an "Otherwise, ...". Interleaved "If...Otherwise, if...Otherwise, ..." statements can be identified by matching "... as follows" or "... the following applies" with the ending "Otherwise, ...".

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0a  &&  condition 0b )
  statement 0
else if ( condition 1a  ||  condition 1b )
  statement 1
…
else
  statement n
```

may be described in the following manner:

... as follows / ... the following applies.

– If all of the following conditions are true, statement 0
  - condition 0a
  - condition 0b

– Otherwise, if any of the following conditions are true, statement 1
  - condition 1a
  - condition 1b

– …

– Otherwise, statement n

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0 )
  statement 0
if ( condition 1 )
  statement 1
```

may be described in the following manner:

When condition 0, statement 0

When condition 1, statement 1

5.8 Method of describing syntax in tabular form

The following table lists examples of pseudo code used to describe the syntax. When **syntax_element** appears, it specifies that a syntax element is parsed from the bitstream and the bitstream pointer is advanced to the next position beyond the syntax element in the bitstream parsing process.

	Descriptor
/* A statement can be a syntax element with an associated syntax category and descriptor or can be an expression used to specify conditions for the existence, type, and quantity of syntax elements, as in the following two examples */	
syntax_element	ue(v)
conditioning statement or structure {	
/* A group of statements enclosed in curly brackets is a compound statement and is treated functionally as a single statement. */	
statement	
statement	
…	
}	
/* A "while" structure is a conditioning statement that specifies a test of whether a condition is true, and if true, specifies evaluation of a statement (or compound statement) repeatedly until the condition is no longer true */	
while(condition)	
statement	
/* A "do … while" structure is a conditioning structure that specifies evaluation of a statement once, followed by a test of whether a condition is true, and if true, specifies repeated evaluation of the statement until the condition is no longer true */	
do	
statement	
while(condition)	
/* An "if … else" structure specifies a test of whether a condition is true, and if the condition is true, specifies evaluation of a primary statement, otherwise, specifies evaluation of an alternative statement. The "else" part of the structure and the associated alternative statement is omitted if no alternative statement evaluation is needed */	
if(condition)	
primary statement	
else	
alternative statement	
/* A "for" structure specifies evaluation of an initial statement, followed by a test of a condition, and if the condition is true, specifies repeated evaluation of a primary statement followed by a subsequent statement until the condition is no longer true. */	
for(initial statement; condition; subsequent statement)	
primary statement	

5.9 Specification of syntax functions, categories, and descriptors

The functions presented here are used in the syntactical description. These functions assume the existence of a bitstream pointer with an indication of the position of the next bit to be read by the decoding process from the bitstream.

byte_aligned( ) is specified as follows.

– If the current position in the bitstream is on a byte boundary, i.e., the next bit in the bitstream is the first bit in a byte, the return value of byte_aligned( ) is equal to TRUE.
– Otherwise, the return value of byte_aligned( ) is equal to FALSE.

more_msg_data( ) is specified as follows.

– If there is more data in the syntax structure msg_data( ), the return value of more_msg_data( ) is equal to TRUE.
– Otherwise, the return value of more_msg_data( ) is equal to FALSE.

The method for enabling determination of whether there is more data in the syntax structure msg_data( ) is specified by the application.

next_bits(n) provides the next bits in the bitstream for comparison purposes, without advancing the bitstream pointer. Provides a look at the next n bits in the bitstream with n being its argument.

read_bits(n) reads the next n bits from the bitstream and advances the bitstream pointer by n bit positions. When n is equal to 0, read_bits(n) is specified to return a value equal to 0 and to not advance the bitstream pointer.

The following descriptors specify the parsing process of each syntax element.

– f(n): fixed-pattern bit string using n bits written (from left to right) with the left bit first. The parsing process for this descriptor is specified by the return value of the function read_bits(n).
– u(n): unsigned integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function read_bits(n) interpreted as a binary representation of an unsigned integer with most significant bit written first.
– ue(v): unsigned integer Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in the following.

Syntax elements coded as ue(v) are Exp-Golomb-coded. The parsing process for these syntax elements begins with reading the bits starting at the current location in the bitstream up to and including the first non-zero bit, and counting the number of leading bits that are equal to 0. This process shall be equivalent to the following:

```
leadingZeroBits = –1;
for( b = 0; !b; leadingZeroBits++ )
    b = read_bits( 1 )
```

The variable codeNum is then assigned as follows:

$$\text{codeNum} = 2^{\text{leadingZeroBits}} - 1 + \text{read\_bits(leadingZeroBits)}$$

where the value returned from read_bits(leadingZeroBits) is interpreted as a binary representation of an unsigned integer with most significant bit written first.

The value of the syntax element is equal to codeNum.

6 Message payloads

6.1 Syntax

msg_data( ) {	**Descriptor**
do	
message( )	
while(more_msg_data( ))	
}	

message( ) {	**Descriptor**
payloadType = 0	
while(next_bits(8) = = 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadType += 255	
}	
last_payload_type_byte	u(8)
payloadType += last_payload_type_byte	
payloadSize = 0	
while(next_bits(8) = = 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadSize += 255	
}	
last_payload_size_byte	u(8)
payloadSize += last_payload_size_byte	
msg_payload(payloadType, payloadSize)	
}	

msg_payload(payloadType, payloadSize) {	**Descriptor**
if(payloadType <= 4)	
ref_pic_id	u(32)
if(payloadType = = 0) {	
num_ref_pics_minus1	ue(v)
for(i = 1; i <= num_ref_pics_minus1; i++)	
good_ref_pic_id[i]	u(32)
} else if(payloadType = = 1)	
delta_ref_pic_id	ue(v)
else if(payloadType = = 2) {	
data_partition_idc	ue(v)
run_length_flag	u(1)
if(run_length_flag) {	
first_blk_lost	ue(v)

num_blks_lost_minus1	ue(v)
} else {	
top_left_blk	ue(v)
bottom_right_blk	ue(v)
}	
} else if((payloadType == 3) || (payloadType == 4)) {	
param_set_type	ue(v)
param_set_crc	u(16)
if(payloadType == 3)	
param_set_id	ue(v)
}	
stop_one_bit /* equal to 1 */	f(1)
while(!byte_aligned( ))	
alignment_zero_bit /* equal to 0 */	f(1)
}	

6.2 Semantics

ff_byte shall be equal to 0xFF.

last_payload_type_byte is the last byte used to specify the payload type of a message.

last_payload_size_byte is the last byte used to specify the size of a message.

The size of the syntax structure msg_payload( ) shall be equal to payloadSize * 8 bits.

Corresponding to the values of payloadType, the message types are as shown in Table 6-1.

Table 6-1/H.271 – Message types

payloadType	Message
0	One or more pictures without detected bitstream error mismatch
1	One or more pictures that are entirely or partially lost
2	A set of blocks of one picture that is entirely or partially lost
3	CRC for one parameter set
4	CRC for all parameter sets of a certain type
5	A "reset" request indicating that the sender should completely refresh the video bitstream as if no prior bitstream data had been received
> 5	Reserved for future use by ITU-T

Messages having payloadType greater than 5 shall be removed and discarded according to the size of the message as indicated by payloadSize.

ref_pic_id identifies a picture. Depending on the value of payloadType, the semantics of ref_pic_id are as specified in Table 6-2.

Table 6-2/H.271 – Semantics of ref_pic_id

payloadType	Semantics of ref_pic_id
0	Specifies a picture without detected bitstream error mismatch.
1	Specifies a picture that has been entirely or partially lost.
2	Specifies a picture that has been partially lost.
3	Specifies a picture. A CRC is present for one parameter set associated with the specified picture.
4	Specifies a picture. One CRC is present for all parameter sets of a certain type that are stored at the time associated with the decoding of the specified picture.

num_ref_pics_minus1 plus 1 specifies the number of the pictures that are without detected bitstream error mismatch. The value of num_ref_pics_minus1 shall be in the range of 0 to 31, inclusive.

good_ref_pic_id[i] specifies the i^{th} picture indicated in the current message that is without detected bitstream error mismatch.

delta_ref_pic_id specifies an increment relative to ref_pic_id identifying a set of pictures that are entirely or partially lost. The value of delta_ref_pic_id shall be in the range of 0 to 31, inclusive.

data_partition_idc indicates that all data (when data_partition_idc is equal to 0) or a partition of the data (when data_partition_idc is not equal to 0) for the blocks specified by top_left_blk and bottom_right_blk is lost. The value of data_partition_idc shall be in the range of 0 to 15, inclusive.

run_length_flag equal to 1 indicates the presence of the syntax elements first_blk_lost and num_blks_lost_minus1. run_length_flag equal to 0 indicates the presence of the syntax elements top_left_blk and bottom_right_blk.

first_blk_lost indicates the block address of the first block in raster scan order. A block address is the index of a block in a block raster scan of the picture starting with zero for the top-left block in a picture.

num_blks_lost_minus1 plus 1 specifies the number of consecutive blocks in raster scan order with the first block being identified by first_blk_lost.

top_left_blk and **bottom_right_blk** specify the block addresses of the top-left and bottom-right blocks, respectively, of the rectangular region that is entirely or partially lost. The following constraints shall be obeyed by the values of the syntax elements top_left_blk and bottom_right_blk, where the variable PicWidthInBlks is the picture width in units of blocks.

– top_left_blk shall be less than or equal to bottom_right_blk.

– bottom_right_blk shall be less than the picture size in units of blocks.

– top_left_blk % PicWidthInBlks shall be less than or equal to bottom_right_blk % PicWidthInBlks.

param_set_type specifies the type of the parameter set(s). The value of param_set_type shall be in the range of 0 to 15, inclusive.

param_set_crc is the CRC for the parameter set identified by param_set_id and param_set_type when payloadType is equal to 3 and is the CRC for all the parameter sets of the type specified by param_set_type together when payloadType is equal to 4.

The value of param_set_crc shall be equal to the value of crcVal obtained by performing the following pseudo-code process.

```
paramSet[pLen     ] = 0
paramSet[pLen + 1] = 0
crcVal = 0xFFFF
for( bitIdx = 0; bitIdx < ( pLen + 2 ) * 8; bitIdx++ ) {
   crcMsb = ( crcVal >> 15 ) & 1
   bitVal = ( paramSet[bitIdx >> 3] >> ( 7 – ( bitIdx & 7 ) ) ) & 1
   crcVal = ( ( ( crcVal << 1 ) + bitVal ) & 0xFFFF ) ^ ( crcMsb * 0x1021 )
}                                                                     (6-1)
```

where the variable paramSet is a string of bytes containing, at the beginning of the string of bytes, the data for which the CRC is computed; the variable pLen is the number of bytes of the data for which the CRC is computed; and the string of bytes of the variable paramSet is of sufficient length for two additional zero-valued bytes to be appended to the end of the data for which the CRC is computed.

When payloadType is equal to 3, paramSet contains, in network byte order, the bytes of the received parameter set identified by param_set_id and param_set_type.

When payloadType is equal to 4, paramSet contains a concatenation, in network byte order, of the bytes of all parameter sets of the type identified by param_set_type, in increasing order of the parameter set identifier. For any parameter set that has never been received, the representation of the parameter set shall be considered to be two bytes long, with the first byte containing the eight MSBs and the second byte containing the eight LSBs of an unsigned 16-bit binary representation of the value of the parameter set identifier.

param_set_id specifies the parameter set identifier of the parameter set of the type specified by param_set_type for which a CRC is present. The value of param_set_id shall be in the range of 0 to 65535, inclusive.

stop_one_bit shall be equal to 1.

alignment_zero_bit shall be equal to 0.

7 Standard specific uses of the messages

In this clause, bit n of a variable refers to the n^{th} least significant bit of the binary representation of the variable. The LSB is counted as bit 0.

7.1 H.261 specific use of the messages

The value of payloadType shall be equal to 0, 1, 2, or 5. Messages with other values of payloadType shall be ignored.

When ref_pic_id or good_ref_pic_id[i] are present, the 5 LSBs of these syntax elements indicate the value of TR, which is equal to ref_pic_id & 0x1F or good_ref_pic_id[i] & 0x1F, of a picture. The remaining bits of ref_pic_id or good_ref_pic_id[i] are reserved for future use by ITU-T, shall be equal to 0, and shall be ignored.

Depending on the value of payloadType, the following applies.

– If payloadType is equal to 0, the picture having TR equal to ref_pic_id & 0x1F or good_ref_pic_id[i] & 0x1F is without detected bitstream error mismatch.
– Otherwise, if payloadType is equal to 1, all pictures, in decoding order, starting with a picture having TR equal to ref_pic_id & 0x1F, up to and including a picture having TR equal to ((ref_pic_id & 0x1F) + delta_ref_pic_id) % 32, when such pictures were present in the video bitstream, are indicated to have been entirely or partially lost.
– Otherwise, if payloadType is equal to 2, the picture having TR equal to ref_pic_id & 0x1F has been partially lost. Each block is a macroblock. The value of data_partition_idc shall be equal to 0, which indicates that all data of the identified macroblocks is considered lost. All other values of data_partition_idc are reserved for future use by ITU-T and shall be ignored.
– Otherwise (payloadType is equal to 5), no further specification.

7.2 H.263 specific use of the messages

The H.263 features that are supported by the present syntax include the following:

– Long term pictures;
– Data partitioning.

The following H.263 features are not supported by the present syntax:

– Interlaced fields.

The value of payloadType shall be equal to 0, 1, 2, or 5. Messages with other values of payloadType shall be ignored.

When ref_pic_id or good_ref_pic_id[i] is present, the 12 LSBs indicate the value of the variable picIdentifier, which is equal to ref_pic_id & 0x0FFF or good_ref_pic_id[i] & 0x0FFF. When Annex U/H.263 is in use and payloadType is equal to 0, bit 12 indicates whether the stored picture is a long-term picture (when the bit is equal to 1) or a short-term picture (when the bit is equal to 0). Bit 13 shall be equal to 0 unless the optional Temporal, SNR and Spatial Scalability mode (Annex O/H.263) is used. When bit 13 is equal to 1, the message refers to an enhancement layer (rather than the base layer), and bits 14 to 17, inclusive, contain the value of ELNUM, which is equal to ref_pic_id & 0x03C000 or good_ref_pic_id[i] & 0x03C000. The remaining bits of ref_pic_id or good_ref_pic_id[i] are reserved for future use by ITU-T, shall be equal to 0, and shall be ignored.

– If Annex U/H.263 is not in use, picIdentifier indicates the value of TR of a picture. The value of TR shall be less than MaxTR that is equal to the maximum possible value of TR plus 1. Depending on the value of payloadType, the following applies.
  • If payloadType is equal to 0, bit 12 of ref_pic_id or good_ref_pic_id[i] shall be equal to 0. The picture having TR equal to picIdentifier, and, when bit 13 of ref_pic_id or good_ref_pic_id[i] is equal to 1, having ELNUM equal to ref_pic_id & 0x03C000 or good_ref_pic_id[i] & 0x03C000, is without detected bitstream error mismatch.
  • Otherwise, if payloadType is equal to 1, bit 12 of ref_pic_id shall be equal to 0. All pictures, in decoding order and in the base layer or the same enhancement layer with ELNUM equal to ref_pic_id & 0x03C000, starting with a picture having TR equal to ref_pic_id & 0x0FFF, up to and including a picture having TR equal to ((ref_pic_id & 0x0FFF) + delta_ref_pic_id) % MaxTR, when such pictures were present in the video bitstream, are indicated to have been entirely or partially lost.

- Otherwise, if payloadType is equal to 2, bit 12 of ref_pic_id shall be equal to 0. The picture having TR equal to picIdentifier and, when bit 13 of ref_pic_id is equal to 1, having ELNUM equal to ref_pic_id & 0x03C000, is partially lost. The value of data_partition_idc indicates that all data (when data_partition_idc is equal to 0), header data partition (when data_partition_idc is equal to 1), motion vector partition (when data_partition_idc is equal to 2) or coefficient data partition (when data_partition_idc is equal to 3) of the identified macroblocks is lost. All other values of data_partition_idc are reserved for future use by ITU-T and shall be ignored.
- Otherwise (payloadType is equal to 5), no further specification.

– Otherwise (Annex U/H.263 is in use), depending on the value of payload type, the following applies.

- If payloadType is equal to 0, ref_pic_id or good_ref_pic_id[i] identifies a stored picture without detected bitstream error mismatch. The variable picIdentifier indicates the value of the stored picture's PN (when the stored picture is a short-term picture) or LPIN (when the stored picture is a long-term picture).
  - If the picture is a short-term picture, the value of picIdentifier shall be less than MaxPN, which is the maximum possible value of PN plus 1.
  - Otherwise (the picture is a long-term picture), the value of picIdentifier shall be less than MaxLPIN, which is the maximum possible value of LPIN plus 1.

– Otherwise, if payloadType is equal to 1, bit 12 of ref_pic_id shall be equal to 0, and picIdentifier indicates the value of PN of a picture. The value of PN shall be less than MaxPN that is equal to the maximum possible value of PN plus 1. All pictures, in decoding order and in the same enhancement or base layer, starting with a picture having PN equal to ref_pic_id & 0x0FFF, up to and including a picture having PN equal to ((ref_pic_id & 0x0FFF) + delta_ref_pic_id) % MaxPN, when such pictures were present in the video bitstream, are indicated to have been entirely or partially lost. MaxPN is equal to the maximum possible value of PN plus 1.

– Otherwise, if payloadType is equal to 2, bit 12 of ref_pic_id shall be equal to 0, and picIdentifier indicates the value of PN of a picture. The value of PN shall be less than MaxPN that is equal to the maximum possible value of PN plus 1. The picture having TR equal to picIdentifier and, when bit 13 of ref_pic_id is equal to 1, having ELNUM equal to ref_pic_id & 0x03C000, is partially lost. Each block is a macroblock. The value of data_partition_idc indicates that all data (when data_partition_idc is equal to 0), header data partition (when data_partition_idc is equal to 1), motion vector partition (when data_partition_idc is equal to 2) or coefficient data partition (when data_partition_idc is equal to 3) of the identified macroblocks is lost. All other values of data_partition_idc are reserved for future use by ITU-T and shall be ignored.

– Otherwise (payloadType is equal to 5), no further specification.

7.3 H.264 specific use of the messages

The H.264 features that are supported by the present syntax include the following:

– Long-term reference pictures;

– Data partitioning;

– Sequence and picture parameter set.

The following H.264 features are not supported by the present syntax:

– Frames coded using macroblock-adaptive frame-field coding;

– Field pictures.

When ref_pic_id or good_ref_pic_id[i] is present, the following applies. The 16 LSBs indicate the value of the variable picIdentifier, which is equal to ref_pic_id & 0xFFFF or good_ref_pic_id[i] & 0xFFFF. When payloadType is equal to 0, bit 16 indicates whether the reference picture is a long-term reference picture (when the bit is equal to 1) or a short-term reference picture (when the bit is equal to 0). The remaining bits of ref_pic_id or good_ref_pic_id[i] are reserved for future use by ITU-T, shall be equal to 0, and shall be ignored.

Depending on the value of payloadType, the following applies.

– If payloadType is equal to 0, ref_pic_id or good_ref_pic_id[i] identifies a reference picture without detected bitstream error mismatch. The variable picIdentifier indicates the value of the reference picture's FrameNum (when the reference picture is a short-term reference picture) or LongTermFrameIdx (when the reference picture is a long-term reference picture).
  - If the picture is a short-term reference picture, the value of picIdentifier shall be less than MaxFrameNum.
  - Otherwise (the picture is a long-term reference picture), the value of picIdentifier shall not be greater than MaxLongTermFrameIdx.
– Otherwise, if payloadType is equal to 1, bit 16 shall be equal to 0, and the variable picIdentifier indicates the value of FrameNum of a reference picture. The value of FrameNum shall be less than MaxFrameNum. All pictures, in decoding order, starting with a picture having FrameNum equal to ref_pic_id & 0xFFFF, up to and including a picture having FrameNum equal to ((ref_pic_id & 0xFFFF) + delta_ref_pic_id) % MaxFrameNum, when such pictures were present in the video bitstream, are indicated to have been entirely or partially lost.
– Otherwise, if payloadType is equal to 2, bit 16 shall be equal to 0, and the variable picIdentifier indicates the value of FrameNum of a reference picture that is partially lost. The value of FrameNum shall be less than MaxFrameNum. Each block is a macroblock. The value of data_partition_idc indicates that all data (when data_partition_idc is equal to 0), data partition A (when data_partition_idc is equal to 1), data partition B (when data_partition_idc is equal to 2) or data partition C (when data_partition_idc is equal to 3) of the identified macroblocks is lost. All other values of data_partition_idc are reserved for future use by ITU-T and shall be ignored.
– Otherwise, if payloadType is equal to 3 or 4, the 16 LSBs of ref_pic_id indicate the FrameNum which is equal to ref_pic_id & 0xFFFF, of a reference picture for which a CRC is present for one or all the associated sequence parameter sets or picture parameter sets. The value of ref_pic_id & 0xFFFF shall be less than MaxFrameNum. The remaining bits of ref_pic_id are reserved for future use by ITU-T, shall be equal to 0, and shall be ignored. A parameter set with param_set_type equal to 0 is a sequence parameter set NAL unit. A parameter set with param_set_type equal to 1 is a picture parameter set NAL unit. When used in calculation of a CRC, the forbidden_zero_bit and nal_ref_idc of a sequence or picture parameter set NAL unit are set equal to 0 and 3, respectively.
– Otherwise (payloadType is equal to 5), no further specification.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2006