

### Contents

	Foreword
	Introduction
1	Scope
2	Normative references
3	Terms and definitions
4	Abbreviated terms
5	Conventions
5.1	General
5.2	Arithmetic operators
5.3	Logical operators
5.4	Relational operators
5.5	Bit-wise operators
5.6	Assignment operators
5.7	Range notation
5.8	Mathematical functions
5.9	Order of operation precedence
5.10	Variables, syntax elements and tables
5.11	Text description of logical operators
5.12	Processes
6	Syntax and semantics
6.1	Method of specifying syntax in tabular form
6.2	Bit ordering
6.3	Specification of syntax functions and data types
6.4	Semantics
7	Data structures
7.1	General
7.2	Data unit
7.3	Raw reference
7.3.1	General
7.3.2	Syntax and semantics
7.4	Parameter set
7.4.1	Syntax and semantics
7.4.2	Encoding parameters
7.4.2.1	General
7.4.2.2	Descriptor configuration syntax and semantics
7.4.2.3	Quality values parameter set syntax and semantics
7.4.2.3.1	General
7.4.2.3.2	Quality values parameter set presets
7.4.2.3.2.1	Support of all printable ASCII characters
7.4.2.3.2.2	Quantized quality values, offset 33, range 0-41
7.4.2.3.2.3	Quantized quality values, offset 64, range 0-40
7.4.2.4	Computed Reference parameter set
7.5	Access unit
7.5.1	Syntax and semantics
7.5.1.1	General

7.5.1.2	Access unit header
7.5.1.3	Block
7.5.1.3.1	General
7.5.1.3.2	Block header
7.5.1.3.3	Block payload
7.5.2	Access unit types
8	Descriptors
9	Sequencing reads
9.1	General
9.2	Supported symbols
9.3	Paired-end reads
9.4	Reverse-complement reads
9.5	Data classes
9.6	Aligned data
9.7	Unaligned data
10	Decoding process
10.1	General
10.2	dataset_type = 0 or 1
10.2.1	General
10.2.2	References padding
10.2.3	Type 1 AU (Class P)
10.2.4	Type 2 AU (Class N)
10.2.5	Type 3 AU (Class M)
10.2.6	Type 4 AU (Class I)
10.2.7	Type 5 AU (Class HM)
10.2.8	Type 6 AU (Class U)
10.2.8.1	General
10.2.8.2	cr_alg_ID = 2
10.2.8.3	cr_alg_ID = 4
10.3	dataset_type = 2
10.3.1	General
10.3.2	Type 1 AU
10.3.3	Type 2 AU
10.3.4	Type 3 AU
10.3.5	Type 4 AU
10.3.6	Type 6 AU
10.4	Genomic descriptors
10.4.1	General
10.4.2	pos
10.4.3	rcomp
10.4.4	flags
10.4.5	mmpos
10.4.6	mmtree
10.4.7	clips
10.4.8	ureads
10.4.9	rln
10.4.10	pair
10.4.11	mscore
10.4.12	mmap
10.4.12.1	General
10.4.12.2	Multiple alignments on different sequences
10.4.13	msar
10.4.14	rtype
10.4.14.1	General
10.4.14.2	Pushln
10.4.15	rgroup
10.4.16	qv
10.4.16.1	General
10.4.16.2	Decoding process of the quality values of a genomic record
10.4.16.3	Decoding processes of quality value codebook indexes and quality values of a segment

- 10.4.17 rname
- 10.4.18 rftp
- 10.4.19 rftt
- 10.4.20 tokentype descriptors
  - 10.4.20.1 General
  - 10.4.20.2 Decoding process
  - 10.4.20.3 Syntax and semantics
  - 10.4.20.4 Decoding process for compressed tokens
    - 10.4.20.4.1 General
    - 10.4.20.4.2 COP
    - 10.4.20.4.3 CAT
    - 10.4.20.4.4 RLE
    - 10.4.20.4.5 CABAC\_METHOD\_0
    - 10.4.20.4.6 CABAC\_METHOD\_1
    - 10.4.20.4.7 X4
  - 10.4.20.5 Assembly of tokens
- 10.5 sequence
  - 10.5.1 General
  - 10.5.2 Aligned reads (Classes P, N, M, I, HM)
  - 10.5.3 Unmapped reads (Class HM, U)
- 10.6 e-cigar
  - 10.6.1 Syntax
  - 10.6.2 Decoding process for the first alignment
    - 10.6.2.1 General
    - 10.6.2.2 Decoding process without spliced reads
    - 10.6.2.3 Decoding process with spliced reads
  - 10.6.3 Decoding process for other alignments
  - 10.6.4 Reference transformation
- 11 Representation of reference sequences
  - 11.1 External reference
  - 11.2 Embedded reference
  - 11.3 Computed reference
    - 11.3.1 General
    - 11.3.2 Supported Algorithms
    - 11.3.3 Reference transformation
    - 11.3.4 PushIn
      - 11.3.4.1 General
      - 11.3.4.2 Process for the construction of the reference
      - 11.3.4.3 Initialization of the reference
      - 11.3.4.4 Update of the reference
    - 11.3.5 Local assembly
      - 11.3.5.1 General
      - 11.3.5.2 Process for adding a decoded aligned read to the list crBuf
      - 11.3.5.3 Process for the construction of the reference
      - 11.3.5.4 Decoding process for rftp and rftt
    - 11.3.6 Global assembly
- 12 Block payload parsing process
  - 12.1 General
  - 12.2 Inverse binarizations
    - 12.2.1 General
    - 12.2.2 Binary (BI)
    - 12.2.3 Truncated unary (TU)
    - 12.2.4 Exponential golomb (EG)
      - 12.2.4.1 General
      - 12.2.4.2 Signed exponential golomb (SEG) binarization
    - 12.2.5 If the output of step 2 is 1, symVal= -1\*symValTruncated exponential golomb (TEG)
    - 12.2.6 Signed truncated exponential golomb (STEG)
    - 12.2.7 Split unit-wise truncated unary (SUTU)
    - 12.2.8 Signed split unit-wise truncated unary (SSUTU)
    - 12.2.9 Double truncated unary (DTU)
    - 12.2.10 Signed double truncated unary (SDTU)
  - 12.3 Decoder configuration

- 12.3.1 Sequences and quality values
- 12.3.2 Support values
- 12.3.3 CABAC binarizations
  - 12.3.3.1 General
  - 12.3.3.2 CABAC binarizations parameters
  - 12.3.3.3 CABAC context parameters
- 12.3.4 Transformation parameters
- 12.3.5 Msar descriptor and read identifiers
- 12.3.6 State variables
  - 12.3.6.1 Number of alphabet symbols
  - 12.3.6.2 Number of contexts per subsymbol
  - 12.3.6.3 Coding order context offset
  - 12.3.6.4 Coding size context offset
  - 12.3.6.5 Number of contexts for LUTs
  - 12.3.6.6 Total number of contexts
- 12.4 Initialization process for context variables
- 12.5 Arithmetic decoding engine
  - 12.5.1 Initialization
  - 12.5.2 Arithmetic decoding process
    - 12.5.2.1 General
    - 12.5.2.2 Arithmetic decoding process for a binary decision
      - 12.5.2.2.1 General
      - 12.5.2.2.2 State transition process
    - 12.5.2.3 Renormalization process in the arithmetic decoding engine
    - 12.5.2.4 Bypass decoding process for binary decisions
    - 12.5.2.5 Decoding process for binary decisions before termination
    - 12.5.2.6 Alignment process prior to aligned bypass decoding
- 12.6 Decoding process for sequence descriptors
  - 12.6.1 General
  - 12.6.2 Block payload decoding process
    - 12.6.2.1 General
    - 12.6.2.2 General decoding process for descriptors
    - 12.6.2.3 Dependency reference lookup
    - 12.6.2.4 Context creation and initialization
    - 12.6.2.5 Decoding LUTs
    - 12.6.2.6 Context selection
    - 12.6.2.7 Decoding subsymbols
    - 12.6.2.8 Inverse subsymbol transformations
    - 12.6.2.9 Internal state update
    - 12.6.2.10 Inverse subsequence transformations
      - 12.6.2.10.1 General
      - 12.6.2.10.2 Equality transformation
      - 12.6.2.10.3 Match transformation
      - 12.6.2.10.4 RLE transformation
      - 12.6.2.10.5 Merge transformation

### 13 Output format

- 13.1 General
- 13.2 MPEG-G record
  - 13.2.1 General
  - 13.2.2 number\_of\_template\_segments
  - 13.2.3 number\_of\_record\_segments
  - 13.2.4 number\_of\_alignments
  - 13.2.5 class\_ID
  - 13.2.6 read\_group\_len
  - 13.2.7 reserved
  - 13.2.8 read\_1\_first
  - 13.2.9 seq\_ID
  - 13.2.10 as\_depth
  - 13.2.11 read\_len
  - 13.2.12 qv\_depth
  - 13.2.13 read\_name\_len
  - 13.2.14 read\_name
  - 13.2.15 read\_group

13.2.16	sequence
13.2.17	quality_values
13.2.18	mapping_pos
13.2.19	ecigar_len
13.2.20	ecigar_string
13.2.21	reverse_comp
13.2.22	mapping_score
13.2.23	split_alignment
13.2.24	delta
13.2.25	split_pos
13.2.26	split_seq_ID
13.2.27	flags
13.2.28	more_alignments
13.2.29	next_pos
13.2.30	next_seq_ID
13.3	Initialization process

**Annex A (informative) Tokenization of reads identifiers**

**Annex B (informative) Mapping quality**

**Annex C (informative) Inverse binarization examples**

C.1	Binary (BI) binarization
C.2	Truncated unary (TU) binarization
C.3	Exponential golomb (EG) binarization
C.4	Truncated exponential golomb (TEG) binarization
C.5	Signed truncated exponential golomb (STEG) binarization
C.6	Split unit-wise truncated unary (SUTU) binarization
C.7	Signed split unit-wise truncated unary (SSUTU) binarization
C.8	Double truncated unary (DTU) binarization
C.9	Signed double truncated unary (SDTU) binarization

Page count: 156