Sequence Alignment/Map Optional Fields Specification

The SAM/BAM Format Specification Working Group

29 May 2020

The master version of this document can be found at `https://github.com/samtools/hts-specs`.
This printing is version f2b4e0c from that repository, last modified on the date shown above.

This document is a companion to the *Sequence Alignment/Map Format Specification* that defines the SAM and BAM formats, and to the *CRAM Format Specification* that defines the CRAM format.[1] Alignment records in each of these formats may contain a number of optional fields, each labelled with a *tag* identifying that field's data. This document describes each of the predefined standard tags, and discusses conventions around creating new tags.

1 Standard tags

Predefined standard tags are listed in the following table and described in greater detail in later subsections. Optional fields are usually displayed as `TAG:TYPE:VALUE`; the *type* may be one of `A` (character), `B` (general array), `f` (real number), `H` (hexadecimal array), `i` (integer), or `Z` (string).

Tag	Type	Description
`AM`	i	The smallest template-independent mapping quality in the template
`AS`	i	Alignment score generated by aligner
`BC`	Z	Barcode sequence identifying the sample
`BQ`	Z	Offset to base alignment quality (BAQ)
`BZ`	Z	Phred quality of the unique molecular barcode bases in the `OX` tag
`CB`	Z	Cell identifier
`CC`	Z	Reference name of the next hit
`CG`	B,I	BAM only: CIGAR in BAM's binary encoding if (and only if) it consists of >65535 operators
`CM`	i	Edit distance between the color sequence and the color reference (see also `NM`)
`CO`	Z	Free-text comments
`CP`	i	Leftmost coordinate of the next hit
`CQ`	Z	Color read base qualities
`CR`	Z	Cellular barcode sequence bases (uncorrected)
`CS`	Z	Color read sequence
`CT`	Z	Complete read annotation tag, used for consensus annotation dummy features
`CY`	Z	Phred quality of the cellular barcode sequence in the `CR` tag
`E2`	Z	The 2nd most likely base calls
`FI`	i	The index of segment in the template
`FS`	Z	Segment suffix
`FZ`	B,S	Flow signal intensities
`GC`	?	Reserved for backwards compatibility reasons
`GQ`	?	Reserved for backwards compatibility reasons
`GS`	?	Reserved for backwards compatibility reasons
`H0`	i	Number of perfect hits
`H1`	i	Number of 1-difference hits (see also `NM`)

[1] See `SAMv1.pdf` and `CRAMv3.pdf` at `https://github.com/samtools/hts-specs`.

Tag	Type	Description
H2	i	Number of 2-difference hits
HI	i	Query hit index
IH	i	Query hit total count
LB	Z	Library
MC	Z	CIGAR string for mate/next segment
MD	Z	String encoding mismatched and deleted reference bases
MF	?	Reserved for backwards compatibility reasons
MI	Z	Molecular identifier; a string that uniquely identifies the molecule from which the record was derived
MQ	i	Mapping quality of the mate/next segment
NH	i	Number of reported alignments that contain the query in the current record
NM	i	Edit distance to the reference
OA	Z	Original alignment
OC	Z	Original CIGAR (deprecated; use OA instead)
OP	i	Original mapping position (deprecated; use OA instead)
OQ	Z	Original base quality
OX	Z	Original unique molecular barcode bases
PG	Z	Program
PQ	i	Phred likelihood of the template
PT	Z	Read annotations for parts of the padded read sequence
PU	Z	Platform unit
Q2	Z	Phred quality of the mate/next segment sequence in the R2 tag
QT	Z	Phred quality of the sample barcode sequence in the BC tag
QX	Z	Quality score of the unique molecular identifier in the RX tag
R2	Z	Sequence of the mate/next segment in the template
RG	Z	Read group
RT	?	Reserved for backwards compatibility reasons
RX	Z	Sequence bases of the (possibly corrected) unique molecular identifier
S2	?	Reserved for backwards compatibility reasons
SA	Z	Other canonical alignments in a chimeric alignment
SM	i	Template-independent mapping quality
SQ	?	Reserved for backwards compatibility reasons
TC	i	The number of segments in the template
TS	A	Transcript strand
U2	Z	Phred probability of the 2nd call being wrong conditional on the best being wrong
UQ	i	Phred likelihood of the segment, conditional on the mapping being correct
X?	?	Reserved for end users
Y?	?	Reserved for end users
Z?	?	Reserved for end users

1.1 Additional Template and Mapping data

AM:i:`score` The smallest template-independent mapping quality of any segment in the same template as this read. (See also `SM`.)

AS:i:`score` Alignment score generated by aligner.

BQ:Z:`qualities` Offset to base alignment quality (BAQ), of the same length as the read sequence. At the i-th read base, $\mathrm{BAQ}_i = Q_i - (\mathrm{BQ}_i - 64)$ where Q_i is the i-th base quality.

CC:Z:`rname` Reference name of the next hit; '`=`' for the same chromosome.

CG:B:I,`encodedCigar` Real CIGAR in its binary form if (and only if) it contains >65535 operations. This is a BAM file only tag as a workaround of BAM's incapability to store long CIGARs in the standard way. SAM and CRAM files created with updated tools aware of the workaround are not expected to contain this tag. See also the footnote in Section 4.2 of the SAM spec for details.

CP:i:`pos` Leftmost coordinate of the next hit.

E2:Z:`bases` The 2nd most likely base calls. Same encoding and same length as SEQ. See also `U2` for associated quality values.

FI:i:`int` The index of segment in the template.

FS:Z:`str` Segment suffix.

H0:i:`count` Number of perfect hits.

H1:i:`count` Number of 1-difference hits (see also `NM`).

H2:i:`count` Number of 2-difference hits.

HI:i:*i* Query hit index, indicating the alignment record is the i-th one stored in SAM.

IH:i:`count` Number of alignments stored in the file that contain the query in the current record.

MC:Z:`cigar` CIGAR string for mate/next segment.

MD:Z:`[0-9]+(([A-Z]|\^[A-Z]+)[0-9]+)*`
String encoding mismatched and deleted reference bases, used in conjunction with the CIGAR and SEQ fields to reconstruct the bases of the reference sequence interval to which the alignment has been mapped. This can enable variant calling without requiring access to the entire original reference.

The `MD` string consists of the following items, concatenated without additional delimiter characters:

- `[0-9]+`, indicating a run of reference bases that are identical to the corresponding SEQ bases;
- `[A-Z]`, identifying a single reference base that differs from the SEQ base aligned at that position;
- `\^[A-Z]+`, identifying a run of reference bases that have been deleted in the alignment.

As shown in the complete regular expression above, numbers alternate with the other items. Thus if two mismatches or deletions are adjacent without a run of identical bases between them, a '`0`' (indicating a 0-length run) must be used to separate them in the `MD` string.

Clipping, padding, reference skips, and insertions ('`H`', '`S`', '`P`', '`N`', and '`I`' CIGAR operations) are not represented in the `MD` string. When reconstructing the reference sequence, inserted and soft-clipped SEQ bases are omitted as determined by tracking '`I`' and '`S`' operations in the CIGAR string. (If the CIGAR string contains '`N`' operations, then the corresponding skipped parts of the reference sequence cannot be reconstructed.)

For example, a string '`10A5^AC6`' means from the leftmost reference base in the alignment, there are 10 matches followed by an A on the reference which is different from the aligned read base; the next 5 reference bases are matches followed by a 2bp deletion from the reference; the deleted sequence is AC; the last 6 bases are matches.

MQ:i:`score` Mapping quality of the mate/next segment.

NH:i:`count` Number of reported alignments that contain the query in the current record.

NM:i:`count` Number of differences (mismatches plus inserted and deleted bases) between the sequence and reference, counting only (case-insensitive) A, C, G and T bases in sequence and reference as potential matches, with everything else being a mismatch. Note this means that ambiguity codes in both sequence and reference that match each other, such as '`N`' in both, or compatible codes such as '`A`' and '`R`', are still counted as mismatches. The special sequence base '`=`' will always be considered to be a match, even if the reference is ambiguous at that point. Alignment reference skips, padding, soft and hard clipping ('`N`', '`P`', '`S`' and '`H`' CIGAR operations) do not count as mismatches, but insertions and deletions count as one mismatch per base.

Note that historically this has been ill-defined and both data and tools exist that disagree with this definition.

PQ:i:`score` Phred likelihood of the template, conditional on the mapping locations of both/all segments being correct.

Q2:Z:`qualities` Phred quality of the mate/next segment sequence in the `R2` tag. Same encoding as QUAL.

R2:Z:`bases` Sequence of the mate/next segment in the template. See also `Q2` for any associated quality values.

SA:Z:(*rname*,*pos*,*strand*,*CIGAR*,*mapQ*,*NM*;)+ Other canonical alignments in a chimeric alignment, formatted as a semicolon-delimited list. Each element in the list represents a part of the chimeric alignment. Conventionally, at a supplementary line, the first element points to the primary line. *Strand* is either '+' or '-', indicating forward/reverse strand, corresponding to FLAG bit 0x10. *Pos* is a 1-based coordinate.

SM:i:`score` Template-independent mapping quality, i.e., the mapping quality if the read were mapped as a single read rather than as part of a read pair or template.

TC:i: The number of segments in the template.

TS:A:`strand` Strand ('+' or '-') of the transcript to which the read has been mapped.

U2:Z: Phred probability of the 2nd call being wrong conditional on the best being wrong. The same encoding and length as QUAL. See also `E2` for associated base calls.

UQ:i: Phred likelihood of the segment, conditional on the mapping being correct.

1.2 Metadata

RG:Z:`readgroup` The read group to which the read belongs. If `@RG` headers are present, then *readgroup* must match the `RG-ID` field of one of the headers.

LB:Z:`library` The library from which the read has been sequenced. If `@RG` headers are present, then *library* must match the `RG-LB` field of one of the headers.

PG:Z:`program_id` Program. Value matches the header `PG-ID` tag if `@PG` is present.

PU:Z:`platformunit` The platform unit in which the read was sequenced. If `@RG` headers are present, then *platformunit* must match the `RG-PU` field of one of the headers.

CO:Z:`text` Free-text comments.

1.3 Barcodes

DNA barcodes can be used to identify the provenance of the underlying reads. There are currently three varieties of barcodes that may co-exist: Sample Barcode, Cell Barcode, and Unique Molecular Identifier (UMI).

- Despite its name, the *Sample Barcode* identifies the *Library* and allows multiple libraries to be combined and sequenced together. After sequencing, the reads can be separated according to this barcode and placed in different "read groups" each corresponding to a library. Since the library was generated from a sample, knowing the library should inform of the sample. The barcode itself can be included in the `PU` field in the `RG` header line. Since the `PU` field should be globally unique, it is advisable to include specific information such as flowcell barcode and lane. It is not recommended to use the barcode as the `ID` field of the `RG` header line, as some tools modify this field (e.g., when merging files).
- The *Cell Barcode* is similar to the sample barcode but there is (normally) no control over the assignment of cells to barcodes (whose sequence could be random or predetermined). The Cell Barcode can help identify when reads come from different cells in a "single-cell" sequencing experiment.

- The *UMI* is intended to identify the (single- or double-stranded) molecule at the time that the barcode was introduced. This can be used to inform duplicate marking and make consensus calling in ultra-deep sequencing. Additionally, the UMI can be used to (informatically) link reads that were generated from the same long molecule, enabling long-range phasing and better informed mapping. In some experimental setups opposite strands of the same double-stranded DNA molecule get related barcodes. These templates can also be considered duplicates even though technically they may have different UMIs. Multiple UMIs can be added by a protocol, possibly at different time-points, which means that specific knowledge of the protocol may be needed in order to analyze the resulting data correctly.

BC:Z:`sequence` Barcode sequence (Identifying the sample/library), with any quality scores (optionally) stored in the `QT` tag. The `BC` tag should match the `QT` tag in length. In the case of multiple unique molecular identifiers (e.g., one on each end of the template) the recommended implementation concatenates all the barcodes and places a hyphen (‘`-`’) between the barcodes from the same template.

QT:Z:`qualities` Phred quality of the sample barcode sequence in the `BC` tag. Same encoding as QUAL, i.e., Phred score + 33. In the case of multiple unique molecular identifiers (e.g., one on each end of the template) the recommended implementation concatenates all the quality strings with spaces (‘␣’) between the different strings from the same template.

CB:Z:`str` Cell identifier, consisting of the optionally-corrected cellular barcode sequence and an optional suffix. The sequence part is similar to the `CR` tag, but may have had sequencing errors etc corrected. This may be followed by a suffix consisting of a hyphen (‘`-`’) and one or more alphanumeric characters to form an identifier. In the case of the cellular barcode (`CR`) being based on multiple barcode sequences the recommended implementation concatenates all the (corrected or uncorrected) barcodes with a hyphen (‘`-`’) between the different barcodes. Sequencing errors etc aside, all reads from a single cell are expected to have the same `CB` tag.

CR:Z:`sequence+` Cellular barcode. The uncorrected sequence bases of the cellular barcode as reported by the sequencing machine, with the corresponding base quality scores (optionally) stored in `CY`. Sequencing errors etc aside, all reads with the same `CR` tag likely derive from the same cell. In the case of the cellular barcode being based on multiple barcode sequences the recommended implementation concatenates all the barcodes with a hyphen (‘`-`’) between the different barcodes.

CY:Z:`qualities+` Phred quality of the cellular barcode sequence in the `CR` tag. Same encoding as QUAL, i.e., Phred score + 33. The lengths of the `CY` and `CR` tags must match. In the case of the cellular barcode being based on multiple barcode sequences the recommended implementation concatenates all the quality strings with with spaces (‘␣’) between the different strings.

MI:Z:`str` Molecular Identifier. A unique ID within the SAM file for the source molecule from which this read is derived. All reads with the same `MI` tag represent the group of reads derived from the same source molecule.

OX:Z:`sequence+` Raw (uncorrected) unique molecular identifier bases, with any quality scores (optionally) stored in the `BZ` tag. In the case of multiple unique molecular identifiers (e.g., one on each end of the template) the recommended implementation concatenates all the barcodes with a hyphen (‘`-`’) between the different barcodes.

BZ:Z:`qualities+` Phred quality of the (uncorrected) unique molecular identifier sequence in the `OX` tag. Same encoding as QUAL, i.e., Phred score + 33. The `OX` tags should match the `BZ` tag in length. In the case of multiple unique molecular identifiers (e.g., one on each end of the template) the recommended implementation concatenates all the quality strings with a space (‘␣’) between the different strings.

RX:Z:`sequence+` Sequence bases from the unique molecular identifier. These could be either corrected or uncorrected. Unlike `MI`, the value may be non-unique in the file. Should be comprised of a sequence of bases. In the case of multiple unique molecular identifiers (e.g., one on each end of the template) the recommended implementation concatenates all the barcodes with a hyphen (‘`-`’) between the different barcodes.

If the bases represent corrected bases, the original sequence can be stored in OX (similar to OQ storing the original qualities of bases.)

QX:Z:qualities+ Phred quality of the unique molecular identifier sequence in the RX tag. Same encoding as QUAL, i.e., Phred score + 33. The qualities here may have been corrected (Raw bases and qualities can be stored in OX and BZ respectively.) The lengths of the QX and the RX tags must match. In the case of multiple unique molecular identifiers (e.g., one on each end of the template) the recommended implementation concatenates all the quality strings with a space ('␣') between the different strings.

1.4 Original data

OA:Z:(*RNAME*,*POS*,*strand*,*CIGAR*,*MAPQ*,*NM*;)+ The original alignment information of the record prior to realignment or unalignment by a subsequent tool. Each original alignment entry contains the following six field values from the original record, generally in their textual SAM representations, separated by commas (','), and terminated by a semicolon (';'): RNAME, which must be explicit (unlike RNEXT, '=' may not be used here); 1-based POS; '+' or '-', indicating forward/reverse strand respectively (as per bit 0x10 of FLAG); CIGAR; MAPQ; NM tag value, which may be omitted (though the preceding comma must be retained).

In the presence of an existing OA tag, a subsequent tool may append another original alignment entry after the semicolon, adding to—rather than replacing—the existing OA information.

The OA field is designed to provide record-level information that can be useful for understanding the provenance of the information in a record. It is not designed to provide a complete history of the template alignment information. In particular, realignments resulting in the the removal of Secondary or Supplementary records will cause the loss of all tags associated with those records, and may also leave the SA tag in an invalid state.

OC:Z:cigar Original CIGAR, usually before realignment. Deprecated in favour of the more general OA.

OP:i:pos Original 1-based POS, usually before realignment. Deprecated in favour of the more general OA.

OQ:Z:qualities Original base quality, usually before recalibration. Same encoding as QUAL.

1.5 Annotation and Padding

The SAM format can be used to represent *de novo* assemblies, generally by using padded reference sequences and the annotation tags described here. See the *Guide for Describing Assembly Sequences* in the *SAM Format Specification* for full details of this representation.

CT:Z:*strand*;*type*(;*key*(=*value*)?)*

Complete read annotation tag, used for consensus annotation dummy features.

The CT tag is intended primarily for annotation dummy reads, and consists of a *strand*, *type* and zero or more *key*=*value* pairs, each separated with semicolons. The *strand* field has four values as in GFF3,[2] and supplements FLAG bit 0x10 to allow unstranded ('.'), and stranded but unknown strand ('?') annotation. For these and annotation on the forward strand (*strand* set to '+'), do not set FLAG bit 0x10. For annotation on the reverse strand, set the *strand* to '-' and set FLAG bit 0x10.

The *type* and any *keys* and their optional *values* are all percent encoded according to RFC3986 to escape meta-characters '=', '%', ';', '|' or non-printable characters not matched by the isprint() macro (with the C locale). For example a percent sign becomes '%25'.

PT:Z:*annotag*(\|*annotag*)* where each *annotag* matches *start*;*end*;*strand*;*type*(;*key*(=*value*)?)*

Read annotations for parts of the padded read sequence.

The PT tag value has the format of a series of annotation tags separated by '|', each annotating a sub-region of the read. Each tag consists of *start*, *end*, *strand*, *type* and zero or more *key*=*value* pairs,

[2]The Generic Feature Format version 3 (GFF3) specification can be found at http://sequenceontology.org.

each separated with semicolons. *Start* and *end* are 1-based positions between one and the sum of the `M/I/D/P/S/=/X` CIGAR operators, i.e., SEQ length plus any pads. Note any editing of the CIGAR string may require updating the `PT` tag coordinates, or even invalidate them. As in GFF3, *strand* is one of '`+`' for forward strand tags, '`-`' for reverse strand, '`.`' for unstranded or '`?`' for stranded but unknown strand.

The *type* and any *keys* and their optional *values* are all percent encoded as in the `CT` tag.

1.6 Technology-specific data

FZ:B:S,`intensities` Flow signal intensities on the original strand of the read, stored as `(uint16_t) round(value * 100.0)`.

1.6.1 Color space

CM:i:`distance` Edit distance between the color sequence and the color reference (see also `NM`).

CS:Z:`sequence` Color read sequence on the original strand of the read. The primer base must be included.

CQ:Z:`qualities` Color read quality on the original strand of the read. Same encoding as QUAL; same length as `CS`.

2 Locally-defined tags

You can freely add new tags. Note that tags starting with '`X`', '`Y`', or '`Z`' and tags containing lowercase letters in either position are reserved for local use and will not be formally defined in any future version of this specification.

If a new tag may be of general interest, it may be useful to have it added to this specification. Additions can be proposed by opening a new issue at `https://github.com/samtools/hts-specs/issues` and/or by sending email to `samtools-devel@lists.sourceforge.net`.

Appendix A Tag History

This appendix lists when standard tags were initially defined or significantly changed, and other historical events that affect how tags are interpreted or what files they may appear in.

March 2020

Transcript strand tag TS added, equivalent to the locally-defined XS tag produced by several RNA aligners.

January 2019

Added the OA tag for recording original/previous alignment information.
Deprecated the OC and OP tags.

July 2018

Clarified the calculation of NM score.

May 2018

Cellular barcode tags CB, CR, and CY added.
Removed the RT:Z tag, which was a long-deprecated synonym for BC.

November 2017

SAM version number `VN:1.6` introduced, indicating the addition of the CG tag representation of very long CIGAR strings. Files that contain records with more than 65,535 CIGAR operators should not declare a version number lower than 1.6 in their `@HD` headers.

August 2017

Unique molecular identifier tags BZ, MI, OX, QX, and RX added.
Usage of sample barcode tag BC clarified.

June 2017

Corrected the description of the E2 (second-most-likely bases) tag, which was previously unclear as to whether it contains bases or base qualities.

September 2016

Predefined tags, previously listed as a brief table within the main SAM specification, have been split out into this new document. There is now space for clearer and more complete tag descriptions.

February 2014

MC tag added.

May 2013

SAM version number `VN:1.5` introduced, with limited impact for tags other than indicating that the CT/PT annotation tag definitions are considered finalised.

SA tag added.

March 2012

Descriptions of CT and PT annotation tags significantly clarified.

October 2011

Sample barcode tags QT and RT added, with RT being identified as a deprecated alternative to BC.
Read annotation tags CT and PT added.

September 2011

FZ tag's type changed from `H` to `B,S`-array.
BC and CO tags added.

April 2011

SAM version number `VN:1.4` introduced, indicating the addition of the `B`-array tag type. Files that contain records with `B`-array fields should not declare a version number lower than 1.4 in their `@HD` headers.

FZ tag added, with type `H`.
MD tag description changed to allow IUPAC ambiguity codes in addition to `ACGTN`.

March 2011

CC and CP tags reinstated with their original meanings.

November 2010

BQ tag added.

July 2010

The specification was rewritten as a LaTeX document specifying SAM version number `VN:1.3`.

Tags FI, FS, OC, OP, OQ, and TC added.
Tags GC:Z, GQ:Z, and GS:Z, briefly proposed for representing repeatedly-sequenced reads, noted as reserved for backwards compatibility. Existing tags MF:i (MAQ pair flag), SQ:H (suboptimal bases), and S2:H (mate's suboptimal bases) removed and noted as reserved for backwards compatibility.
CC and CP tags temporarily removed.

July 2009

The original SAM "0.1.2-draft" specification specified version number `VN:1.0` and defined a total of thirty standard tags (though SQ and S2 were already deprecated in favour of E2 and U2):

AM	CM	CS	H1	IH	MF	NM	PU	RG	SQ
AS	CP	E2	H2	LB	MQ	PG	Q2	S2	U2
CC	CQ	H0	HI	MD	NH	PQ	R2	SM	UQ