Data type	Name	Value
itf8	reference sequence id	reference sequence identifier or -1 for unmapped reads -2 for multiple reference sequences. This value must match that of its enclosing container.
itf8	alignment start	the alignment start position
itf8	alignment span	the length of the alignment
itf8	number of records	the number of records in the slice
ltf8	record counter	1-based sequential index of records in the file/stream
itf8	number of blocks	the number of blocks in the slice
itf8[ ]	block content ids	block content ids of the blocks in the slice
itf8	embedded reference bases block content id	block content id for the embedded reference sequence bases or -1 for none
byte[16]	reference md5	MD5 checksum of the reference bases within the slice boundaries. If this slice has reference sequence id of -1 (unmapped) or -2 (multi-ref) the MD5 should be 16 bytes of \0. For embedded references, the MD5 can either be all-zeros or the MD5 of the embedded sequence.
byte[ ]	optional tags	a series of tag,type,value tuples encoded as per BAM auxiliary fields.

The alignment start and alignment span values should only be utilised during decoding if the slice has mapped data aligned to a single reference (reference sequence id >= 0). For multi-reference slices or those with unmapped data, it is recommended to fill these fields with value 0.

MD5sums should not be validated if the stored checksum is all-zero. Embedded references should follow the same capitalisation and alphabetical rules as applied to external references prior to MD5sum calculations. If an embedded reference is used, it is not a requirement that it exactly matches the reference used for sequence alignments. For example, it may contain "N" bases where coverage is absent or it could have different base calls for SNP variants. Hence when embedded sequences are used, the MD5sum refers to the checksum of the embedded sequence and should not be validated against any external reference files.

Note where an embedded reference differs to the original reference used for alignment, the MD and NM tags may need to be stored verbatim for records where the respective embedded and external reference substrings differ.

The optional tags are encoded in the same manner as BAM tags. I.e. a series of binary encoded tags concatenated together where each tag consists of a 2 byte key (matching [A-Za-z][A-Za-z0-9]) followed by a 1 byte type ([AfZHcCsSiIB]) followed by a string of bytes in a format defined by the type.

Tags starting in a capital letter are reserved while lowercase ones or those starting with X, Y or Z are user definable. Any tag not understood by a decoder should be skipped over without producing an error.

At present no tags are defined.

8.6 Core data block

A core data block is a bit stream (most significant bit first) consisting of data from one or more CRAM records. Please note that one byte could hold more then one CRAM record as a minimal CRAM record could be just a few bits long. The core data block has the following fields:

Data type	Name	Value
bit[ ]	CRAM record 1	The first CRAM record
...	...	...
bit[ ]	CRAM record N	The Nth CRAM record