(5) The very start of the file has an identification item (magic number). This item also describes the CBF version or level (see Section 2.3.3.1 below).

(6) In the CBF binary format, the descriptive tags and values may be viewed as a 'header' section for the binary data section. The start of a header section is delimited by the usual CIF `data_` token (see Section 2.3.3.2 below).

(7) The header information must contain sufficient data names to fully describe the binary data sections.

(8) The binary data are presented as 'binary strings', which in CBF are specially formatted binary data within a semicolon-delimited text string. In imgCIF files, the binary data are MIME-encoded within true ASCII text fields.

(9) White space may be used within the pseudo-ASCII sections prior to the 'start of binary section' identifier to align the start-binary data sections to word or block boundaries. Similar use may be made of unused bytes within binary sections. However, no blank lines should be introduced among the MIME headers, since that would terminate processing of those headers and start the scan for binary data. In general, no guarantee is made of block or word alignment in a CBF of unknown origin.

(10) The end of the file need not be explicitly indicated, but including a comment of the form `###_END_OF_CBF` (including the carriage return, line feed pair) can help in debugging.

(11) For the most efficient processing of a CBF file, all binary data described in a single data block should appear as the last information in that data block. However, since binary strings can be parsed anywhere within the context of a CBF or imgCIF file, it is *recommended* that processing software for CBF accept such strings in any order, and it is *mandatory* that processing software for imgCIF accept such strings in any order. The binary identifier values used within a given data block section, and hence the binary data, must be unique for any given `*.array_id`, and it would be best to make them globally unique. However, a different data block may reuse binary identifier values. (This allows concatenation of files without renumbering the binary identifiers, and provides a certain level of localization of data within the file to avoid programs having to search potentially huge files for missing binary sections.)

(12) The recommended file extension for a CBF is 'cbf'. This allows users to recognize file types easily and gives programs a chance to 'know' the file type without having to prompt the user. However, a program should check for at least the file identifier to ensure that the file type is indeed CBF.

(13) The recommended file extensions for imgCIF are 'icf' or 'cif'.

(14) CBF format files are binary files, so when ftp is used to transfer files between different computer systems 'binary' or 'image' mode transfer should be selected.

(15) imgCIF files are ASCII files, so when ftp is used to transfer files between different computer systems 'ascii' transfer should be selected.

2.3.3.1. Details of the magic number

The magic number identifier is `###CBF: VERSION`. This must always be present so that a program can easily identify a CBF by simply inputting the first 15 characters. [The space is a blank (ASCII 32) and not a tab. All identifier characters must be upper case.] The first hash means that this line within the CIF is a comment line. Three hashes mean that this is a line describing the binary file layout for CBF. (All CBF internal identifiers start with three hashes, and all others must immediately follow a 'line separator'.) No white space may precede the first hash sign. Following the file identifier is the version number of the file; *e.g.* the full line might appear as `###CBF: VERSION 1.0`. The version number must be separated from the file identifier characters by white space, *e.g.* a blank (ASCII 32). The version number is defined as a major version number and minor version number separated by a decimal point. A change in the major version may mean that a program for the previous version cannot input the new version as some major change has occurred to CBF. A change in the minor version may also mean incompatibility if the CBF has been written using some new feature. For example, a new form of linearity scaling may be specified; this would be considered a minor version change. A file with the new feature would not be readable by a program supporting only an older version of the format.

2.3.3.2. Details of the header section

The start of a header section is delimited by the usual CIF `data_` token. Optionally, a header identifier, `###_START_OF_HEADER`, may be used before the `data_` token, followed by the carriage return, line feed pair, as an aid in debugging, but it is not required. (Naturally, another carriage return, line feed pair should immediately precede this and all other CBF identifiers, with the exception of the CBF file identifier at the very start of the file.) A header section, including the identification items which delimit it, uses only ASCII characters and is divided into 'lines'. The 'line separator' symbols `\r\n` (carriage return, line feed) are the same regardless of the operating system on which the file is written. This is an important difference from CIF, but must be so, as the file contains binary data and cannot be translated from one operating system to another, which is the case for ASCII text files. While a properly functioning CBF API should write the full `\r\n` line separator, it should recognize any of three sequences `\r`, `\n`, `\r\n` as valid line separators so that hand-edited headers will not be rejected.

The header section in a CBF obeys all CIF rules (Chapter 2.2) with the exception of the line separators, *i.e.*:

(i) 'Lines' are a maximum of 80 characters long. (This will be increased to 2048 in line with the CIF version 1.1 specification.)

(ii) All data names (tags) start with an underscore character '_'.

(iii) The hash symbol '`#`' (outside a quoted character string) means that all text up to the line separator is a comment.

(iv) White space outside character strings is not significant.

(v) Data names are case-insensitive.

(vi) The data item follows the data-name separator and may be of one of two types: character string (char) or number (numb). The type is specified for each data name.

(vii) Character strings may be delimited with single or double quotes, or blocks of text may be delimited by semicolons occurring as the first character on a line.

(viii) The `loop_` mechanism allows a data name to have multiple values. Immediately following the `loop_`, one or more data names are listed without their values, as column headings. Then one or more rows of values are given.

(ix) Any CIF data name may occur within the header section.

The tokens `data_` and `loop_` have special meaning in CIF and should not be used except in their indicated places. The tokens `save_`, `stop_` and `global_` also have special meaning in CIF's parent language, STAR, and should also not be used.

A single header section may contain one or more `data_` blocks.