Example 2.3.4.1. *A multiple-block CBF with several images.*

```
###CBF: VERSION 1.0
# CBF file written by cbflib v0.6

# A comment cannot appear before the file identifier,
# but can appear anywhere else, except within the
# binary sections.

# Here the first data block starts
data_xxx

### ... various CIF tags and values here
###     but none that define array data items

# The "data_" identifier finishes the first data
# block and starts the second
data_yyy

### ... various CIF tags and values here including
###     ones that define array data items

loop_
_array_data.array_id
_array_data.binary_id
_array_data.data

image_1 1
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
     conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BINARY
X-Binary-Size: 3745758
X-Binary-ID: 1
X-Binary-Element-Type: "signed 32-bit integer"
Content-MD5: 1zsJjWPfol2GYl2V+QSXrw==

START_OF_BIN
<D5>^P<B8>P^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@ ...
[This is where the raw binary data would be – we can't print them here]

--CIF-BINARY-FORMAT-SECTION----
;
```

Example 2.3.4.1. *(cont.)*

```
# Following the "end of binary" identifier the file
# is pseudo-ASCII again, so comments are valid
# up to the next "start of binary" identifier.  Note
# that we have increased the binary ID by one.

image_1 2
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
     conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BINARY
X-Binary-Size: 3745758
X-Binary-ID: 2
X-Binary-Element-Type: "signed 32-bit integer"
Content-MD5: xR5kxiOetd9T/Nr5vMfAmA==

START_OF_BIN
<D5>^P<B8>P^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@ ...
[This is where the raw binary data would be – we can't print them here]

--CIF-BINARY-FORMAT-SECTION----
;

# Third binary section; note that we have a new
# array ID.

image_2 3
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
     conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BINARY
X-Binary-ID: 3
Content-MD5: yS5kxiOetd9T/NrqTLfAmA==

START_OF_BIN
*************'9*****`********* ...
[This is where the raw binary data would be – we can't print them here]

--CIF-BINARY-FORMAT-SECTION----
;

# Second header section
data_zzz

### ... various CIF tags and values here
###     including ones that define array
###     data items

# Since we only have one block left, we won't
# use a loop

_array_data.array_id   image
_array_data.binary_id  1
_array_data.data

# Note that we can put a comment here
;
--CIF-BINARY-FORMAT-SECTION--
Content-Type: application/octet-stream;
     conversions="x-CBF_PACKED"
Content-Transfer-Encoding: BINARY
X-Binary-ID: 1
Content-MD5: fooxiOetd9T/serNufAmA==

START_OF_BIN
*************'9*****`********* ...
[This is where the raw binary data would be – we can't print them here]

--CIF-BINARY-FORMAT-SECTION----
;

###_END_OF_CBF
```

2.3.4. A complex example

In Example 2.3.4.1 only the ARRAY_DATA category CIF identifiers are shown. The other CIF data items are not shown. This shows a possible structuring of a more complicated example. There are two header sections; the first contains two data blocks and defines three binary sections. CIF comment lines, starting with a hash mark '#', are used to clarify the structure.

2.3.5. imgCIF encodings

For an imgCIF, there are several alternative encodings for binary image data as ASCII text. Each binary image may use a different encoding in the same imgCIF data set or even in the same data block. The choice of encoding is specified in the 'Content-Transfer-Encoding' MIME header.

If the Transfer Encoding is `X-BASE8`, `X-BASE10` or `X-BASE16`, the data are presented as octal, decimal or hexadecimal data, respectively, organized into lines or words. Each word is created by composing octets of data in fixed groups of 2, 3, 4, 6 or 8 octets, either in the order . . .4321 ('big-endian') or 1234. . . ('little-endian'). If there are fewer than the specified number of octets to fill the last word, then the missing octets are presented as '`==`' for each missing octet. Exactly two equal signs are used for each missing octet even for octal and decimal encoding. The format of lines is

rnd xxxxxx xxxxxx xxxxxx

where *r* is H, O or D for hexadecimal, octal or decimal, *n* is the number of octets per word, and *d* is '<' or '>' for the '. . .4321' and '1234. . .' octet orderings, respectively. The '`==`' padding for the last word should be on the appropriate side to correspond to the missing octets, *e.g.*

```
H4< FFFFFFFF FFFFFFFF 07FFFFFF ====0000
```

or

```
H3> FF0700 00====
```

For these hexadecimal, octal and decimal formats only, comments beginning with '`#`' are permitted to improve readability.

BASE64 encoding follows MIME conventions. Octets are in groups of three, $c1$, $c2$, $c3$. The resulting 24 bits are reorganized in the following way (where we use the C operators $\gg$, $\ll$, & and | to denote, respectively, a right shift, a left shift, bit-wise intersection and bit-wise union). Four six-bit quantities are specified, starting with the high-order six bits $(c1 \gg 2)$ of the first octet, then the low-order two bits of the first octet followed by the high-order four bits of the second octet $((c1 \,\&\, 3) \ll 4 \,|\, (c2 \gg 4))$, then the bottom four bits of the second octet followed by the high-order two bits of the last octet $((c2 \,\&\, 15) \ll 2 \,|\, (c3 \gg 6))$, then the bottom six bits of the last octet $(c3 \,\&\, 63)$. Each of these four quantities is translated into an ASCII character using the mapping

```
          1         2         3         4
01234567890123456789012345678901234567890123456789
 | | | | | | |
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwx
5         6
01234567890123
 | | | | | | |
yz0123456789+/
```

Short groups of octets are padded on the right with one '`=`' if $c3$ is missing, and with '`==`' if both $c2$ and $c3$ are missing.

QUOTED-PRINTABLE encoding also follows MIME conventions, copying octets without translation if their ASCII values are 32...38, 42, 48...57, 59...60, 62, 64...126 and the octet is not a '`;`' in column 1. All other characters are translated to '`=nn`', where `nn` is the hexadecimal encoding of the octet. All lines are 'wrapped' with a terminating '`=`' (*i.e.* the MIME conventions for an implicit line terminator are never used).

Appendix 2.3.1
Deprecated CBF conventions

There was an earlier, now deprecated, CBF format in which the compression type was given as eight bytes of binary header. In this case, the eight bytes used for the compression type are subtracted from the size, so that the same size will be reported if the compression type is supplied in the MIME header. Use of the MIME header is the recommended way to supply the compression type.

These earlier versions of the specification also included three eight-byte words of information in binary that replicated information now available in the MIME header:

5...12	Binary section identifier (see `_array_data.binary_id`), 64-bit, little-endian
13...20	The size (n) of the binary section in octets (*i.e.* the offset from octet 29 to the first byte following the data)
21...28	Compression type:
	`CBF_NONE` 0x0040 (64)
	`CBF_CANONICAL` 0x0050 (80)
	`CBF_PACKED` 0x0060 (96)
	...

The three eight-byte words were followed by binary data. These words are not included when a MIME header is provided.

We are grateful to Frances C. Bernstein, Sydney R. Hall, Brian McMahon and Nick Spadaccini for their helpful comments and suggestions.

References

Freed, N. & Borenstein, N. (1996*a*). *Multipurpose Internet Mail Extensions (MIME) part five: Conformance criteria and examples.* RFC 2049. Network Working Group. http://www.ietf.org/rfc/rfc2049.txt.

Freed, N. & Borenstein, N. (1996*b*). *Multipurpose Internet Mail Extensions (MIME) part one: Format of Internet message bodies.* RFC 2045. Network Working Group. http://www.ietf.org/rfc/rfc2045.txt.

Freed, N. & Borenstein, N. (1996*c*). *Multipurpose Internet Mail Extensions (MIME) part two: Media types.* RFC 2046. Network Working Group. http://www.ietf.org/rfc/rfc2046.txt.

Freed, N., Klensin, J. & Postel, J. (1996). *Multipurpose Internet Mail Extensions (MIME) part four: Registration procedures.* RFC 2048. Network Working Group. http://www.ietf.org/rfc/rfc2048.txt.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

IEEE (1985). *IEEE standard for binary floating-point arithmetic.* ANSI/IEEE Std 754–1985. New York: The Institute of Electrical and Electronics Engineers, Inc.

Moore, K. (1996). *MIME (Multipurpose Internet Mail Extensions) part three: Message header extensions for non-ASCII text.* RFC 2047. Network Working Group. http://www.ietf.org/rfc/rfc2047.txt.

Rivest, R. (1992). *The MD5 message-digest algorithm.* RFC 1321. Network Working Group. http://www.ietf.org/rfc/rfc1321.txt.