biorad1sc_doc Documentation

Release 1.0

Matthew A. Clapp

Feb 27, 2019

Contents:

CHAPTER 1

Introduction

The information in this document has been gleaned from many hours of detective work and examination of 1sc files.

Concurrently with the analysis of this 1sc file format, an implementation of a 1sc file reader was created in python. See: biorad1sc_reader on github and biorad1sc-reader on pypi.

Many thanks to the team of the Open Microscopy Environment. Their software package Bio-Formats and their implementation of "Bio-Rad Gel 1sc file format" gave me the start in investigating the structure of this file.

1.1 License Information

CHAPTER 2

Overall File Structure

All known files are little-endian. Seems to be what “Intel Format” at the top of the file is indicating.

The file is made up of a file header from bytes 0 through 4139, followed by 11 contiguous Data Blocks.

Table 1: **File Header**

File bytes	Numbers or ASCII	Description
0-1	Numbers	0xAF, 0xAF ('Magic Number' 1sc file ID)
2-31	ASCII	`Stable File Version 2.0`\r\n<2 spaces>\r\n
32-55	ASCII	`Intel Format`<10 spaces>\r\n
56-95	ASCII	`Bio-Rad Scan File - ID`<space><17-digit number>
96-135	ASCII	<38 spaces>\r\n
136-139	Numbers	0xC8, 0x00, 0x00, 0x00 (or uint32 0x000000C8 = 200)
140-143	Numbers	0x03, 0x00, 0x00, 0x00 (or uint32 0x00000003 = 3)
144-147	Numbers	0x00, 0x00, 0x00, 0x00 (or uint32 0x00000000 = 0)
148-151	uint32	Start of Data Block 0 (byte offset from start of file)
152-155	uint32	<length of file after file header> Number of bytes from start of Data Block 0 to End Of File.
156-159	Numbers	0x00, 0x00, 0x01, 0x00 (or uint32 0x10000 = 4096)
160-379	Numbers	Data Fields describing Data Blocks 11x 20-byte Fields
380-4139	Numbers	3760 bytes of 0x00
4140-	Mixed	Start of Data Block 0

Within each Data Block are a series of Data Fields. (See Sections *Field Structure* and *Field Types* for descriptions)

Fields can contain references to other fields, by using a uint32 Data ID to refer to other fields. Each referenceable field has its own unique Data ID recorded in its Field Header.

The entire file can be parsed by reading a data format definition of a data Collection in each even-numbered Data Block, with the actual data in the following odd-numbered Data Block. The first Field in each odd-numbered Data Block is always (often?) the root Field of a hierarchical set of data based on references to other Data Fields inside the same Data Block.

Alternatively, after byte 4140, the entire file can be parsed as a series of contiguous Data Fields, with special parsing for Field Type 0 (End Of Data Block). If parsing the entire file at once, (and not each Data Block in isolation,) one

can use the following method when encountering Field Type 0:

1. Parse the End Of Data Block Field
    - Field Type: 0
    - Field Len: 8
    - Field ID: 0
2. Parse the Data Block Footer
    - Keep reading groups of 7x uint16 values until the end of this Data Block, known from reading of the Data Block info fields in the File Header.
3. Parse the next Data Block Header
    - Read 2x uint32 values.

CHAPTER 3

Data Block Structure

Note: Data Block 10, the "Image Data" Data Block, has no Data Block Header, no Data Block Footer, and no Data Fields. It only consists of image data.

All other Data Blocks follow the structure described below.

3.1 Data Block Header

The start of each Data Block starts with 2x uint32 numbers.

The first number is the length in bytes of this Data Block Header and all the following Data Block fields, (including the last field, Field Type 0.) This length does **not** include the Data Block Footer.

The second number is currently of unknown significance. It has been observed to be one of: 1, 2, 4, 7, 8.

Table 1: **Data Block Header**

Bytes	Type	Description
0-3	uint32	Data Block Length in Bytes (Header + Fields)
4-7	uint32	Unknown (1, 2, 4, 7, or 8)

3.2 Data Block Fields

Following the bytes of the Data Block Header, the fields inside the Data Block are parsed contiguously as normal.

The last field of the Data Block fields is Field Type 0. Field Type 0, Field Len 8 signifies End Of Data Block. This field is only a Field Header–the length of 8 bytes only allows for the length of a Field Header.

3.3 Data Block Footer

The data after this Field Type 0 until the end of the Data Block is the Data Block Footer.

The footer is a summary of information about the fields seen in this Data Block. It is composed of groups of 14 bytes. Each group summarizes information on a particular Field Type. The groups are in the following format:

Table 2: **Data Block Footer**

Bytes	Type	Description
0-1	uint16	Item 0 Field Type
2-5	uint32	Item 0 Num. Occurrences A
6-9	uint32	Item 0 Num. Occurrences B
10-13	uint32	Item 0 Unknown
14-15	uint16	Item 1 Field Type
...	...	...

"Occurrences A" and "Occurrences B" sum to the total number of occurrences of the Field Type in the Data Block. They must refer to different types of occurrences, but in which way is unknown.

The Unknown field may be (?) the number of times a given Field Type has been referenced in the Data Block.

CHAPTER 4

Field Structure

Each field in the file is composed of an 8-byte Header, followed by data in the Payload.

Field IDs can be different for the same string in different files. They are not consistent across files.

4.1 Header

Table 1: **Field Header**

Field Bytes	Type	Description
0-1	uint16	Field Type
2-3	uint16	Field Length in bytes (including Header bytes) Value of 1 indicates Field Length of 20
4-7	uint32	Field ID

4.2 Payload

Table 2: **Field Payload**

Field Bytes	Type	Description
8 - <End Of Field>	byte or uint16 or uint32 or mix	Payload Data

CHAPTER 5

Field Types

5.1 Field Referencing Sequence

After the File Header, the basic progression of Fields is as follows:

1. Field Type 102 defining a Collection, with a Label string reference and reference to a Field Type 101 containing definitions of the data in the Collection.
2. Field Type 101 defining multiple data items. Each item has a string reference serving as a label, the Field Type which would contain the actual data, and a corresponding Field Type 100 reference which serves as the Data Key to explain the regions of the data. The Field(s) containing the data follow this Field, **until the next Field Type 102 is found.** When the next Field Type 102 is found, it redefines all info about Data Fields. If Field Type 102 is found before the actual data Field Type is found, then the actual data does not exist for this item.
3. A series of Field Type 100's, serving as Data Keys for each of the Data Items.
4. A series of data container fields, with Field Types greater than 102, usually 1000 and above.

This cycle starts over when the next Field Type 102 is encountered.

The Data Blocks come in pairs. Each even-numbered Data Block (starting with 0) contains field types 102, 101, and 100. These define the structure of the data following in the next Data Block. The following odd-numbered Data Block contains the actual data in field types numbered greater than 102.

The exception to the pattern of pairs of Data Blocks is Data Block 10, containing image data. It has no fields, no previous structure definition, and only contains raw image data.

5.2 NOP Fields

Table 1: **NOP Field Types Summary**

Field Type	Contains References to types	Is Referenced by types	Notes
0	**None**	**None**	End Of Data Block field_id = 0 Data Block Footer and next Data Block Header follows.
2	**None**	1015	nop field? - payload is all 0's, otherwise normal header

5.3 Data Block Info Fields

Data Block Info Fields are special fields found only in the File Header. They define the location and size of the Data Blocks in the file.

Table 2: **Data Block Info Field Type Summary**

Field Type	Contains References to types	Is Referenced by types	Notes
126, 127, 128, 129, 130, 132, 133, 140, 141, 142, or 143	**None**	**None**	Field ID = 0 Field Len = 20 (bytes 2-3 in header uint16 = 0x0001)

5.3.1 Structure

All Data Block Info Fields have the following structure:

Table 3: **Data Block Info Field Structure**

Field bytes	Number Format	Description
0-1	uint16	Field Type
2-3	uint16	0x0001 = 1 Field Len of 20
4-7	uint32	0x0000 = 0 Field ID of 0
8-11	uint32	Data Block start Byte offset from start of file.
12-15	uint32	Data Block length Number of bytes in Data Block.
16-17	uint16?	Data Block number? (except 11 for Data Block 0 Info)
18-19	uint16?	Unknown

5.3.2 Field Types

Table 4: **Data Block Info Field Types**

Field Type	Notes
142	Data Block 0 info
143	Data Block 1 info
132	Data Block 2 info
133	Data Block 3 info
141	Data Block 4 info
140	Data Block 5 info
126	Data Block 6 info
127	Data Block 7 info
128	Data Block 8 info
129	Data Block 9 info
130	Data Block 10 info (image data)

5.4 String Field

Table 5: **String Field Type Summary**

Field Type	Contains References to types	Is Referenced by types	Notes
16	**None**	100, 101, 102, 131, 1000	Previous data fields reference this via Field ID. Null-terminated string. (0x00 is always last byte of payload)

5.5 Data Description Fields

5.5.1 Data Description Fields Hierarchy

In even-numbered Data Blocks, Field Types 102, 101, 100, (and 16) reference each other as follows:

```
102 -> 101 -> 100 -> 16
    \-> 16 \-> 16
```

5.5.2 Field Type 102

Data Collection definition. A **Root Field** of hierarchy.

Table 6: **Field Type 102 Summary**

Field Type	Contains References to types	Is Referenced by types	Notes
102	16, 101	**None**	First field of even-numbered Data Blocks.

Table 7: **Field Type 102 Structure**

Field bytes	Number Format	Description
8-9	uint16	Unknown0
10-11	uint16	Unknown1
12-13	uint16	Unknown2 (1000)
14-15	uint16	Items in Collection
16-19	uint32	Collection: Reference to Field Type 101
20-23	uint32	Label: Reference to Field Type 16 string

5.5.3 Field Type 101

Data Item definitions.

Every 20 bytes defines a data item (one following data container Field Type) until end of field.

Table 8: **Field Type 101 Summary**

Field Type	Contains References to types	Is Referenced by types	Notes
101	16, 100	102	Second field of even-numbered Data Blocks.

Table 9: **Field Type 101 Structure**

Field bytes	Number Format	Description
8-9	uint16	Item 0 Field Type containing data
10-11	uint16	Item 0 Unknown0 (4,5,6,7,16,20,21,22,23)
12-13	uint16	Item 0 Unknown1 (1000)
14-15	uint16	Item 0 Number of regions in data.
16-19	uint32	Item 0 Data Key: Reference to Field Type 100
20-23	uint16	Item 0 Total bytes in data.
24-27	uint32	Item 0 Label: Reference to Field Type 16 string
28-31	uint16	Item 1 Field Type containing data
...	...	...

5.5.4 Field Type 100

Data Key explaining each Data Item in a Collection.

Every 36 bytes is a data region definition, starting at beginning of Field Payload, until end of field. Field ID references are to String Fields later in file.

Num Words, Pointer Byte Offset, and Word Size refer to the payload of a future data container Field Type tied to this key in a Data Item definition in Field Type 101.

It is possible for total bytes in a payload of a corresponding data container field to be a multiple of the bytes defined by this Field Type 100. In this case, the regions defined here would be repeated when parsing the data container field.

Table 10: **Field Type 100 Summary**

Field Type	Contains References to types	Is Referenced by types	Notes
100	16	101	After Field Type 101, this field type has repeated instances until the end of even-numbered Data Block.

Table 11: **Field Type 100 Structure**

Field bytes	Number Format	Description
8-9	uint16	Region 0 Data Type
10-11	uint32	Region 0 Index
12-15	uint32	Region 0 Num Words
16-19	uint32	Region 0 Pointer Byte Offset
20-23	uint32	Region 0 Label: Reference to Field Type 16 string
24-27	uint16	Region 0 Unknown1
28-31	uint32	Region 0 Word Size (bytes) (**or 0x00000000**)[1]
32-33	uint16	Region 0 Unknown2
34-35	uint16	Region 0 Field Type pointed to (if Data Type is reference)
36-39	uint16	Region 0 Unknown4a, 4b (ref.-related)
40-43	uint16	Region 0 Unknown5a, 5b (ref.-related)
44-47	uint16	Region 1 Unknown0
...	...	...

Data Type can be one of the following:

Table 12: **Field Type 100 Region Data Types**

Data Type code	Description	Word Size (bytes)
1	byte	1
2	byte / ASCII	1
3	u?int16	2
4	u?int16	2
5	u?int32	4
6	u?int32	4
7	u?int64	8
9	u?int32	4
10	double (float)	8
15	uint32 Reference	4
17	uint32 Reference	4
21	u?int32	4
100		8
102		16
103		8
107		8
110		8
115		4
120		8
131		12
1000		4
1001		8 (end, first, last, start, taglist), or 24 (cal)
1002		24
1003		8 (faint_loc, small_loc, first, last), or 16 (bounds, where)
1004		8 (tagdef_list), or 16 (bkgd_box, large_box, in, out)
1005		64

Continued on next page

[1] Frustratingly, it appears that in some files for unknown reasons, the Region Word Size sub-field can be 0 for all/most/some regions. In this case word size must be deduced from the Data Type sub-field.

Table 12 – continued from previous page

Data Type code	Description	Word Size (bytes)
1006		12 (runs), or 640 (qinf)
1007		8
1008		8
1009		16
1010		4 (this), or 144 (params)
1011		8
1012		16
1016		440
1017		44
1018		32
1019		8 (a, r), or 36 (sample_list)
1020		32
1021		56
1023		24
1027		8
1028		40
1032		12
1034		16
1035		44
1036		8
1037		8
1038		20
1039		24
1040		20
1041		20
1042		100
1043		20
1044		4
1047		12
1048		40
1049		24
1051		16

5.6 Data Container Fields

Data container fields have Field Types greater than 102. (Note: this may not strictly be true. (?) To be sure treat any Data Field in odd-numbered Data Blocks as data container fields.)

Each of these contains data, the format of which is determined by the last Field Type 100 that is paired with them by an item in Field Type 101.

Field Types of data container fields are often but not limited to: 131, 1000, many numbers greater than 1000.

Part of the data format of data container fields may include references to other field IDs, allowing a hierarchical structure of data container fields. If a region Data Type indicates a Reference, but the actual data is 0, then the region contains no data and should be ignored.

CHAPTER 6

List of Data Blocks

6.1 Data Block 0

Defines the data format for Collection “Overlay Header”.

Field Types: 16, 100, 101, 102

Possible Data Items and their Regions:

- OverlaySave
  - eType
  - color
  - where
  - parentIndex
  - start
  - end
  - startArrow
  - endArrow
  - rotationAngle
  - orientation
  - runs
  - alignment
  - bkgColor
  - bTransparentBkg
  - volumeDataPtr
  - lassoPtr

- OverImgloc
  - x
  - y
- OverImgbox
  - first
  - last
- OverlaySaveArray
  - array
  - avail
  - used
  - regressionType
- OverTextRun
  - string
  - font
  - fontFace
  - fontSize
  - color
  - scriptStyle
  - isBold
  - isItalic
  - isUnderlined
- OverTextRunArray
  - array
  - avail
  - used
- OverVolumeData
  - sumTotal
  - sumBorders
  - numPixels
  - numPixelsBorders
  - minPixelValue
  - maxPixelValue
  - stdDeviation
  - concentration
  - type
  - hasUserLabel

    - string
    - overlaySavePtr
- OverLasso
    - start
    - bounds
    - nsteps
    - swused
    - swavail
    - steps
    - integden
    - pixcnt
    - maxpix
    - minpix

6.2 Data Block 1

Actual data for Collection "Overlay Header". See Data Block 0 for details on possible types of data.

6.3 Data Block 2

Defines the data format for Collection "Q1 Description".

Field Types: 16, 100, 101, 102

Possible Data Items and their Regions:

- Gel
    - file_ver
    - stripe
    - notes
    - nt_used
    - nt_avail
    - stdname
    - stdunits
    - stdtype
    - blotrows
    - blotcols
    - smplwidth
    - bkgden
    - bkgtype

- calcflags
- nbacklog
- backlog
- tdisp_md
- lbkg_md
- lbkg_disk
- lbkg_window
- sensitivity
- min_peak
- noise_filter
- shoulder_sens
- size_scale
- normalize
- use_bandlimit
- shadow
- lbkg_flags
- bandlimit
- tolerance
- match_flags
- qcused
- qcavail
- calcurves
- qtyunits
- vntr_ambig
- flank
- repeat
- vntr_flags
- sim_flags
- sim_tolerance
- sim_required
- asl_used
- asl_avail
- as_links
- allele_set_code
- db_name
- db_path

  - db_filename
  - db_id
  - mod_time
  - taglist
  - db_gelnum
  - db_unit
  - mobilmap
  - db_update
  - db_type
  - adb_gelnum
  - adb_unit
  - adb_taglist
  - flags
  - bstyle
  - difdsp
  - lanes
  - lnused
  - lnavail
  - nxties
  - nyties
  - nties
  - ties
- Stripe
  - dens
  - denused
  - denavail
  - bkgbox
  - minimum
  - average
  - maximum
- Lane
  - name
  - nyties
  - crossings
  - segtrace
  - segused

  - segavail
  - bands
  - bandused
  - bandavail
  - gpk
  - gaussused
  - gaussavail
  - dentrace
  - stdlanenum
  - right_stdlanenum
  - right_frac
  - smplwidth
  - lanenum
  - flags
  - calcflags
  - sumden
  - sumd_bands
  - lbkg_disk
  - lbkg_window
  - lbkg_flags
  - dtparm
  - db_sample
  - db_band_set
  - db_standard
  - dmt_used
  - dmt_avail
  - db_mobil
  - db_bset_flags
  - adb_band_set
  - adb_sample
  - lbkg_md
- Lane Pointer
  - lane pointer
- Trace
  - dvused
  - dvavail

  - dvals
  - srcstrace
  - navg
  - min
  - max
  - avg
  - bkdvals
  - gaussdvused
  - gaussdvavail
  - gaussdvals
- Tdiag
  - diag
  - xaxis
  - yaxis
  - data
  - srctrace
  - dsttrace
  - lanenum
  - datawidth
  - firstden
  - max
- Band
  - name
  - sumden
  - rf
  - stdval
  - quality
  - norm_den
  - calnum
  - qty
  - this
  - first
  - peak
  - last
  - maxpix
  - minpix

  - lasso
  - db_btp_code
  - db_btp_flags
  - adb_btp_code
  - adb_btp_flags
  - stdsource
  - flags
  - qtysource
- Band Pointer
  - band pointer
- Lasso
  - start
  - bounds
  - nsteps
  - swused
  - swavail
  - steps
  - integden
  - pixcnt
  - maxpix
  - minpix
- Band Link
  - lanenum
  - Bandnum
- Imgloc
  - x
  - y
- Imgbox
  - first
  - last
- Band Pointer
  - unowned band pointer
- Calcurve
  - name
  - desc
  - from

  - cbused
  - cbavail
  - calbands
  - ninterp
  - intps
  - slope
  - intercept
  - corr_coef
  - calnum
  - mcode
  - model
  - extrapolate
  - status
  - type
  - named
- Calcurve Pointer
  - calcurve pointer
- Calband
  - band
  - measure
  - qty
  - reldev
  - dilution
  - dilution_txt
  - qtysource
  - relstat
- Calintp
  - measure
  - qty
- Crosstie
  - left
  - ax
- Crdloc
  - x
  - y
- Stretcloc

    - a
    - r
- MobilTie
    - rf
    - mobility
    - bst_idx
    - btp_code
- AlleleSetLink
    - name
    - id_safety
    - allele_set
    - als_item
- UserDetect
    - sensitivity
    - min_peak
    - noise_filter
    - shoulder_sens
    - size_scale
    - normalize
    - use_bandlimit
    - shadow
    - bandlimit
- BackLog
    - type
    - minden
    - maxden
- Note
    - head
    - tail
    - text_start
    - text
    - flags
- tag
    - pr_code
    - vl_code
- taglist

  - used
  - avail
  - tags
- StandardTie
  - std
  - mobility
- MobilMap
  - lanenum
  - used
  - stdties
- DifDsp Layout
  - mode
  - ratio
  - differ
- GaussPeak
  - center
  - sigma
  - height
  - gauerr
  - lolim
  - hilim
- GaussPeak Pointer
  - gspk pointer

6.4 Data Block 3

Actual data for Collection "Q1 Description". See Data Block 2 for details on possible types of data.

6.5 Data Block 4

Defines the data format for Collection "DDB Description".

Field Types: 16, 100, 101, 102

Possible Data Items and their Regions:

- tag
  - pr_code
  - vl_code
- taglist

  - used
  - avail
  - tags
- tag_value
  - references
  - decode
- tagdef
  - prompt
  - references
  - used
  - avail
  - values
- tagdef_list
  - used
  - avail
  - tagdefs
- band
  - quality
  - std_value
  - norm_den
  - btp_code
  - flags
  - peak
- lane
  - bands_used
  - bands_avail
  - bands
  - sample_code
  - bst_code
  - flags
  - dentrace
  - dmt_used
  - dmt_avail
  - db_mobil
- gel
  - path

  - filename
  - id
  - name
  - description
  - cre_time
  - mod_time
  - update
  - lanes_used
  - lanes_avail
  - lanes
  - taglist
  - mobilmap
  - lanewidth
  - detection
  - unit
  - gidx
  - stdtype
  - lbkg_md
  - lbkg_disk
  - lbkg_status
  - layout
- gel pointer
  - gel pointer
- sample
  - name
  - cre_time
  - description
  - taglist
  - idx_used
  - idx_avail
  - indices
  - flags
- sample pointer
  - sample pointer
- band_type
  - name

  - btp_code
  - index
  - gidx
  - lanenum
  - low_std
  - ideal_std
  - high_std
  - low_sf
  - ideal_sf
  - high_sf
- band set
  - name
  - cre_time
  - mod_time
  - idx_used
  - idx_avail
  - index
  - comment
  - id
  - tolerance
  - bst_idx
  - bt_used
  - bt_avail
  - bt_valid
  - band_types
  - taglist
  - tagdefs
  - unit
  - norm_btp_code
  - gidx
  - lanenum
  - method
  - modified
  - code_style
  - display_names
  - report_names

  - type
  - unit_change
  - model_vers
- band set pointer
  - band set pointer
- base
  - name
  - description
  - cre_time
  - mod_time
  - id
  - pathname
  - gels_used
  - gels_avail
  - gels
  - gel_sorting
  - gel_sort_tag
  - gel_count
  - gtpl_used
  - gtpl_avail
  - gtpl_count
  - gel_templates
  - smpl_used
  - smpl_avail
  - samples
  - sample_sorting
  - sample_count
  - bst_used
  - bst_avail
  - band_sets
  - bst_sorting
  - bst_count
  - srch_used
  - srch_avail
  - srch_count
  - searches

  - tagdef_list
  - layouts
  - units_used
  - units_avail
  - units
  - pop_used
  - pop_avail
  - pop_count
  - pop_links
  - seg_map
  - db_type
- layouts
  - sum
  - gel_list
  - sample_detail
  - sample_list
  - gel_detail
  - bset
  - srch
  - odrep
  - dbp
  - difdsp
  - detect
- gel_list_layout
  - sel_name
  - sel_date_from
  - sel_date_to
  - sel_tag1
  - sel_tag2
  - sort_by
  - lst_pr_code
  - dbpos
- sample_detail_layout
  - tagdefs
  - dbpos
- sample_list_layout

  - sel_tagdef1
  - sel_tagdef2
  - lst_tagdef1
  - lst_tagdef2
  - sort_by
  - dbpos
- geldet_layout
  - gel_tagdef1
  - gel_tagdef2
  - sample_tagdef1
  - sample_tagdef2
  - sort_by
  - flags
  - dbpos
- bset_layout
  - unit
  - tagdefs
  - default_bset
  - lg_dbpos
  - sm_dbpos
- unit
  - longname
  - shortname
  - unitname
  - interp
  - order
  - flags
- unit pointer
  - unit pointer
- reference lane
  - gidx
  - lanenum
  - bst_idx
- search
  - name
  - smplname

  - date_from
  - date_to
  - taglist
  - tagdefs
  - match
  - ref_smpl
  - match_percent
  - nlanes
  - ref_lanes
  - srchnum
  - search_by
  - compare
  - sim_method
  - weighting
  - edited
  - include
  - useGaussModelsIfPresent
- search pointer
  - search pointer
- search layout
  - match_percent
  - srchnum
  - tagdefs
  - sim_method
  - include
  - weighting
  - dbpos
- lane index
  - gidx
  - lanenum
  - bst_idx
- pop link
  - name
  - plidx
  - dir_block
  - data_block

- pop link pointer
  - poplink pointer
- segment map
  - first
  - nsegs
  - segs
- dbp_pr_coldata_fields
  - type
  - value
- pr layout
  - ref_lnum
  - cols_used
  - coldata
  - flags
  - font
- sum layout
  - style
  - lg_dbpos
  - sm_dbpos
- imgloc
  - x
  - y
- imgres
  - x
  - y
- ddb position
  - loc
  - size
  - flags
- dbp ptree layout
  - dp_pos
  - method
- dbp pca layout
  - dp_pos
- dbp popfrm layout
  - dp_pos

- dbp layouts
  - popfrm
  - pr
  - ptree
  - pca
  - irp
- irp layout
  - cols_used
  - coldata
  - ref
  - order
  - active
  - style
  - pg_layout
  - show_btypes
  - ruler
  - ref_lnum
- odrep layout
  - od_types
- mobilmap
  - lanenum
  - used
  - stdties
- standardtie
  - std
  - mobility
- DifDsp Layout
  - mode
  - ratio
  - differ
- detect layout
  - userdet
  - screenloc
  - lane_width
  - manual
  - style

    - valid
- userdetect
    - sensitivity
    - min_peak
    - noise_filter
    - shoulder_sens
    - size_scale
    - normalize
    - use_bandlimit
    - shadow
    - bandlimit
- dentrace
    - dvused
    - dvavail
    - dvals
    - srctrace
    - navg
    - min
    - max
    - avg
    - bkdvals
    - gaussdvused
    - gaussdvavail
    - gaussdvals
    - gaussmax
    - gaussmin
- imgbox
    - first
    - last
- db_mobil.
    - rf
    - mobility
    - bst_idx
    - btp_code

6.6 Data Block 5

Actual data for Collection "DDB Description". See Data Block 4 for details on possible types of data.

6.7 Data Block 6

Defines the data format for Collection "Audit Trail".

Field Types: 16, 100, 101, 102

Possible Data Items and their Regions:

- AuditTrail
  - m_entries
  - m_userPool
  - m_descPool
  - m_appPool
- AuditTrailEntry
  - m_time
  - m_user
  - m_description
  - m_details
  - m_detailX1
  - m_detailY1
  - m_detailX2
  - m_detailY2
  - m_version
  - m_comment
  - m_filter
  - m_locked
- AuditTrailEntryPtr
  - AuditTrailEntryPtr
- AuditTrailEntryPtrVector
  - m_mmvectorList
  - m_mmvectorUsed
  - m_mmvectorAvail
- AuditTrailStringPool
  - m_pool
- AuditTrailStringVector
  - m_mmvectorList

  - m_mmvectorUsed
  - m_mmvectorAvail
- Imgloc
  - x
  - y
- Imgres
  - x
  - y
- Imgbox
  - first
  - last
- Crdloc
  - x
  - y
- Crdres
  - x
  - y
- Crdbox
  - first
  - last
- Crdscale
  - x
  - y
- ImgState
  - mincon
  - maxcon
  - in
  - out
  - low_frac
  - high_frac
  - state
  - gamma
  - aspect
- Savemap
  - center
  - scale

- CRealPoint
  - m_x
  - m_y
- CRealSize
  - m_width
  - m_height
- CRealDistance
  - m_x
  - m_y
- CRealLine
  - m_start
  - m_end
- CRealRect
  - m_top
  - m_left
  - m_right
  - m_bottom
- CImagePoint
  - m_x
  - m_y
- CImageSize
  - m_width
  - m_height
- CImageDistance
  - m_x
  - m_y
- CImageLine
  - m_start
  - m_end
- CImageRect
  - m_top
  - m_left
  - m_right
  - m_bottom
- CWindowPoint
  - m_x

    - m_y
- CWindowSize
    - m_width
    - m_height
- CWindowDistance
    - m_x
    - m_y
- CWindowLine
    - m_start
    - m_end
- CWindowRect
    - m_top
    - m_left
    - m_right
    - m_bottom
- sm_string
    - m_buffer
    - m_length
- mm_string
    - m_buffer
    - m_length

6.8 Data Block 7

Actual data for Collection "Audit Trail". See Data Block 6 for details on possible types of data.

6.9 Data Block 8

Defines the data format for Collection "Scan Header".

Field Types: 16, 100, 101, 102

Possible Data Items and their Regions:

- SCN
    - filevers
    - creation_date
    - last_use_date
    - user_id
    - prog_name

- scanner
- old_description
- old_comment
- desc
- pH_orient
- Mr_orient
- nxpix
- nypix
- data_fmt
- bytes_per_pix
- endian
- max_OD
- pix_at_max_OD
- img_size_x
- img_size_y
- min_pix
- max_pix
- mean_pix
- data_ceiling
- data_floor
- cal
- formula
- imgstate
- qinf
- params
- history
- color
- light_mode
- size_mode
- norm_pix
- bkgd_pix
- faint_loc
- small_loc
- large_box
- bkgd_box
- dtct_parm_name

  - m_id32
  - m_scnId
  - m_imagePK
- ScnCalibInfo
  - calfmt
  - dettyp
  - isotop
  - gel_run_date
  - cnts_loaded
  - xpo_start_date
  - xpo_length
- ScnFormula
  - type
  - units
  - c_pro
  - c_exp
- ScnImgloc
  - x
  - y
- ScnImgbox
  - first
  - last
- ScnImgState
  - mincon
  - maxcon
  - in
  - out
  - low_frac
  - high_frac
  - state
  - gamma
  - aspect
- ScnQtyInfo
  - qty_range
  - qty_units
  - blackIsZero

  - scanner_maxpix
  - scanner_units
  - scanner_bias
  - scanner_maxqty
  - calstep_count
  - calstep_raw
  - calstep_qty
  - calstep_qty_offset
  - gray_response_data
  - gray_response_len
  - gray_response_factor
- ScnCrdloc
  - x
  - y
- ScnCrdres
  - x
  - y
- ScnCrdbox
  - first
  - last
- ScnParams
  - resolution
  - scan_area
  - exposure_time
  - ref_bkg_time
  - gain_setting
  - light_mode
  - color
  - intf_type
  - size_mode
  - imaging_mode
  - filter_name1
  - filter_name2
  - filter_name3
  - filter_name4
  - filter_name5

  - filter_id1
  - filter_id2
  - filter_id3
  - filter_id4
  - filter_id5
  - laser_name1
  - laser_name2
  - laser_name3
  - laser_name4
  - laser_name5
  - laser_id1
  - laser_id2
  - laser_id3
  - laser_id4
  - laser_id5
  - pmt_voltage
  - dark_type
  - live_count
  - app_name
  - flat_field
- GrayResponseData
  - GR_Data

6.10 Data Block 9

Actual data for Collection "Scan Header". See Data Block 8 for details on possible types of data.

6.11 Data Block 10

Only image data, no fields

Image data in this block is only pixel data, organized starting from bottom-left of image to upper-right. The first bytes of this data define the pixels of the bottom row, from left to right. The next bytes are the second-to-bottom row from left to right, etc.

All known images are little-endian, 16-bit grayscale. Although the metadata may define another format. (See e.g. 'Scan Header' -> 'SCN' -> {'endian', 'bytes_per_pix', 'data_fmt' })

CHAPTER 7

Indices and tables

- genindex
- search