EVLA Correlator Output Binary Data Format Specification

Martin Pokorny

Revision history

Revision	Date	Description	Name
0.1	16 Jan 2007	Initial version.	mpp
0.2	15 Feb 2007	Update after first internal EVLA review.	mpp
0.3	In progress	Partly updated after input from ALMA and CASA personnel.	mpp

Contents

Comments on this document vii

1 Introduction 1
- 1.1 Context . . . 1
- 1.2 Relation to ALMA correlator ouput data stream document . . . 1
- 1.3 Relation to CASA *Science Data Model* . . . 1

2 Output format MIME structure 3

3 Integration header 6
- 3.1 `sdmDataHeader` element . . . 6
  - 3.1.1 Children . . . 6
  - 3.1.2 Attributes . . . 6
- 3.2 `time` element . . . 7
- 3.3 `dataOID` element . . . 7
  - 3.3.1 Attributes . . . 7
- 3.4 `execBlock` element . . . 7
  - 3.4.1 Children . . . 7
  - 3.4.2 Attributes . . . 7
- 3.5 `scanNum` element . . . 8
- 3.6 `subscanNum` element . . . 8
- 3.7 `integrationNum` element . . . 8
- 3.8 `numAntenna` element . . . 8
- 3.9 `axes` attribute . . . 8
- 3.10 `baselineFlags` element . . . 8
  - 3.10.1 Attributes . . . 8
- 3.11 `actualTimes` element . . . 9
  - 3.11.1 Attributes . . . 9
- 3.12 `actualDurations` element . . . 9
  - 3.12.1 Elements . . . 9
  - 3.12.2 Attributes . . . 9
- 3.13 `lut` element . . . 9
  - 3.13.1 Elements . . . 9
  - 3.13.2 Attributes . . . 10
- 3.14 `zeroLags` element . . . 10
  - 3.14.1 Attributes . . . 10
- 3.15 `numAPC` element . . . 10
- 3.16 `baseband` element . . . 10
  - 3.16.1 Children . . . 10
- 3.17 `spectralWindow` element . . . 11
  - 3.17.1 Attributes . . . 11
- 3.18 `crossData` element . . . 11
  - 3.18.1 Attributes . . . 11
- 3.19 `autoData` element . . . 12

3.19.1 Attributes . . . 12

4 Binary tables **13**
4.1 Null size tables . . . 14
4.2 Omitting axes . . . 14
4.3 Value ordering . . . 15
4.4 Table descriptions . . . 15
4.4.1 *actualTimes* table . . . 15
4.4.2 *actualDurations* table . . . 16
4.4.3 *baselineFlags* table . . . 17
4.4.4 *zeroLags* table . . . 17
4.4.5 *crossData* table . . . 17
4.4.6 *autoData* table . . . 18

A Appendices **19**
A.1 Output data file example . . . 19
A.2 Example of omitted axes in binary table . . . 21
A.3 Binary table sizes . . . 22
A.4 Data header XML schema . . . 23
A.4.1 RELAX NG compact syntax schema . . . 23
A.4.2 W3C XML schema . . . 28

1 References **33**

Figures

4.1 Binary table index structure 13

Tables

4.1 Binary table index order and dependencies. Abbreviations: bl, baseline; bb, baseband; sw, spectral window; pb, phase bin; ab, APC bin; sc, spectral channel; pp, polarization product. 14
A.1 Table size variables 23

Examples

A.1 Omitted axes table header snippet 22

Comments on this document

This document is currently undergoing revision. Some changes have been made relative to the last numbered revision (0.2), but further changes will yet occur.

A direct comparison of an aspect of this document with the corresponding ALMA document is indicated as a "Note", *e.g.*,

Note

this is a note.

Other comparisons may, of course, be made by the reader.

Comments on the text are indicated slightly differently:

Remark

this is a comment.

Such comments are used to indicate unsettled issues regarding this document, or questions that I have not yet satisfactorily answered; they will be removed from the final version of the document.

1 Introduction

1.1 Context

The correlator output binary data format applies to the output data stream of the fast formatter component of the WIDAR correlator backend. The data produced in the binary data format are primarily spectral in nature, although limited supplemental, non-spectral data are also a part of the format. Typically, the data stream is sent to the EVLA archive and TelCal, while header data are also sent to DCAF for inclusion in the science data model. Note that there may be other data products produced by the correlator backend; however, their specification is not addressed by the present document.

As of this writing, the fast formatter will write its output data stream to a shared filesystem. In the EVLA design, the backend consists of a cluster of computers that accepts the correlator hardware-produced lag frames as input, processes the lag sets, and writes the output data files. One node in the cluster will have the role of the fast formatter, which will initialize the output files prior to access by the other cluster nodes, provide output data not available to individual nodes, and send the completed files to the EVLA archive (additionally, the fast formatter node will also provide data to DCAF, but that data is not of concern here.) Cluster nodes handling the lag frames will write data to the output file(s) concurrently as needed. Of course, nothing in this specification prohibits sending the data stream over a network rather than being produced as a file.

1.2 Relation to ALMA correlator ouput data stream document

The present document was based on the ALMA *Correlator Output Data Stream Description* document [CODSD] (dated 2006-03-08) as a jumping-off point. In the process of creating the present specification, the ALMA document has been altered to meet the requirements of the EVLA project, and hence the present document should not be seen as a direct descendent of the ALMA document. Where appropriate, comments have been inserted into the present document to highlight and describe differences with the ALMA document for those readers familiar with that document. Nevertheless, the present document is meant to stand on its own as a self-contained and complete specification for the EVLA project. In addition, whereas the ALMA document specifies both the output in the binary data format and the output sent to ALMA's "data capture" subsystem, the present document addresses only the binary data format output.

1.3 Relation to CASA *Science Data Model*

The correlator output binary data format defines files that are to be a part of the complete scientific data set associated with an EVLA observation. The format of the complete scientific data of an observation is defined by the so-called *Science Data Model* (or SDM) maintained jointly by the **CASA** group at NRAO, and the EVLA project. The CASA group will provide the image processing software for the EVLA, and, as such, will be designing and implementing one of the primary "consumers" of the correlator output binary data files. The specification of both the SDM and the binary data format given by this document are, therefore, jointly developed and maintained by both groups. A document for the EVLA project specifying the SDM, and providing important contextual information related to the present specification, will be produced in the near future.

Remark

Joe McMullin should be listed as a co-author or co-editor of this document after he has had the opportunity to review it.

2 Output format MIME structure

The overall format of a binary output data file is given by the MIME[1] specification [RFC2045], [RFC2046], [RFC2387]. In the context of the MIME standard, the binary output data file consists of one *multipart* MIME message, comprising one or more multipart sub-parts. In other words, a data file is a nested (or tree-structured) multipart MIME message.

One MIME message, or file, comprises data for a set of integrations from one subarray, where the data for each integration are themselves contained in a multipart message that is one sub-part of the top-level MIME message. At the top level, the file has a MIME type of *multipart/mixed*, which is the standard MIME type for multipart messages wherein the sub-parts are ordered[2]. In turn, the data for one integration are formatted as a MIME message of type *multipart/related*, which indicates that the sub-parts within the message are related and may have references to other sub-parts within the same message. The message for an integration has one sub-part for the data header, and several additional sub-parts for the binary data tables.

Note

This specification allows the aggregation of multiple integrations into a single data file, whereas that is not the case for ALMA.

Which data are aggregated into a single file is ultimately an operational or policy issue. It is likely that all of the integrations in one sub-scan will be aggregated into a single file. However, to limit file size, it may be necessary to limit the number of integrations in one file, in which case the integrations in one sub-scan will span several files. Nothing in this specification requires either policy, nor any other policy in this regard.

An example of a partial binary output data file is provided below to illustrate the high-level structure. Note that the example is not normative; except as noted, any discrepancies in the example with respect to the MIME standard are to be resolved in favor of the MIME standard.

```
MIME-Version: 1.0 ❶

Content-type: multipart/mixed; boundary="integration_boundary"; ❷
type="text/plain"
Content-description: correlator spectral data set
❸
--integration_boundary ❹
Content-type: multipart/related;
              boundary="abcd";
              start=<hdr//X1/1/0/0>
Content-description: correlator spectral data
Content-id: <uid://X1/1/0/0> ❺

--abcd ❻
Content-type: text/xml; charset=iso-8859-1
Content-transfer-encoding: 8bit
```

[1] Multipurpose Internet Mail Extensions

[2] Ordering of the sub-parts is required only because the final sub-part of the top-level message does not contain integration data, but rather provides an index to the integrations within the file.

Output format MIME structure

```
Content-id: <hdr//X1/1/0/0> ❼
<sdmDataHeader>
...
<actualTimes href="cid:actualTimes//X1/1/0/0" ❽.../>
...
</sdmDataHeader>

--abcd ❾
Content-type: application/octet-stream ❿
Content-id: <actualTimes//X1/1/0/0>
[BINARY DATA]
--abcd
... ⓫
--abcd--

--integration_boundary ⓬
Content-type: multipart/related;
              boundary="ABCD";
              start=<hdr//X1/1/0/1>
Content-description: correlator spectral data
Content-id: <uid://X1/1/0/1>

--ABCD
Content-type: text/xml; charset=iso-8859-1
Content-transfer-encoding: 8bit
Content-id: <hdr//X1/1/0/1>
<sdmDataHeader>
<actualTimes href="cid:actualTimes//X1/1/0/1" .../>
...
</sdmDataHeader>

--ABCD
Content-type: application/octet-stream
Content-id: <actualTimes//X1/1/0/1>
[BINARY DATA]
--ABCD
...
--ABCD--

--integration_boundary
uid://X1/1/0/0 XXX ⓭
uid://X1/1/0/1 YYY
--integration_boundary--
```

❶ CRLF line endings required throughout.

❷ Boundary delimiters must be generated carefully to adhere to the MIME standard, especially in light of the nested multipart nature of this file. While it would be desirable to conform to the standard, it is not clear whether conformance in this regard is necessary or practical for EVLA purposes.

❸

Note

ALMA has other MIME header fields, but they are optional according to the standard.

❹ Each MIME sub-part at the top level corresponds to one backend integration. The data for each integration are themselves formatted as a multipart MIME message. This MIME header provides the MIME container for the integration, and associates an identifier with this part of the data file.

❺ `Content-id` uniquely identifies the data in this sub-part (*i.e.*, the data of this integration).

❻ Data header for this integration. This sub-part contains a true *data* header, providing metadata for the binary data.

❼ `Content-id` value is derived from the unique identifier for data of this integration.

❽ References to `Content-id` values within *multipart/related* MIME messages are described in [RFC2111].

❾ First binary data table for this integration. `Content-id` value is again derived from the unique identifier for the data of this integration.

❿

Note

ALMA uses *binary/octet-stream* `Content-type`.

⓫ Other binary tables.

⓬ The second integration in the data file.

⓭ Index for mapping the `Content-id` value of each integration to the byte offset within the file of the beginning of the MIME part for that integration.

Note

ALMA has a `alma-uid` MIME header field, which, for EVLA, has been integrated with the `Content-ID` field of the MIME sub-part of each integration.

A more complete example of an output file may be found in **Section A.1**.

3 Integration header

The primary purposes of this header, which appears once per integration, are to provide an identifier for the data file, and to provide the metadata needed to read the binary data from the file. References to the binary tables in the file are represented in the header by MIME *Content-ID*s, which are identifiers for the sub-parts of multipart MIME documents. The header is in the form of an XML document conforming to the schema given in **Section A.4**. The following sections provide short descriptions of the header element and its descendents.

Note

Several elements in the header have a `ref` attribute, used to refer to other sub-parts of the same multipart MIME message (that is, within the same integration). These references use the Content-ID URI scheme [RFC2111], which provides a syntax for referring to Content-IDs within multipart MIME messages. Because Content-IDs are required to be world-unique, the values of the `ref` attributes cannot be fixed for EVLA, in contrast to the ALMA specification.

The following sections should be read with reference to the schema (**Section A.4**). These sections are meant to provide a descriptive summary of the elements and attributes in the integration header, and do not provide enough information to produce a conforming XML header document.

3.1 `sdmDataHeader` element

The top-level element; all other elements are descendents of this element.

3.1.1 Children

time, **dataOID**, **execBlock**, **numAntenna**, **baselineFlags**, **actualTimes**, **actualDurations**, **zeroLags**, **numAPC**, **baseband**, **crossData** , **autoData**

3.1.2 Attributes

`byteOrder`

Binary data endianness.

Note

ALMA has fixed the value to be “IEEE Low_Endian”. EVLA prefers to allow the choice of values “little endian” and “big endian” because 1) there’s no reason for the schema to fix the value if the attribute is present (whether or not it’s fixed in practice is another issue), and 2) the term is non-standard.

`axisOrder`

Axes are given an integer id as follows: 1. polarization product, 2. spectral channel, 3. apc, 4. bin, 5. spectral window, 6. baseband, 7. baseline. Data in tables are ordered by a *column-major* traversal of the axes in this order. Note that the attribute value is fixed to be ’1234567’.

`schemaVersion`

Schema version number.

Note

There should be some convention for the version numbering that accomodates both ALMA and EVLA yet differentiates them.

3.2 `time` element

Scheduled time for the centroid of the integration. Units are MJD plus fractional day.

Remark

The representation of this element has not been determined.

3.3 `dataOID` element

Unique identifier for the integration. This is actually a reference to the data for this integration itself, in essence a self-reference. It will be used by the SDM as a reference to this data.

3.3.1 Attributes

`xlink:href`

Reference to the identifier for this integration.

`xlink:title`

Description of the data.

3.4 `execBlock` element

Execution block identifier; uniquely identifies an instance of each execution of a scheduling block.

3.4.1 Children

scanNum, **subscanNum**, **integrationNum**

3.4.2 Attributes

`xlink:href`

Reference to containing(?) execBlock.

3.5 scanNum element

Scan number.

3.6 subscanNum element

Sub-scan number.

3.7 integrationNum element

Integration number.

3.8 numAntenna element

Number of antennas in subarray.

3.9 axes attribute

Attribute to declare axes used in supplemental data binary tables. Although the order of the axes cannot be changed, some axes may be omitted. The axis names correspond to the axis integer identifiers as given in the "axisOrder" attribute of the "sdmDataHeader" element. Note that it is the set of *non-omitted* (i.e, present) axes that must be given.

Note

This attribute is an EVLA extension, it does not exist for ALMA.

3.10 baselineFlags element

baselineFlags table metadata.

3.10.1 Attributes

`size`

Size of binary table. (bytes)

`ref`

MIME message reference to MIME Content-id of binary table, may be in URI format (RFC 2111).

`axes`

See **Section 3.9**.

3.11 `actualTimes` element

actualTimes table metadata.

3.11.1 Attributes

`size`

Size of binary table. (bytes)

`ref`

MIME message reference to MIME Content-id of binary table, may be in URI format (RFC 2111).

`axes`

See **Section 3.9**.

3.12 `actualDurations` element

actualDurations table metadata.

3.12.1 Elements

lut

3.12.2 Attributes

`size`

Size of binary table. (bytes)

`ref`

MIME message reference to MIME Content-id of binary table, may be in URI format (RFC 2111).

`axes`

See **Section 3.9**.

3.13 `lut` element

Lookup table definition. Index values are implicit in the ordering of the floating point values given by the child elements of the 'lut' element, from index 0 up to, at most, index 2^{nbits} - 1.

3.13.1 Elements

Double precision floating point values

3.13.2 Attributes

`nbits`

Lookup index bitfield length.

3.14 zeroLags element

zeroLags table metadata.

3.14.1 Attributes

`size`

Size of binary table. (bytes)

`ref`

MIME message reference to MIME Content-id of binary table, may be in URI format (RFC 2111).

`axes`

See **Section 3.9**.

3.15 numAPC element

Number of data sets for atmospheric phase correction. For EVLA, the value is normally equal to one.

Note

`numAPC` has been raised from the ALMA `correlationMode` element, which EVLA doesn't have.

3.16 baseband element

Container for "spectralWindows" in a baseband.

Note

Attributes of this element for ALMA have been pushed down to the `spectralWindow` level for EVLA because of the level at which these attributes apply. For example, the number of channels may vary with spectral window for EVLA, whereas, for ALMA, it varies with baseband.

`baseband` elements have been raised from ALMA `correlationMode` element as well.

3.16.1 Children

spectralWindow

3.17 spectralWindow element

Metadata of a spectral window related to structure of the binary tables.

3.17.1 Attributes

numSpectralPoint

Number of spectral channels.

numBin

Number of phase bins.

Note

The nominal use of the bins for EVLA is pulsar phase binning.

numPolProduct

Number of polarization products.

scaleFactor

Scaling factor for spectral data.

When the type attribute of the crossData element has the value "float", this attribute shall be ignored.

Note

This attribute is optional for EVLA, but not for ALMA. EVLA will not use this attribute initially because the spectral data will be in floating point format.

3.18 crossData element

Table structure metadata of the cross-correlation spectral data binary table.

Note

crossData has been raised from ALMA correlationMode element too.

3.18.1 Attributes

size

Size of binary table. (bytes)

type

Data type of spectral data. 'float': IEEE 754 single precision floating point; 'short': 16-bit twos-complement integer; 'long': 32-bit twos-complement integer.

`ref`

MIME message reference to MIME Content-id of binary table, may be in URI format (RFC 2111).

3.19 `autoData` element

Table structure metadata of the auto-correlation spectral data binary table.

3.19.1 Attributes

`size`

Size of binary table. (bytes)

`ref`

MIME message reference to MIME Content-id of binary table, may be in URI format (RFC 2111).

4 Binary tables

The binary tables contain the data associated with a backend integration. As the name suggests, these tables contain binary data, and remain unencoded in the MIME format. Indexing of each table can be represented by a constant height ordered tree, with fixed length data elements stored in the leaf nodes of the tree (see **Figure 4.1** for a small example). Note that, although the indexing of the binary tables assumes a tree structure, the data elements are organized in contiguous, linearly-addressed memory segments. The order of the data elements in memory corresponds to an in-order traversal of the tree's leaf nodes; that is, the indexes are lexicographically ordered.

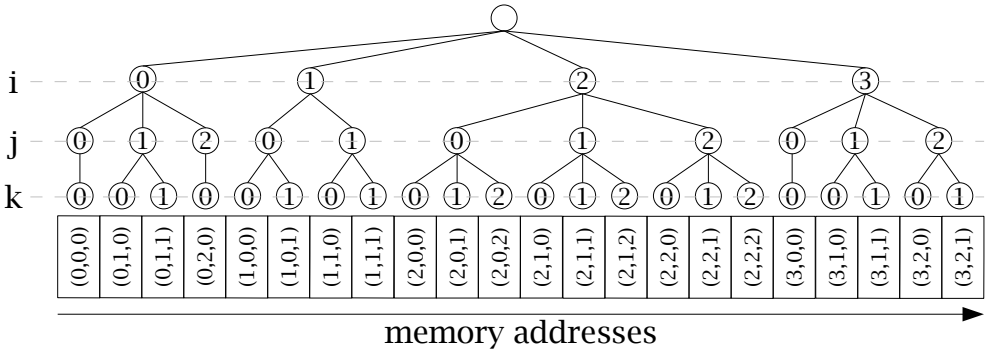

Figure 4.1 Binary table index structure

While the tree representation of binary table indexes is completely generic, the definitions of the binary tables exhibit constraints on the number of child nodes at every level of the tree. Additionally, the constraints may be a function of the tree branch (or sub-tree) in which the nodes occur. The following sections describe the order of the axes (or tree levels) in each binary table, any hard constraints on the size of each axis, the dependencies of the size of each axis on preceding axis coordinate values (equivalently, the dependency on tree branch), and the data stored in the tables.

The following table (**Table 4.1**) shows all of the axes that may be present in the binary tables, their order, and allowed size dependencies. Note that the axes present in any given binary table may be a subsequence of the axis sequence shown below. An "X" in the table below indicates that the size of the axis named in that row may depend on the coordinate value of the axis named in that column. If there is no "X" in a row, the axis named in that row has a constant size in each data file. The order of the axes in the binary tables is represented below by the order of the rows from top to bottom.

Note

The axis ordering presented in this document is in the direction of descending the levels of the index tree as described above, which results in a lexicographic ordering of tree indices (which is a generalization of row-major array order). In contrast, although the ALMA document specifies the same ordering of elements in memory, the axis order presented in that document is the reverse of what is given here, which implies that the indices must be traversed in column-major order to result in an unaltered ordering of data elements. Unfortunately, because the ALMA convention is embedded in the definition of an SDM header element attribute, the EVLA *schema* uses the ALMA convention; however, this *document* describes the ordering

based on the lexicographic convention because it agrees with the model presented above, is likely to be more familiar to C, C++ and Java programmers, and is consistent with an "array of arrays" implementation these programmers might use.

axis	dependence						
	bl	bb	sw	pb	ab	sc	pp
bl							
bb							
sw		X					
pb		X	X				
ab							
sc		X	X				
pp		X	X				

Table 4.1 Binary table index order and dependencies. Abbreviations: bl, baseline; bb, baseband; sw, spectral window; pb, phase bin; ab, APC bin; sc, spectral channel; pp, polarization product.

4.1 Null size tables

Whenever the integration header indicates that the size of a table is zero bytes, that binary table shall be omitted from the data stream (which means that the entire MIME part for the affected table will not be present.) This may occur when, for example, no flags are set during an integration.

Note

The ALMA document discusses the occurrence of optional tables in the data stream (dependent upon the correlator in use), but such tables never occur for EVLA.

4.2 Omitting axes

As an implementation optimization, certain axes in the binary tables may be omitted.[3] These omissions provide a means of data compression, and do not affect the set of axes over which the data apply. This form of data compression is not done "on-the-fly", but is instead configured before the start of an integration, and merely constrains the data that is written to the data stream. Logically, the data in each data table are always applicable over all axes, whereas the tables in the data stream, created with the knowledge that data will not vary over some of the axes, may be implemented with a reduced dimensionality. This optimization is an EVLA extension, which has no counterpart in the ALMA format at this time. An omitted axis indicates simply that the data are the same for all values along that logical axis (in the subspace defined by the coordinate values on the (row-major-wise) preceding axes.) The designation of omitted axes is provided in the data stream header. See **Section A.2** for an example of a binary table using omitted axes.

[3] Another way of thinking about the omitted axes is to consider them to be present, but with a length of one.

Allowing table axes to be omitted has another application besides data compression. All binary tables may vary over all seven axes, but, in some instances, certain axes will never appear in practice. Thus, some tables will appear to be, for example, six dimensional rather than seven dimensional. However, under our model, tables with reduced dimensionality may be implemented as seven dimensional tables with omitted axes. The only difference in this application compared with the use of omitted axes for data compression is that the *logical* table index may be presented as having fewer than seven dimensions. Because this application of axes omission requires changes to neither the implementation nor the specification of the data format, we will continue to present the specification as if all binary data tables had seven axes, with the allowance that axes may be omitted.

4.3 Value ordering

An ordered set of values exists along each axis of the binary tables. This section explicitly specifies the ordering for each axis, which is constant over all tables.

baseline axis

Column-major traversal of the upper triangle of the *(antenna A, antenna B)* matrix (with antennas having integer ordering), followed by the ordered diagonal elements of the same matrix. While the previous sentence may be concise, it's probably not clear, so we present an example.

Assume there are four antennas with index values $\{1, 2, 3, 4\}$. The ordering of baseline axis values is then

$$((1,2),(1,3),(2,3),(1,4),(2,4),(3,4),(1,1),(2,2),(3,3),(4,4)).$$

baseband axis, spectral window axis, bin number axis, APC bin axis, spectral channel axis

Integer ordering.

polarization product axis

Determined by the SDM, and it may vary. Normally, the order will be RR, RL, LR, LL.

4.4 Table descriptions

This section provides descriptions of the data elements in each of the binary tables in the output format. An ordered list of the axes over which each table ranges is also provided, although it should be remembered that the lists only represent the typical cases and are not restrictions in the format.

4.4.1 *actualTimes* table

quantity

Timestamp at the middle of the integration.

Note

The analogous quantity in the ALMA binary format marks the centroid of the actual integration interval, taking into account data blanking.

units

Modified Julian Date plus fractional day

element size

16 bytes

data type

double precision floating point pair

Remark

Tentative data type.

axes (typical)

baseline, baseband, spectral window

4.4.2 *actualDurations* table

Note that there are two supported formats for this table: a simple table of elements (similar to every other table), or a table of index values to be used in conjunction with a lookup table. When using a lookup table, the lookup table is stored as the **lut** child element of the `actualDurations` child element of the `sdmDataheader` element.

quantity

Integration duration (specifically *not* including lost data). A special usage is allowed for coarse-grained flagging, which is signified by using a negative value for the duration.

units

seconds

element size

8 bytes by default, but may be otherwise through the use of a lookup table.

data type

Double precision floating point; or, in the case of a mapping defined by a lookup table, an integer-valued bitstring that maps to a double precision floating point value.

axes (typical)

baseline, baseband, spectral window, bin, APC, spectral channel, polarization product

4.4.3 *baselineFlags* table

quantity

Baseline-based flags that occur during the integration.

units

undecided (bitfield?)

element size

4 bytes

data type

unsigned integer (or N/A)

axes (typical)

baseline, baseband, spectral window, bin, APC, spectral channel, polarization product

4.4.4 *zeroLags* table

quantity

Lag zero values.

units

undecided

element size

2 × 4 bytes

data type

complex single precision floating point

axes (typical)

baseline, baseband, spectral window, bin, APC, polarization product

4.4.5 *crossData* table

quantity

Cross-correlation data; may be either lag spectra or lags.

units

undecided

element size

2 × (2 or 4 bytes), depending on value of **type** attribute of `crossData` header element

data type

complex single precision floating point or complex integer, depending on value of **type** attribute of `crossData` header element

axes (typical)

baseline, baseband, spectral window, bin, APC, spectral channel, polarization product

4.4.6 *autoData* table

quantity

Auto-correlation data; may be either lag spectra or lags.

units

undecided

element size

2×4 bytes

data type

complex single precision floating point

axes (typical)

baseline, baseband, spectral window, bin, APC, spectral channel, polarization product

A Appendices

A.1 Output data file example

Remark

The table sizes in this example need to be corrected.

```
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="integration_boundary";
              type="text/plain"
Content-description: correlator spectral data set

--integration_boundary
Content-type: multipart/related; boundary="abcd";
              start=<hdr//X1/1/0/0>
Content-description: correlator spectral data
Content-id: <uid://X1/1/0/0>

--abcd
Content-type: text/xml; charset=iso-8859-1
Content-transfer-encoding: 8bit
Content-id: <hdr//X1/1/0/0>
<sdmDataHeader axisOrder="12345678" byteOrder="little endian"
               schemaVersion="0.9"
               xmlns:xlink="http://www.w3.org/1999/xlink">
  <time>123456789</time>
  <dataOID xlink:href="uid://X1/1/0/0"
           xlink:title="correlator spectral data"/>
  <execBlock xlink:href="uid://X1/1/2/3">
    <scanNum>1</scanNum>
    <subscanNum>1</subscanNum>
    <integrationNum>25</integrationNum>
  </execBlock>
  <numAntenna>24</numAntenna>
  <baselineFlags size="8208384" ref="cid:baselineFlags//X1/1/0/0"
                 axes="a1 a2 a3 a4 a5 a6 a7"/>
  <actualTimes size="12064" ref="cid:actualTimes//X1/1/0/0"
                          axes="a1 a3 a4 a5 a6 a7"/>
  <actualDurations size="16416768" ref="cid:actualDurations//X1/1/0/0"
                   axes="a1 a2 a3 a4 a5 a6 a7"/>
  <zeroLags size="6032" ref="cid:zeroLags//X1/1/0/0" axes="a1 a3 a4 a5 a6 a7"/>
  <numAPC>1</numAPC>
  <baseband>
    <spectralWindow numBin="1" numSpectralPoint="512" numPolProduct="4"/>
    <spectralWindow numBin="1" numSpectralPoint="512" numPolProduct="4"/>
  </baseband>
  <baseband>
    <spectralWindow numBin="1000" numSpectralPoint="1024" numPolProduct="1"/>
    <spectralWindow numBin="500" numSpectralPoint="2048" numPolProduct="1"/>
  </baseband>
  <correlatedData type="float" size="8208384" ref="cid:crossData//X1/1/0/0"/>
  <autoData type="float" size="8208384" ref="cid:autoData//X1/1/0/0"/>
</sdmDataHeader>
```

```
--abcd
Content-type: application/octet-stream
Content-id: <actualTimes//X1/1/0/0>
[BINARY DATA]
--abcd
Content-type: application/octet-stream
Content-id: <actualDurations//X1/1/0/0>
[BINARY DATA]
--abcd
Content-type: application/octet-stream
Content-id: <crossData//X1/1/0/0>
[BINARY DATA]
--abcd
Content-type: application/octet-stream
Content-id: <zeroLags//X1/1/0/0>
[BINARY DATA]
--abcd
Content-type: application/octet-stream
Content-id: <baselineFlags//X1/1/0/0>
[BINARY DATA]
--abcd
Content-type: application/octet-stream
Content-id: <autoData//X1/1/0/0>
[BINARY DATA]
--abcd--

--integration_boundary
Content-type: multipart/related; boundary="ABCD";
              start=<hdr//X1/1/0/1>
Content-description: correlator spectral data
Content-id: <uid://X1/1/0/1>

--ABCD
Content-type: text/xml; charset=iso-8859-1
Content-transfer-encoding: 8bit
Content-id: <hdr//X1/1/0/1>
<sdmDataHeader axisOrder="12345678" byteOrder="little endian"
               schemaVersion="0.9"
               xmlns:xlink="http://www.w3.org/1999/xlink">
  <time>123460000</time>
  <dataOID xlink:href="uid://X1/1/0/1"
           xlink:title="correlator spectral data"/>
  <execBlock xlink:href="uid://X1/1/2/3">
    <scanNum>1</scanNum>
    <subscanNum>1</subscanNum>
    <integrationNum>26</integrationNum>
  </execBlock>
  <numAntenna>24</numAntenna>
  <baselineFlags size="8208384" ref="cid:baselineFlags//X1/1/0/1"
                 axes="a1 a2 a3 a4 a5 a6 a7"/>
  <actualTimes size="12064" ref="cid:actualTimes//X1/1/0/1"
                          axes="a1 a3 a4 a5 a6 a7"/>
  <actualDurations size="16416768" ref="cid:actualDurations//X1/1/0/1"
                   axes="a1 a2 a3 a4 a5 a6 a7"/>
  <zeroLags size="6032" ref="cid:zeroLags//X1/1/0/1" axes="a1 a3 a4 a5 a6 a7"/>
  <numAPC>1</numAPC>
  <baseband>
```

```
    <spectralWindow numBin="1" numSpectralPoint="512" numPolProduct="4"/>
    <spectralWindow numBin="1" numSpectralPoint="512" numPolProduct="4"/>
  </baseband>
  <baseband>
    <spectralWindow numBin="1000" numSpectralPoint="1024" numPolProduct="1"/>
    <spectralWindow numBin="500" numSpectralPoint="2048" numPolProduct="1"/>
  </baseband>
  <correlatedData type="float" size="8208384" ref="cid:crossData//X1/1/0/1"/>
  <autoData type="float" size="8208384" ref="cid:autoData//X1/1/0/1"/>
</sdmDataHeader>

--ABCD
Content-type: application/octet-stream
Content-id: <actualTimes//X1/1/0/1>
[BINARY DATA]
--ABCD
Content-type: application/octet-stream
Content-id: <actualDurations//X1/1/0/1>
[BINARY DATA]
--ABCD
Content-type: application/octet-stream
Content-id: <crossData//X1/1/0/1>
[BINARY DATA]
--ABCD
Content-type: application/octet-stream
Content-id: <zeroLags//X1/1/0/1>
[BINARY DATA]
--ABCD
Content-type: application/octet-stream
Content-id: <baselineFlags//X1/1/0/1>
[BINARY DATA]
--ABCD
Content-type: application/octet-stream
Content-id: <autoData//X1/1/0/1>
[BINARY DATA]
--ABCD--

--integration_boundary
uid://X1/1/0/0 XXX
uid://X1/1/0/1 YYY
--integration_boundary--
```

A.2 Example of omitted axes in binary table

Consider using the *actualDurations* table to store one weight per polarization product per spectral window per baseband per baseline. Assume, further, that the number of phase bins and APC bins is fixed to be one in all spectral windows. Note that the number of spectral channels may vary with spectral window, but there is only *one weight per polarization product that applies to all spectral channels* within each spectral window. The key phrase in the foregoing sentence, which is emphasized, is a sign that the spectral channels axis may be omitted from the table in the data stream. The following XML snippet would be part of the `sdmHeaderData` element that describes this table.

Remark

This example needs some updating.

```
<actualDurations size="18144"❶ ref="actualDurations"
                 axes="a1 a3 a4 a5 a6 a7"❷/>
<numAntenna>27</numAntenna>
<numAPC>1</numAPC>
<baseband>
  <spectralWindow numBin="1" numSpectralPoint="512" numPolProduct="2"/>
  <spectralWindow numBin="1" numSpectralPoint="1024" numPolProduct="4"/>
</baseband>❸
<correlatedData type="float" size="15482880"❹ ref="crossData"/>
```

❶ $8 \cdot 378 \cdot (4 + 2)$ [using **Equation A.1** and **Section 4.4.2**]

❷ "a2", the spectral channel axis, is missing.

❸ The number of spectral channels and polarization products vary among spectral windows within the baseband.

❹ $2 \cdot 4 \cdot 378 \cdot (512 \cdot 2 + 1024 \cdot 4)$ [using **Equation A.1** and **Section 4.4.5**]

Example A.1 Omitted axes table header snippet

The *actualDurations* table will be indexed as $A_{bl,bb,sw,pb,ab,sc,pp}$. However, for optimization, the table in the data stream has omitted the *sc* axis since the data have no dependency on spectral channel; that is, the data in the table are indexed as $T_{bl,bb,sw,pb,ab,pp}$. The mapping of an index for A to the equivalent index for T is simply $(bl, bb, sw, pb, ab, sc, pp) \mapsto (bl, bb, sw, pb, ab, pp)$. For example, the following statements hold.

$A_{15,0,0,0,0,100,1} = A_{15,0,0,0,0,200,1}$
$A_{15,0,0,0,0,100,1} = T_{15,0,0,0,0,1}$
$A_{15,0,0,0,0,200,1} = T_{15,0,0,0,0,1}$
$A_{15,0,0,0,0,42,1} \neq A_{15,0,1,0,0,42,1} \Leftrightarrow T_{15,0,0,0,0,1} \neq T_{15,0,1,0,0,1}$

A.3 Binary table sizes

A formula is given in this section for the number of data elements in the binary tables.

Variables that are not noted in **Table A.1** as being constant may appear with subscripts in the formula. Each of the variables from the table appears in the formula with a "prime" symbol to indicate that, if the corresponding axis has been omitted from the binary table in the data stream, the variable has a value of one.

variable	description (number of)	range (EVLA)	notes
B	basebands	1-8	constant
N	antennas	1-32	constant
L	baselines	1-496	constant for any table: *crossData* table includes only off-diagonal baselines, *autoData* and *zeroLag* tables include only on-diagonal baselines, all other tables include all baselines
W	spectral windows	1-16	can vary by baseband
F	phase bins	1-65,536	can vary by baseband and spectral window
A	APC bins	1-2	constant
C	spectral channels	1-262,144	can vary by baseband and spectral window
P	polarization products	1,2,4	can vary by baseband and spectral window

Table A.1 Table size variables

$$L'A'\sum_{i=1}^{B'}\sum_{j=1}^{W_i'}F'_{ij}C'_{ij}P'_{ij}$$

Equation A.1 Binary table size

A.4 Data header XML schema

The schema is specified here in the compact syntax form of the RELAX NG schema language.[4] A schema in the W3C XML schema language is also provided for reference, although it should be noted that this schema has been derived from the RELAX NG format schema.

A.4.1 RELAX NG compact syntax schema

```
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"
namespace xlink = "http://www.w3.org/1999/xlink"

start = sdmDataHeader

sdmDataHeader =
  element sdmDataHeader {
```

[4] See **http://relaxng.org** for information.

```
    [ a:documentation [ xml:id = "doc.byteOrder"
      "Binary data endianness." ] ]
    attribute byteOrder { "little endian" | "big endian" },

    [ a:documentation [ xml:id = "doc.axisOrder"
      "Axes are given an integer id as follows: 1. polarization " ~
      "product, 2. spectral channel, 3. apc, 4. bin, 5. spectral window, " ~
      "6. baseband, 7. baseline. Data in tables are ordered by a " ~
      "*column-major* traversal of the axes in this order. Note that " ~
      "the attribute value is fixed to be '1234567'." ] ]
    attribute axisOrder { "1234567" },

    [ a:documentation [ xml:id = "doc.schemaVersion"
      "Schema version number." ] ]
    attribute schemaVersion { xsd:decimal },

    [ a:documentation [ xml:id = "doc.time"
      "Scheduled time for the centroid of the integration. Units are " ~
      "MJD plus fractional day." ] ]
    element time { xsd:double, xsd:double },
    dataOID,
    execBlock,
    numAntenna,
    baselineFlags,
    actualTimes,
    actualDurations,
    zeroLags,

    numAPC,

    baseband+,

    crossData,

    autoData
  }

[ a:documentation [ xml:id = "doc.axes"
  'Attribute to declare axes used in supplemental data binary tables. ' ~
  'Although the order of the axes cannot be changed, some ' ~
  'axes may be omitted. The axis names correspond to the axis integer ' ~
  'identifiers as given in the "axisOrder" attribute of the ' ~
  '"sdmDataHeader" element. Note that it is the set of *non-omitted* ' ~
  '(i.e, present) axes that must be given.' ] ]
axes =
   attribute axes {
      list { "a1"?, "a2"?, "a3"?, "a4"?, "a5"?, "a6"?, "a7"? }
   }

[ a:documentation [ xml:id = "doc.baseband"
  'Container for "spectralWindows" in a baseband.' ] ]
baseband = element baseband { spectralWindow+ }

[ a:documentation [ xml:id = "doc.spectralWindow"
  "Metadata of a spectral window related to structure of the " ~
  "binary tables." ] ]
spectralWindow =
```

```
  element spectralWindow {
    [ a:documentation [ xml:id = "doc.numSpectralPoint"
      "Number of spectral channels." ] ]
    attribute numSpectralPoint { xsd:positiveInteger },

    [ a:documentation [ xml:id = "doc.numBin"
      "Number of phase bins." ] ]
    attribute numBin { xsd:positiveInteger },

    [ a:documentation [ xml:id = "doc.numPolProduct"
      "Number of polarization products." ] ]
    attribute numPolProduct {
      xsd:integer "1" | xsd:integer "2" | xsd:integer "4"
    },

    [ a:documentation [ xml:id = "doc.scaleFactor"
      "Scaling factor for spectral data." ] ]
    attribute scaleFactor { xsd:float }?
  }

[ a:documentation [ xml:id = "doc.crossData"
  "Table structure metadata of the cross-correlation spectral data " ~
   "binary table." ] ]
crossData =
  element crossData {
    [ a:documentation [ xml:id = "doc.crossData.size"
      "Size of binary table. (bytes)" ] ]
    attribute size { xsd:positiveInteger },

    [ a:documentation [ xml:id = "doc.crossData.type"
      "Data type of spectral data. 'float': IEEE 754 single precision " ~
      "floating point; 'short': 16-bit twos-complement integer; " ~
      "'long': 32-bit twos-complement integer." ] ]
    attribute type { "float" | "short" | "long" },

    [ a:documentation [ xml:id = "doc.crossData.ref"
      "MIME message reference to MIME Content-id of binary table, " ~
      "may be in URI format (RFC 2111)."] ]
    attribute ref { xsd:NMTOKEN }
  }

[ a:documentation [ xml:id = "doc.autoData"
  "Table structure metadata of the auto-correlation spectral data " ~
   "binary table." ] ]
autoData =
  element autoData {
    [ a:documentation [ xml:id = "doc.autoData.size"
      "Size of binary table. (bytes)" ] ]
    attribute size { xsd:positiveInteger },

    [ a:documentation [ xml:id = "doc.autoData.ref"
      "MIME message reference to MIME Content-id of binary table, " ~
      "may be in URI format (RFC 2111)."] ]
    attribute ref { xsd:NMTOKEN }
  }
```

```
[ a:documentation [ xml:id = "doc.baselineFlags"
  "baselineFlags table metadata." ] ]
baselineFlags =
  element baselineFlags {
    [ a:documentation [ xml:id = "doc.baselineFlags.size"
      "Size of binary table. (bytes)" ] ]
    attribute size { xsd:positiveInteger },

    [ a:documentation [ xml:id = "doc.baselineFlags.ref"
      "MIME message reference to MIME Content-id of binary table, " ~
      "may be in URI format (RFC 2111)." ] ]
    attribute ref { xsd:NMTOKEN },
    axes
  }

[ a:documentation [ xml:id = "doc.actualTimes"
  "actualTimes table metadata." ] ]
actualTimes =
  element actualTimes {
    [ a:documentation [ xml:id = "doc.actualTimes.size"
      "Size of binary table. (bytes)" ] ]
    attribute size { xsd:positiveInteger },

    [ a:documentation [ xml:id = "doc.actualTimes.ref"
      "MIME message reference to MIME Content-id of binary table, " ~
      "may be in URI format (RFC 2111)." ] ]
    attribute ref { xsd:NMTOKEN },
    axes
  }

[ a:documentation [ xml:id = "doc.lut"
  "Lookup table definition. Index values are implicit in the ordering " ~
  "of the floating point values given by the child elements of the 'lut' " ~
  "element, from index 0 up to, at most, index 2^{nbits} - 1." ] ]
lut =
  element lut {
    [ a:documentation [ xml:id = "doc.lut.nbits"
      "Lookup index bitfield length." ] ]
    attribute nbits { xsd:positiveInteger },

    xsd:double+
  }

[ a:documentation [ xml:id = "doc.actualDurations"
  "actualDurations table metadata." ] ]
actualDurations =
  element actualDurations {
    [ a:documentation [ xml:id = "doc.actualDurations.size"
      "Size of binary table. (bytes)" ] ]
    attribute size { xsd:positiveInteger },

    [ a:documentation [ xml:id = "doc.actualDurations.ref"
      "MIME message reference to MIME Content-id of binary table, " ~
      "may be in URI format (RFC 2111)." ] ]
    attribute ref { xsd:NMTOKEN },
```

```
    axes,
    lut?
  }

[ a:documentation [ xml:id = "doc.zeroLags"
  "zeroLags table metadata." ] ]
zeroLags =
  element zeroLags {
    [ a:documentation [ xml:id = "doc.zeroLags.size"
      "Size of binary table. (bytes)" ] ]
    attribute size { xsd:positiveInteger },

    [ a:documentation [ xml:id = "doc.zeroLags.ref"
      "MIME message reference to MIME Content-id of binary table, " ~
      "may be in URI format (RFC 2111)." ] ]
    attribute ref { xsd:NMTOKEN },
    axes
  }

[ a:documentation [ xml:id= "doc.numAntenna"
  "Number of antennas in subarray." ] ]
numAntenna = element numAntenna { xsd:positiveInteger }

[ a:documentation [ xml:id = "doc.scanNum"
  "Scan number." ] ]
scanNum = element scanNum { xsd:positiveInteger }

[ a:documentation [ xml:id = "doc.subscanNum"
  "Sub-scan number." ] ]
subscanNum = element subscanNum { xsd:positiveInteger }

[ a:documentation [ xml:id = "doc.integrationNum"
  "Integration number." ] ]
integrationNum = element integrationNum { xsd:positiveInteger }

[ a:documentation [ xml:id = "doc.numAPC"
  "Number of data sets for atmospheric phase correction." ] ]
numAPC = element numAPC { xsd:positiveInteger }

[ a:documentation [ xml:id = "doc.execBlock"
  "Execution block identifier; uniquely identifies an instance of " ~
  "each execution of a scheduling block." ] ]
execBlock =
  element execBlock {
    [ a:documentation [ xml:id = "doc.execBlock.href"
      "Reference to containing(?) execBlock." ] ]
    attribute xlink:href { text },
    scanNum,
    subscanNum,
    integrationNum
  }

[ a:documentation [ xml:id = "doc.dataOID"
  "Unique identifier for the integration. This is actually a reference " ~
  "to the data for this integration itself, in essence a " ~
  "self-reference. It will be used by the SDM as a reference to this " ~
  "data." ] ]
```

```
dataOID =
  element dataOID {
    [ a:documentation [ xml:id = "doc.dataOID.href"
      "Reference to the identifier for this integration." ] ]
    attribute xlink:href { text },

    [ a:documentation [ xml:id = "doc.dataOID.title"
      "Description of the data." ] ]
    attribute xlink:title { "Correlator spectral data" }?
  }
```

A.4.2 W3C XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" xmlns:xlink="http://www.w3.org/1999/xlink
  <xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>
  <xs:element name="sdmDataHeader">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="time"/>
        <xs:element ref="dataOID"/>
        <xs:element ref="execBlock"/>
        <xs:element ref="numAntenna"/>
        <xs:element ref="baselineFlags"/>
        <xs:element ref="actualTimes"/>
        <xs:element ref="actualDurations"/>
        <xs:element ref="zeroLags"/>
        <xs:element ref="numAPC"/>
        <xs:element maxOccurs="unbounded" ref="baseband"/>
        <xs:element ref="crossData"/>
        <xs:element ref="autoData"/>
      </xs:sequence>
      <xs:attribute name="byteOrder" use="required">
        <xs:annotation>
          <xs:documentation>Binary data endianness.</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="little endian"/>
            <xs:enumeration value="big endian"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="axisOrder" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="1234567"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="schemaVersion" use="required" type="xs:decimal">
        <xs:annotation>
          <xs:documentation>Schema version number.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
```

```
  </xs:complexType>
</xs:element>
<xs:element name="time">
  <xs:simpleType>
    <xs:union memberTypes="xs:double xs:double"/>
  </xs:simpleType>
</xs:element>
<xs:attributeGroup name="axes">
  <xs:attribute name="axes" use="required">
    <xs:simpleType>
      <xs:restriction>
        <xs:simpleType>
          <xs:list>
            <xs:simpleType>
              <xs:restriction base="xs:token">
                <xs:enumeration value="a1"/>
                <xs:enumeration value="a2"/>
                <xs:enumeration value="a3"/>
                <xs:enumeration value="a4"/>
                <xs:enumeration value="a5"/>
                <xs:enumeration value="a6"/>
                <xs:enumeration value="a7"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:list>
        </xs:simpleType>
        <xs:maxLength value="7"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:attributeGroup>
<xs:element name="baseband">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="spectralWindow"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="spectralWindow">
  <xs:complexType>
    <xs:attribute name="numSpectralPoint" use="required" type="xs:positiveInteger">
      <xs:annotation>
        <xs:documentation>Number of spectral channels.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="numBin" use="required" type="xs:positiveInteger">
      <xs:annotation>
        <xs:documentation>Number of phase bins.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="numPolProduct" use="required">
      <xs:annotation>
        <xs:documentation>Number of polarization products.</xs:documentation>
      </xs:annotation>
      <xs:simpleType>
        <xs:restriction base="xs:integer">
          <xs:enumeration value="1"/>
```

```
          <xs:enumeration value="2"/>
          <xs:enumeration value="4"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="scaleFactor" type="xs:float">
      <xs:annotation>
        <xs:documentation>Scaling factor for spectral data.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
</xs:element>
<xs:element name="crossData">
  <xs:complexType>
    <xs:attribute name="size" use="required" type="xs:positiveInteger">
      <xs:annotation>
        <xs:documentation>Size of binary table. (bytes)</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="float"/>
          <xs:enumeration value="short"/>
          <xs:enumeration value="long"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="ref" use="required" type="xs:NMTOKEN"/>
  </xs:complexType>
</xs:element>
<xs:element name="autoData">
  <xs:complexType>
    <xs:attribute name="size" use="required" type="xs:positiveInteger">
      <xs:annotation>
        <xs:documentation>Size of binary table. (bytes)</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="ref" use="required" type="xs:NMTOKEN"/>
  </xs:complexType>
</xs:element>
<xs:element name="baselineFlags">
  <xs:complexType>
    <xs:attribute name="size" use="required" type="xs:positiveInteger">
      <xs:annotation>
        <xs:documentation>Size of binary table. (bytes)</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="ref" use="required" type="xs:NMTOKEN"/>
    <xs:attributeGroup ref="axes"/>
  </xs:complexType>
</xs:element>
<xs:element name="actualTimes">
  <xs:complexType>
    <xs:attribute name="size" use="required" type="xs:positiveInteger">
      <xs:annotation>
        <xs:documentation>Size of binary table. (bytes)</xs:documentation>
```

```
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="ref" use="required" type="xs:NMTOKEN"/>
    <xs:attributeGroup ref="axes"/>
  </xs:complexType>
</xs:element>
<xs:element name="lut">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:double">
        <xs:attribute name="nbits" use="required" type="xs:positiveInteger">
          <xs:annotation>
            <xs:documentation>Lookup index bitfield length.</xs:documentation>
          </xs:annotation>
        </xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="actualDurations">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" ref="lut"/>
    </xs:sequence>
    <xs:attribute name="size" use="required" type="xs:positiveInteger">
      <xs:annotation>
        <xs:documentation>Size of binary table. (bytes)</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="ref" use="required" type="xs:NMTOKEN"/>
    <xs:attributeGroup ref="axes"/>
  </xs:complexType>
</xs:element>
<xs:element name="zeroLags">
  <xs:complexType>
    <xs:attribute name="size" use="required" type="xs:positiveInteger">
      <xs:annotation>
        <xs:documentation>Size of binary table. (bytes)</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="ref" use="required" type="xs:NMTOKEN"/>
    <xs:attributeGroup ref="axes"/>
  </xs:complexType>
</xs:element>
<xs:element name="numAntenna" type="xs:positiveInteger"/>
<xs:element name="scanNum" type="xs:positiveInteger"/>
<xs:element name="subscanNum" type="xs:positiveInteger"/>
<xs:element name="integrationNum" type="xs:positiveInteger"/>
<xs:element name="numAPC" type="xs:positiveInteger"/>
<xs:element name="execBlock">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="scanNum"/>
      <xs:element ref="subscanNum"/>
      <xs:element ref="integrationNum"/>
    </xs:sequence>
    <xs:attributeGroup ref="xlink:href"/>
```

```
    </xs:complexType>
  </xs:element>
  <xs:element name="dataOID">
    <xs:complexType>
      <xs:attributeGroup ref="xlink:href1"/>
      <xs:attribute ref="xlink:title"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

1 References

[CODSD] J Pisano and F Viallefond, *Correlator Output Data Stream Description.* (2006-03-08).

[RFC2045] N Freed and N Boorenstein, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* . (IETF, 1996-11).

[RFC2046] N Freed and N Boorenstein, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types* . (IETF, 1996-11).

[RFC2387] E Levinson, *The MIME Multipart/Related Content-type* . (IETF, 1998-08).

[RFC2111] E Levinson, *Content-ID and Message-ID Uniform Resource Locators* . (IETF, 1997-03).