Science Data ModelDM Binary Data Format

Version 1.0rc432

2008-03-1861507

Authors:

M. Pokorny, NRAO (EVLA)

J. Pisano, NRAO (ALMA)

1 INTRODUCTION 3

1.1 CONTEXT 3

2 GLOSSARY 5

3 HIGH-LEVEL ORGANIZATION 7

3.1 OVERLOADED TERMS, AND HOW THEY ARE RELATED 9
3.2 XML EXAMPLE 10

4 ENUMERATIONS 12

5 DATA STREAM TYPES 15

6 BINARY COMPONENTS 16

6.1 TYPES 16
6.2 TREE STRUCTURE 16
6.3 AXES 16
6.3.1 Lists 17
6.3.2 Sizes 17
6.3.3 List reduction 20
6.3.4 Element ordering 20
6.4 COMPONENTS 21
6.4.1 Data stream dependencies 24

7 HEADER ELEMENTS 25

7.1 MAIN DATA HEADER 25
7.1.1 Data stream dependencies 31
7.2 DATA SUBSET HEADER 32

8 APPENDICES 36

8.1 XML SCHEMATA 36
8.2 MIME FORMAT EXAMPLE 36
8.3 SCHEMATIC REPRESENTATION OF TIMES 37
8.4 ALMA IMPLEMENTATION DETAILS 37
8.4.1 Spectral data stream 37
8.4.1.1 Binary component sizes 37
8.4.1.2 Example of Full Resolution Data 40
8.4.1.3 Data rate 42
8.4.2 Channel average data stream 42
8.4.2.1 Binary component sizes 42
8.4.2.2 Example of Channel Average Data 45
8.4.2.3 Data rate 47
8.4.3 Data capture information 47
8.4.4 Data transmission 48
8.4.4.1 Overview 48
8.4.4.2 Subscan start 48
8.4.4.3 Sending data 48
8.4.4.4 Subscan end 48

8.5 EVLA WIDAR IMPLEMENTATION 49
8.5.1 Data stream example *49*
8.5.2 Data rates *49*
8.6 REFERENCES 50

1 Introduction

The format described in this document is part of the Science Data Model (SDM) being used by the ALMA and EVLA projects. Each application (*e.g.*, ALMA-B and ACA correlator data, ALMA total power data, EVLA WIDAR correlator data) is expected to implement only a part of the format described herein, and in that sense, this format is a generalization of the specific formats used by all applications. The generic SDM binary format, and, consequently, its application-specific variants, are all supported by the CASA post-processing software through a CASA-provided measurement set filler.

Where necessary, notations in this document are used to provide application-specific implementation features or details related to the subject under description.

An SDM dataset is composed of a set of tables with specified in-memory representations, and representations as XML documents for data exchange and persistence. The amount of actual data produced by various processors (correlators, square law detectors, radiometers, *etc.*) being in general very large, data from these sources are stored in binary blocks with unique labels for reference from the SDM tables. These data, comprising the actual data (cross-correlations, auto-correlations), auxiliary data (zero-lags) and associated meta-data (flags, *etc.*), are grouped into binary large objects (BLOBs). Each processor produces data sets (integrations and sub-integrations) ~~dumps~~ sequentially in time as the observations proceed, and the data flow is fed by a set of processors producing data concurrently. A single BLOB may consist of a time series of integrations ~~dumps~~, or, alternatively, a more complex object when the time axis is part of the data structure. The former case is driven by the need to support data streaming; the latter case, by the need to provide efficient storage with minimal overhead for those processors that regularly produce smaller “chunks” of data, although possibly at a moderately high rate (*e.g.*, the data produced by ALMA's baseband-wide continuum detectors).

Although the data format described in this document could support rather complex data organizations, in practice, for both ALMA and EVLA, we are considering one BLOB per processor per sub-scan. For the ALMA-B and ACA correlators the finest temporal granularity of the SDM data is a sub-integration, whereas for the EVLA WIDAR correlator the finest granularity is an integration. Therefore, for the purposes of this document, a BLOB always contains data for the set of (sub-)integrations comprising a sub-scan for a given set of antennas *(i.e.*, subarray).

The specification of the data format as described by this document is, in itself, incomplete. A complete specification is provided by the present document together with the associated XML schemata (see section 8.1). All of the documents required for a complete specification are maintained as a single project under a revision control system. In an attempt to maintain a somewhat higher level of description in the present document, some of the low-level details of the format that are specified by the XML schemata are not provided here. An implementation that reads or writes data in this format will necessarily require information about the XML parts of the format that are best provided by the XML schemata.

> *ALMA note*
>
> This initial version of this document accurately describes the data produced by the ALMA-B correlator. The ACA correlator is not fully described by this document, but will be in the next revision of this document which is due in mid-May 2008.

1.1 Context

Here we present a short description of the context in which this data format is used by each of the applications.

EVLA WIDAR note

The correlator output binary data format applies to the output data stream of the WIDAR correlator backend. The data produced in the binary data format are primarily spectral in nature, although supplemental, non-spectral data are also a part of the format. Typically, the data stream is sent to the EVLA archive and TelCal, while metadata are sent to MCAF 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, correlator backend 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 (the fast formatter node will also provide data to MCAF, but those data are 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.

ALMA note

The ALMA-B correlator produces lag results which are transferred (or *dumped*) to the *CDP (Correlator Data Processor) Node* computers via physical connections. The dump interval is programmable. The *CDP nodes* process the lag data into raw spectra which are then transferred to the *CDP Master* computer at each integration or sub-integration – there are an integral number of sub-integrations per integration. The *master* computer organizes these raw spectra according to the binary data format described in this document and transfers them to a data distributor via a network connection to the *Archive*, *TelCal*, and *QuickLook Pipeline*.

The ACA correlator software functions in a similar mode, except that ACA correlator is an FX type generating spectra as its raw output and that the dump interval is non-programmable.

Several other data products are produced along the signal path which are not generated by the ALMA correlators. The details of these data products are not completely defined in this document. These data products include total power from the baseband/IF processors, water vapor radiometry data, and holography data.

2 Glossary

We define terms used in this document. Note that these definitions may not be exactly the same as the definitions used by the ALMA and EVLA projects in other situations.

channel aAverage

A group of spectral channels averaged together over a time interval resulting in a single value.

data set

The contents of a single document in the binary data format.

dumpDump

The acquisition of data from the correlator corresponding to a dump duration (see below).The basic unit of data produced by a data processor in the binary data format.

dump Duration The smallest interval of time for which a set of correlated data can be accumulated and output from the correlator.

integration

A data *dump* from a correlator **A set of dumps , all identical in configuration (except for the antenna motion and some others), that is accumulated and forms the basic recorded unit.**The basic unit of data produced by a data processor in the binary data format at the highest available spectral resolution. Generally exists at a lower time resolution and higher spectral resolution than a sub-integration.

scan

A sequence of *sub-scans*. The concept of "scan" is not a part of the binary data format aside from the occurrence of opaque references to scans defined in the SDM.

spectral window A segment of adjacent frequencies within a baseband which have a unique correlator configuration and are processed independently of other frequencies in the baseband. A baseband can have multiple spectral windows.subarray

The complete set of antennas producing the data of a *data set* in the binary data format.

sub-integration

The time interval of processing of channel average dataA dumpdata set of channel-averaged data from a correlator. Generally exists at a higher time resolution and lower spectral resolution than an integration.

sub-scan

A sequence of data sets ***dumps*** from a correlator *(i.e.*, integrations or sub-integrations) comprising a single *data set* in the binary data format.

URI

Uniform Resource Identifier; a sequence of characters that identifies an abstract or physical resourcea.

~~zero lags For the ALMA-B correlator, this is the real correlation function for antenna and polarization self-products used to obtain the original correlation value as the ALMA correlator software normalizes correlation functions to unity.~~

3 High-level organization

In the binary data format, the data are organized hierarchically. While there is considerable flexibility in the format, header fields are sufficient to describe the organization of any instance of the format. The data are organized at the top level using the MIME (Multipurpose Internet Mail Extensions) format. In the application of the MIME format to the binary data format, it provides a hierarchically organized container for data headers in XML format and binary data sections.

Although other applications of the format may vary, as an illustrative example, this document will describe the high-level organization of the data as it is used for the ALMA and EVLA correlators. The binary data format supports applications using an organizational hierarchy with greater depth than is currently used by the aforementioned correlators, but no such applications are known to exist, and the format lacks some specification to fully support this feature. Applications with a lesser hierarchy are supported fully (*e.g.*, the ALMA total power data), but are not yet illustrated with examples in this document. The specification of the hierarchy used to organize the data in any instance of the binary data format is provided by the `dimensionality` and/or `numTimes` header elements (which are described in section 7.1).

For the ALMA and EVLA correlators, a BLOB in the binary data format comprises spectral data blocks for a single sub-scan. Each sub-scan contains all the spectral data for all integrations for all baselines of a given set of antennas, all bins, all polarizations, and APC (Atmospheric Phase Correction) data sets.

The parts of a MIME message in the binary data format are of three types: the main data header, the data subset headers, and the binary data components. The main data header contains general information about the sub-scan, and all properties shared by the (sub-)integrations. The subset headers contain information specific to each integration or sub-integration in the sub-scan. The binary data for the (sub-)integration, grouped into several components, follow each subset header. Schematically, the data are organized as follows (for the sub-scan-based grouping used by the ALMA and EVLA correlators).

Data container (sub-scan)

Main data header (sub-scan-wide metadata)

Data subset container 1 (first (sub-)integration)

Data subset header ((sub-)integration metadata)

Binary component 1.1

Binary component 1.2

…

Binary component $1.n_1$

Data subset container 2 (second (sub-)integration)

Data subset header ((sub-)integration metadata)

Binary component 2.1

…

Binary component $2.n_2$

…

Data subset container N (N^{th} (sub-)integration)

Data subset header ((sub-)integration metadata)

Binary component N.1

…

Binary component $N.n_N$

Note that, in general, each additional level of hierarchy in the structure of the data would introduce, in a nested fashion, an equivalent of the "data subset containers" shown in the example above.

The schematic view represented above can be converted into the structure of a MIME message by replacing each "container" with a multipart MIME message. We do not reiterate here all of the relevant MIME specifications; instead we describe the application of the MIME standards to the binary data format, noting only what is specific to this application, including any exceptions to the MIME standards that are allowed under this application. Further information regarding the MIME standards may be found in references [RFC2045], [RFC2046], [RFC2387], and [RFC2557]. With the overall MIME structure of the data format as background, we note the following additional specifications for MIME messages that conform to the binary data format.

- The `Content-Type` header value of the top-level MIME message is `multipart/mixed`.
- The `Content-Type` header value of the MIME messages corresponding to the data subset containers is `multipart/related`.
- The top-level MIME `Content-Description` header value is of the form *telescopeName/processorType/processorName/spectralResolution*.

> *EVLA WIDAR note*
>
> The top-level MIME `Content-Description` header value is EVLA/CORRELATOR/WIDAR/FULL_RESOLUTION.

ALMA note

For ALMA, we have *telescopeName* = ALMA, *processorType* = CORRELATOR, *processorName* = ALMA_ACA *or* ALMA_BASELINE *or* ALMA_BASELINE_ATF *or* ALMA_BASELINE_PROTO_OSF, *spectralResolution* = CHANNEL_AVERAGE or FULL_RESOLUTION

- To preclude scanning through large chunks of binary data before generating a MIME boundary string, sequences of bytes that duplicate a MIME boundary string *are* allowed to occur in the parts of the MIME message with a `Content-type` of `application/octet-stream`. Because the sizes of these parts can be determined from the data headers, true MIME boundary strings can be differentiated from sequences of bytes in the binary parts that happen to match a boundary string. Clearly, with this relaxation of the MIME standard the BDF documents cannot be guaranteed to conform to the MIME standard, and special applications are required to read the binary format reliably; generic MIME applications could, in theory, fail to parse these messages correctly. In practice, if the boundary strings are long, the chances of a string appearing in a binary component with the same characters will be small, and the risk of potential problems will be small. All other standard requirements on MIME boundary string values apply to this format.
- The `Content-Location` header value of the top-level MIME message is the base URI for the resolution of relative URIs in the descendant MIME parts. The value is effectively the data object identifier (dataOID) of the data in the MIME message.
- The `Content-Location` header value of the MIME parts for the main and subset data headers is the relative URI of the header (with respect to the base URI given in the `Content-Location` header value of the top-level MIME message). The value consists of the relative project path for the data described by the header and the term "desc.xml", joined by a "/" character. The relative project path for the main data header comprises the execution block number, the scan number and the sub-scan number of the data in the MIME message, wherein these numbers are joined by "/" characters. An example of a complete `Content-Location` value for a main data header is `1/10/3/desc.xml`. For the subset data headers, the relative project path begins with the relative project path in the main data header, and has appended to it a (sub-)integration number. An example of a complete `Content-Location` value for a subset data header is `1/10/3/2/desc.xml`.
- The `Content-Location` header value of each of the MIME parts for the binary components is similarly the relative URI for the binary component. The trailing part of each of these URIs is a name derived from the type of the binary component together with a ".bin" suffix, for example, `1/10/3/2/actualDurations.bin`. The name in the header value is that of the subset header element associated with the binary component type as described in section 6.4.

Snippets of MIME messages that conform to the data format can be found in the appendices.

3.1 Overloaded terms, and how they are related

Several terms in the description of this format are used repeatedly, but with different meanings, to reflect the logical relationships between various elements of the data format. Because this overloading of terms can be confusing, we attempt to describe the relationships at a high level in this section. To that end, we will use a single example based on the flagging data that is recorded in a data set. The actual flagging data

appear in a binary component of the FLAGS type. An XML element in the main data set header, named `flags`, provides metadata that (partially) defines the shape all the FLAGS-type binary components that are in the data set.[1] Finally, in each data subset header, there appears an XML attribute named `flags` that provides a link to the actual flagging data recorded in that data subset.

Associated with each type of binary data in the binary data format, there are three uses of a single term:

- an all-uppercase version *e.g.*, FLAGS (a term that never appears in the data sets themselves) is used to indicate a type of binary component;
- an XML element in the main data header, *e.g.*, `flags`, is used to define the shape of the instances of the binary component type in the data set; and
- an XML attribute in the subset data header, *e.g.*, `flags`, is used to identify an instance of that binary component type in the subset.

3.2 XML Example

Complete examples can be found in the appendices, but a simple example of the XML headers serve as a guide for detailed descriptions of the XML elements and structure. Note that MIME boundaries are removed.

```
<?xml version="1.0" encoding="UTF-8"?>
<sdmDataHeader
  xmlns="http://TBD/XSDM/sdmbin"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xvers="http://TBD/XVERSION"
  xsi:schemaLocation="http://TBD/XSDM/sdmbin 1/sdmDataObject.xsd"
  xvers:schemaVersion="0"
  xvers:revision="0.0.96"
  mainHeaderId="sdmDataHeader"
  byteOrder="IEEE_LoLittlew_Endian"
  projectPath="3/1/2/">
  <startTime>4647257068000000000</startTime>
  <dataOID xlink:type="locator"
    xlink:href="uid://X1/1/0/0"
    xlink:title="ALMA BL Correlator Spectral Data"/>
  <dimensionality axes="TIM">1</dimensionality>
  <execBlock xlink:href="uid://X1/1/0/1" xlink:type="simple"/>
  <numAntenna>2</numAntenna>
  <correlationMode>CROSS_AND_AUTO</correlationMode>
  <spectralResolution>FULL_RESOLUTION</spectralResolution>
  <dataStruct xsi:type="CrossAndAutoDataFullResolution" apc="AP_UNCORRECTED">
    <baseband name="BB_1">
      <spectralWindow id="spw_2" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7680" numBin="1" sideband="NOSB"/>
    <flags size="6" axes="BAL ANT BAB"/>
    <actualTimes size="6" axes="BAL ANT BAB"/>
    <actualDurations size="6" axes="BAL ANT BAB"/>
    <crossData size="122880" axes="BAL BAB SPW SPP"/>
    <autoData size="122880" axes="ANT BAB SPW SPP"/>
    <zeroLags size="16" axes="BAL BAB SPW"/>
  </dataStruct>
</sdmDataHeader>
```

```
<sdmDataSubsetHeader
  xsi:type="BinaryCrossAndAutoDataFullResolution"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  projectPath="3/1/2/1/">
  <schedulePeriodTime>
```

[1] The complete definition of the shapes of the binary components depends in general on the complete set of elements that are present in the main data set header.

```
    <time>4647257073120000000</time>
    <interval>10240000000</interval>
  </schedulePeriodTime>
  <dataStruct ref="sdmDataHeader"/>
  <flags xlink:href="3/1/2/1/flags.bin"/>
  <actualTimes xlink:href="3/1/2/1/actualTimes.bin"/>
  <actualDurations xlink:href="3/1/2/1/actualDurations.bin"/>
  <crossData xlink:href="3/1/2/1/crossData.bin" type="SHORT_TYPE"/>
  <autoData xlink:href="3/1/2/1/autoData.bin"/>
  <zeroLags xlink:href="3/1/2/1/zeroLags.bin"/>
</sdmDataSubsetHeader>
```

4 Enumerations

Several enumerations are used by the data format to provide a semantic basis for certain terms, and to create a bridge to the enumerations used in the other components of the SDM. Enumeration values have both string and integer representations. In the table below, enumeration values are presented in their proper order; however, the integer values are not specified, because they are of no direct use.

Enumeration type	Enumeration values	Description
AtmPhaseCorrection	AP_UNCORRECTED	uncorrected for atmospheric phase
	AP_CORRECTED	corrected for atmospheric phase
	AP_MIXED	mixture of baselines with and without atmospheric phase correction
AxisName	TIM	time
	BAL	baseline
	ANT	antenna
	BAB	baseband
	SPW	spectral window
	BIN	bin (e.g., pulsar phase, frequency switching)
	APC	atmospheric phase correction bin
	SPP	spectral point (channel)
	POL	polarization
	HOL	holography
BasebandName	BB_1	baseband 1
	BB_2	baseband 2
	BB_3	baseband 3
	BB_4	baseband 4
	BB_5	baseband 5
	BB_6	baseband 6
	BB_7	baseband 7
	BB_8	baseband 8
CorrelationMode	CROSS_ONLY	cross-correlation data only
	AUTO_ONLY	auto-correlation data only
	CROSS_AND_AUTO	both cross-correlation and auto-correlation data

CorrelatorName	ALMA_ACA	
	ALMA_BASELINE	
	ALMA_BASELINE_ATF	
	ALMA_BASELINE_PROTO_OSF	
	EVLA_WIDAR	
	HERSCHEL	
	IRAM_PDB	
	IRAM_30M_VESPA	
PrimitiveDataType	INT16_TYPE	16-bit signed integer
	SHORT_TYPE	IDL short, XMLW3C short, C/C++ short, Java short
	INT32_TYPE	32-bit signed integer
	INT_TYPE	IDL int, XMLW3C int, C/C++ int, Java
	INT64_TYPE	64-bit signed integer
	LONGLONG_TYPE	64-bit signed integer
	FLOAT32_TYPE	IEEE 754 single format floating point number (32 bits)
	FLOAT64_TYPE	IEEE 754 double format floating point number (64 bits)
ProcessorType	CORRELATOR	digital correlator
	RADIOMETER	radiometer
	SPECTROMETER	multi-channel spectrometer
NetSideband	NOSB	no sideband
	USB	upper sideband
	LSB	lower sideband
	DSB	double sideband
SpectralResolutionType	FULL_RESOLUTION	high spectral resolution data
	CHANNEL_AVERAGE	lower spectral resolution data
	BASEBAND_WIDE	data applying to the whole baseband

StokesParameter	RR	right-right polarization product
	RL	right-left polarization product
	LR	left-right polarization product
	LL	left-left polarization product
	XX	X-X polarization product
	XY	X-Y polarization product
	YX	Y-X polarization product
	YY	Y-Y polarization product

Table 1: Enumerations

> *ALMA note*
>
> The ALMA-B and ACA correlators currently do not use the *CorrelatorName* enumeration.
>
> *NetSideband* is used by exclusively by ALMA as it has single-sideband, two-sideband and dual-sideband receivers. Receiver type information is used in the BDF to assist in the identification of which spectral windows have which sideband data.
>
> *Holography* refers to single dish holography which uses a reference horn on a single antenna and produces bandwidth data. The holography data format is not defined in this document.

5 Data stream types

The binary data are classified according to the "mode" under which the data processor has produced the data. The binary data format supports a concept of mode under which the data format varies slightly according to this mode. Because the data formats are somewhat different for the different modes, we classify a sequence of files (*i.e.*, MIME messages) in the binary data format produced under a given mode (by a single data processor) by the term *data stream.* A data stream is parameterized by two values, one of type *CorrelationMode*, and one of type *SpectralResolutionType.* Any given data processor will likely produce data in only a small subset of the ~~nine~~ twelve possible types of data streams.

The specification of a data stream type affects the binary data format by restricting the set of allowed binary components, as well as their structures. Effects are also present in the main data header and the data subset headers as a consequence.

> *EVLA WIDAR note*
>
> The EVLA WIDAR correlator backend produces a single data stream per subarray. For this stream, the *CorrelationMode* value is CROSS_AND_AUTO, and the *SpectralResolutionType* value is FULL_RESOLUTION. Note that the data stream may be smoothed or averaged by the correlator backend — setting the *SpectralResolutionType* to FULL_RESOLUTION is done simply to match the ALMA convention, where this means "highest *available* spectral resolution".

> *ALMA note*
>
> For ALMA, *CorrelationMode* = CROSS_ONLY is not allowed.

To highlight the dependencies in the format on the data stream type, small grids filled with one-letter annotations are used whenever some aspect of the format that varies with data stream type is being described. These grids appear in the left margin of the page, adjacent to the presentations of elements, attributes, and components as needed. The following table shows the structure of these marginal grids; the grids themselves appear without row headers, column headers, or grid lines.

		SpectralResolutionType			
		FULL_RESOLUTION (F)	CHANNEL_AVERAGE (C)	BASEBAND_WIDE (B)	*undefined (U)*
CorrelationMode	CROSS_ONLY (C)	M/O/X	M/O/X	M/O/X	M/O/X
	AUTO_ONLY (A)	M/O/X	M/O/X	M/O/X	M/O/X
	CROSS_AND_AUTO (B)	M/O/X	M/O/X	M/O/X	M/O/X

Table 2: Key to format variation marginal grids. Symbols: M - mandatory, O - optional, X – excluded. The letters in parentheses are the abbreviated forms of the table headers used in the marginal grids.

F C B U
C O O O O
A X X X X
B M M O O
EXAMPLE

For example, the grid in the margin to the left shows that the element named "EXAMPLE" is optional whenever the *CorrelationMode* value of the data stream is CROSS_ONLY, or the *CorrelationMode* value is CROSS_AND_AUTO and the *SpectralResolutionType* value is undefined or has the value BASEBAND_WIDE; excluded (that is, prohibited) whenever the *CorrelationMode* value is AUTO_ONLY; and mandatory whenever the *CorrelationMode* value is CROSS_AND_AUTO and the *SpectralResolutionType* value is either FULL_RESOLUTION or CHANNEL_AVERAGE.[2]

[2] A format change to be adopted soon will define `zeroLags` for both full resolution and channel average, so the *SpectralResolutionType* will become meaningless and will be dropped.

6 Binary components

6.1 Types

Seven types of binary components are supported by the binary data format. The set of binary component types is meant to be extensible, but for the purposes of the present version of the format described in this document, the following list shall be considered exhaustive. The descriptions in the following list of binary component types are merely brief summaries; for more detailed information, refer to section 6.4.

FLAGS (metadata)

: flagging information

ACTUAL_TIMES (metadata)

: time centroid associated with data

ACTUAL_DURATIONS (metadata)

: total amount of time used in deriving the data

WEIGHTS (metadata)

: data weights

ZERO_LAGS (auxiliary data)

: zero lags

CROSS_DATA

: cross-correlation data

AUTO_DATA

: auto-correlation data

> *EVLA WIDAR note*
>
> The EVLA correlator backend is able to produce all of the above binary component types.

> *ALMA note*
>
> The ALMA CDP does not produce WEIGHTS. The ACA correlator produces neither WEIGHTS nor ZERO_LAGS. This document defines ZERO_LAGS as required elements, for this reason, this version does not accurately define the ACA correlator output.

6.2 Tree structure

The data in the binary components are organized in a tree structure, in which the trees are of uniform depth, and the nodes are ordered. Data elements are associated with the leaf nodes of the tree, and the in-memory and persistent representations of the data correspond to an in-order traversal of the leaf nodes of the tree. Each level of the tree corresponds with a data *axis* along which the data are indexed. Each axis in an ordered list corresponds with a level in the tree structure of the binary data, the levels descending the tree as the axis list is traversed in order.

6.3 Axes

The axes that can be used to index the binary data components are listed in Table 1 under the *AxisName* enumeration type. Note that the order of the axes given in Table 1 is significant for the binary components; although not all axes are required for a binary

component, the axes that are used must be ordered as given by the table. For example, if the N^{th} level of a tree is indexed by the SPW axis, the $N+1^{st}$ level may be indexed by any one of the BIN, APC, SPP, or POL axes, but not the TIM, BAL, ANT or BAB axes. The axis ordering therefore imposes a hierarchy on the data structure of all binary components. As an example, the number of spectral windows may depend on the baseband, but not on the set (or list) of polarizations.

> *EVLA WIDAR note*
>
> The bin axis, BIN, is used primarily for pulsar phase binning.
>
> The atmospheric phase correction axis, APC, initially will have a size of one; the single element having a value of AP_UNCORRECTED.
>
> *ALMA note*
>
> The bin axis is used for nutator or frequency switching.

An important exception to the hierarchical nature of the axes named by the *AxisName* enumeration type is that the BAL and ANT axes are at the same level. This means that ~~the~~ when the sequence "BAL, ANT" is given in a list of axes: 1) the ANT nodes are to be considered as equivalent to a baseline of length zero; 2) the "true" baseline-based (*i.e.*, non-zero baseline length) data appear before the antenna-based data when traversing the tree nodes in order; and 3) both types of nodes appear at a single level of the tree. Note that the "BAL, ANT" axis may only be used with the metadata binary components.

6.3.1 Lists

The structure of a binary component is thus greatly dependent upon the axes used for indexing the data in the component. Metadata for each binary component specifies an ordered list of axes along which the data are indexed. Applications of this format may restrict the set of allowed axes lists for the various binary components, but such restrictions are not a part of the format specification.

6.3.2 Sizes

The size (or length) of an axis is equal to the number of children of a node at the level corresponding to that axis. For a generic tree, each node at any given level could have a different number of children, but in the binary data format, the number of children at various nodes in a tree may be constrained.

The sizes of the axes of a binary component tree are specified in various ways, as follows.

- TIM: Size is set by the `numTimes` element of the main data header, if present; otherwise, the axis size is one.

> *EVLA WIDAR note*
>
> The size of the TIM axis is *one* in the EVLA WIDAR format (for sub-band cross- and auto-correlations), meaning that each data subset contains data associated with a single time (that being all the data within one integration interval).
>
> *ALMA note*
>
> The size of the TIM axis is *one* in the ALMA-B and ACA correlator format (for sub-band cross- and auto-correlations), meaning that each data subset contains data associated with a single time (that being all the data within one integration or sub-integration interval).

- BAL: Size is determined by the value of the `numAntenna` element of the main data header. If the value of `numAntenna` is represented by N_a, the size of this axis is $N_a(N_a-1)/2$.

- ANT: Size is set by the numAntenna element of the main data header. Note that, as defined in section 6.3.4, the unified "BAL+ANT" axis has a size of $N_a(N_a+1)/2$, where N_a is equal to the value of numAntenna.
- BAB: Size is equal to the number of baseband elements in the main data header.
- SPW: Size is equal to the number of spectralWindow elements within a baseband element (hence, the axis size may vary with baseband).
- BIN: Size is set by the numBin attribute of a spectralWindow element (hence, the axis size may vary with spectral window).

ALMA note

numBin is common to all spectral windows defined in a baseband.

- APC: Size is equal to the number of enumerators appearing in the (list) value of the apc attribute of the dataStruct element of the main data header.
- SPP: Size is set by the numSpectralPoint attribute of a spectralWindow element of the main data header (hence, the number of spectral points may vary with spectral window).
- POL: ~~For AUTO_DATA and CROSS_DATA binary components, the axis size is equal to the number of enumerators in the (list) value of the sdPolProduct and crossPolProduct attributes, respectively, of a spectralWindow element. For the FLAGS binary component, the size of this axis depends on whether the flagging is done on the basis of polarizations or polarization products. In the case of polarization-based flagging, this means that if the sdPolProduct or crossPolProduct lists have more than two items, the POL axis size is two, otherwise, it is one. For WEIGHTS, ACTUAL_TIMES and ACTUAL_DURATIONS binary components, the size of the axis can be 1, 2, 3 or 4.~~Size varies depending on the binary component type, and whether an ANT or BAL axis appears at higher levels of the tree (and in the case of a combined BAL+ANT axis, it depends on whether a baseline or an antenna is being indexed). The size of the POL axis in the metadata and auxiliary data components largely depends on the size of the POL axis in the AUTO_DATA and CROSS_DATA components; the AUTO_DATA component being controlling for ANT axis values, and the CROSS_DATA component, for BAL axis values.

Component type		**POL axis size**	**Notes**
AUTO_DATA		1, 2, or 3	Equal to length of sdPolProduct list of spectralWindow element.
CROSS_DATA		1, 2 or 4	Equal to length of crossPolProduct list of spectralWindow element.
FLAGS	antenna	1 or 2	Size depends on whether flags are recorded for an ANT or BAL axis value.
	baseline	1, 2 or 4	
ACTUAL_TIMES	antenna	1, 2, or 3	Size depends on whether times are recorded for an ANT or BAL axis value.
	baseline	1, 2, or 4	
ACTUAL_DURATIONS	antenna	1, 2, or 3	Size depends on whether durations are recorded for an ANT or BAL axis value.
	baseline	1, 2, or 4	
WEIGHTS	antenna	1, 2, or 3	Size depends on whether weights are recorded for an ANT or BAL axis value.
	baseline	1, 2, or 4	
ZERO_LAGS		1 or 2	Equal to number of parallel-hand products in sdPolProduct list of spectralWindow element

~~ZERO_LAGS~~	

Because the size of the POL axis for the AUTO_DATA and CROSS_DATA components is set at the SPW level, when a metadata or auxiliary data component is indexed at the BAB, but not the SPW, level, the union of the polarization products in all of the spectral windows in the baseband must be used to determine the size of the POL axis. For example, consider a FLAGS component that has BAL, BAB and POL axes. Assume that one of the basebands has exactly two spectral windows: one with `crossPolProduct`="XX", and one with `crossPolProduct`="YY". In this case, the POL axis for that baseband in the FLAGS component must have a size of two. In contrast, if the FLAGS component axes are BAL, BAB, SPW and POL, then the size of each POL axis in the baseband is one.

6.3.3 List reduction

Generally, a tree can always be indexed with its complete set of axes. However, when the size of an axis is one (in all tree branches), and the logical node value on that axis is constant in all tree branches, that axis may be omitted from the list of axes. Reducing the axis list in this way has no impact on either the number of elements in the tree, or on the structure of elements in the tree. The only effect of axis list reduction is on the description of the tree. Note the requirement that for list reduction the logical node value on the axis must be constant on all tree branches; this means that, for example, if on one tree branch the POL axis has data for the single value XX, no other tree branch has data on the POL axis for a different value of the *StokesParameter* enumeration.

6.3.4 Element ordering

The ordering of index values along the axes depends on the axis. For the axes that are described below as having "integer order", a mapping from the integer to and from some externally meaningful value is provided by the SDM.~~Ordering of index values along an axis depends on the axis as follows.~~

~~ANT~~

~~Integer order (*e.g.*, 1, 2, 5, 10, 12)~~

> ~~*ALMA note*~~
>
> ~~ALMA antennas are alphanumeric strings and ordered accordingly, e.g., A01, A02, A10, D1, D3.~~

BAL

In a matrix formed by labelling rows and columns with antenna indexes, and where each matrix element is the pair (row antenna index, column antenna index), the ordering of the BAL axis corresponds to the column-major traversal of the matrix's upper triangle. This ordering allows baselines introduced through the addition of a new antenna to appear at the end of the list.

For example, for antennas in the set {1, 2, ~~4, 7~~3, 4}, the order of baselines is (1, 2), (1, ~~4~~3), (2, ~~4~~3), (1, ~~7~~4), (2, ~~7~~4), (3~~4~~, ~~7~~4).

Note that for the special case in which the axis list contains the sequence "BAL, ANT", in the previous example, the complete list of antenna "pairs" along the unified "BAL, ANT" (or "BAL+ANT") axis is (1, 2), (1, ~~4~~3), (2, ~~4~~3), (1, ~~7~~4), (2, ~~7~~4), (~~4~~3, ~~7~~4), (1, 1), (2, 2), (~~4~~3, ~~4~~3), (~~7~~4, ~~7~~4).

Finally, note that the order of the correlation products recorded in the binary must agree with the order of antennas in each baseline. For example if the baseline is (1, 2), the product must be 1 * 2 (*i.e.*, the correlation of antennas 1 and 2, in that order), not 2 * 1.

ANT

Integer order. Index values map to antenna labels or names as defined in the SDM.

BAB

Order of *BasebandName* type values (see Table 1).

SPW

Order of `spectralWindow` elements within a `baseband` element in the main data header.

> *ALMA note*
>
> Both ALMA-B and ACA correlators share the following constraint:
>
> Two spectral windows should be defined contiguously if these spectral windows are in the image sideband to each other. LSB first and USB follows in that case.
>
> The ACA correlator has following constraint on the order of SPW:
>
> The spectral windows should be arranged in ascending order of the first channel of the spectral window in terms of 3.8 KHz channels.

BIN

Integer order. Index values are mapped in the SDM to a type of quantity that depends on the use of the BIN axis in the BDF.

> EVLA WIDAR note
>
> Bins are used for phase binning in the WIDAR correlator; the SDM provides the mapping from bin index to/from phase.

> *ALMA note*
>
> *Bins are used for nutator or frequency switching*

APC

Order of *AtmPhaseCorrection* type values (see Table 1).

SPP

Integer order. Index values map to frequencies or lags as defined in the SDM.

> *EVLA WIDAR note*
>
> The WIDAR correlator can produce either frequency or lag spectra. The conversion from integer SPP axis values to frequency or lag number is provided by the SDM metadata.

> *ALMA note*
>
> For upper sidebands, the frequency order ~~decreases~~increases with increasing channel number and for lower sidebands, the frequency order ~~increases~~decreases with increasing channel number.

POL

Order of *StokesParameter* type values (see Table 1).

6.4 Components

The data in each binary component type are described below in terms of the quantity represented, the units used to express that quantity, and ~~and~~ the type used to represent the data. Additionally, for each binary component type, the name of the element in the main or subset data header that provides metadata for the binary component is identified. ~~Finally, because~~ Also, because the presence or absence of a binary component is meaningful, a description of such meaning is ~~also~~ given where relevant. Any restrictions on the allowed set of axes that may be present in a component are noted, as well.

F C B U
C O O O O
A O O O O
B O O O O
FLAGS

FLAGS

- datum: occurrence of flag conditions
- units: N/A, bitfield
- data type: 32 bit integer (INT32_TYPE)
- header element: `flags`
- Absence of this component means that no flag conditions occurred during data acquisition.
- The occurrence of a flagged condition is represented by a one in the bit position reserved for that condition. A zero-valued bit indicates that the condition has not been flagged.
- ~~Absence of this component means that no flag conditions occurred during data acquisition. Note that F~~the ~~f~~lag conditions are application specific.

F C B U
C O O O O
A O O O O
B O O O O
ACTUAL_TIMES

ACTUAL_TIMES

- datum: centroid of (MJD) time interval associated with data subset, allowing for blanking
- units: nanoseconds
- data type: 64 bit integer (INT64_TYPE)
- header element: `actualTimes`
- If this component is absent, then no data blanking has occurred during the time interval, and the time stored in the data subset header (by the `time` element) is to be used for the time associated with the data subset.
- The time basis (UTC, IAT, local, *etc.*) is defined in the SDM, and the times recorded in this component are expressed in that basis.

F C B U
C O O O O
A O O O O
B O O O O
ACTUAL_DUR'NS

ACTUAL_DURATIONS

- datum: total amount of time used in deriving the data subset, allowing for blanking (sometimes called “exposure”). Proportional to the number of valid samples.
- units: nanoseconds
- data type: 64 bit integer (INT64_TYPE)
- header element: `actualDurations`
- If this component is absent, then no data blanking has occurred during the time interval, and the duration stored in the data subset header (by the `interval` element) is to be used for the exposure time associated with the data subset.

F C B U
C O O O O
A O O O O
B O O O O
ZERO_LAGS

ZERO_LAGS

- datum: lag zero value (real-valued)
- data type: single format IEEE 754 floating point number (FLOAT32_TYPE)
- header element: `zeroLags`
- ANT axis is mandatory; BAL and BAL+ANT axes are prohibited
- The data values are always real-valued because only the parallel-hand polarization products are present in this binary component type.

> *ALMA note*
>
> For the ~~ALMA-BL~~ALMA-B correlator, data are always present for FULL_RESOLUTION and CHANNEL_AVERAGE. The ACA correlator does not produce ZERO_LAGS.

> *EVLA WIDAR note*
>
> The WIDAR correlator may use zero lag values for scaling the auto-correlation data.

F C B U
C X X X X
A M M M M
B M M M M
AUTO_DATA

AUTO_DATA

- datum: auto-correlation value (real-valued for parallel polarization products or complex-valued for cross polarization products)
- data type: 1 or 2 * single format IEEE 754 floating point number (FLOAT32_TYPE)
- header element: `autoData`
- The choice of a real- *vs.* complex-valued datum is dependent upon the polarization product that is recorded; parallel-hand polarizations are real-valued, while cross-hand polarizations are complex-valued. The `sdPolProducts` attribute of the `spectralWindow` elements may be used to determine the polarization products and their order.
- ANT axis is mandatory; BAL and BAL+ANT axes are prohibited

F C B U
C M M M M
A X X X X
B M M M M
CROSS_DATA

CROSS_DATA

- datum: cross-correlation value (complex-valued)
- data type: 2 * single format IEEE 754 floating point number (FLOAT32_TYPE), integer (INT_TYPE), or short integer (SHORT_TYPE) (specified in the data subset header `crossDataType` element)
- header element: `crossData`
- BAL axis is mandatory; ANT and BAL+ANT axes are prohibited

> *EVLA WIDAR note*
>
> Initially EVLA will use 2 * FLOAT32_TYPE.

> *ALMA note*
>
> ALMA uses ~~scaled 16- or 32-bit signed integers~~SHORT_TYPE or INT_TYPE. See [Scott].

F C B U
C O O O O
A O O O O
B O O O O
WEIGHTS

WEIGHTS

- datum: data weight lookup table index (lookup table is located in `weights` element of main data header)
- data type: variable, a word of N bits to encode (one of 2^N) –lookup table index values (the value of N is determined by counting the number of entries in the

lookup table defined by the `weights` header element)

- header element: `weights`
- The size of this binary component is always an integer number of eight-bit bytes, which means that pad bits may appear at the end of the component to ensure that the total size of the component meets this constraint. The following formulas can be used to compute the total size of a WEIGHTS component, as well as the number of pad bits that occur at the end of a WEIGHTS component (after the recorded weights) as a function N, the number of bits used to encode a single weight, and M, the number of weights recorded in the component. Note that the value of the pad bits must be zero, and that the number of pad bits must be recorded in the "padding" attribute of the MIME attachment containing this binary component.

  Total size = $\lceil MN/8 \rceil$

  Number of pad bits = (8 - MN mod 8) mod
- This component type is provided as an enhancement of the ACTUAL_DURATIONS component. It would typically be used whenever data weights are provided for data values at the lowest tree index levels (in the CROSS_DATA or AUTO_DATA components), while reducing the data volume that would have been required to store those weights in the ACTUAL_DURATIONS component.

> *EVLA WIDAR note*
>
> Initially, the number of bits allowed for the weights will be small multiples of 8; eventually, other values will be accommodated.
>
> *ALMA note*
>
> ALMA does not use weights.

6.4.1 Data stream dependencies

The following table lists the mandatory and optional binary components as a function of data stream type (specifically, the value of a *CorrelationMode* parameter only).

Binary component type	CorrelationMode		
	CROSS_ONLY	AUTO_ONLY	CROSS_AND_AUTO
FLAGS	O	O	O
ACTUAL_TIMES	O	O	O
ACTUAL_DURATIONS	O	O	O
ZERO_LAGS	O	O	O
AUTO_DATA	X	M	M
CROSS_DATA	M	X	M
WEIGHTS	O	O	O

Table 3: Mandatory and optional binary components. Symbols: O - optional, X - excluded, M - mandatory

7 Header elements

The parts of a MIME message in the binary data format that contain header information are represented as XML elements. The following sections provide an overview of these elements, their content, and their attributes. Further detailed, technical information about these elements, such as the formats and/or types of attribute values, can be found in the XML schemata associated with this document (see section 8.1).

7.1 Main data header

The main data header appears once in every binary data format BLOB (that is, once per sub-scan). The header itself has the name `sdmDataHeader`, and it has the following attributes:

- `xmlns`, a string which defines the SDM binary schema namespace with the fixed value: “http://TBD/XSDM/sdmBin”. Note that 'TBD' ~~must be defined for the appropriate archive repository~~has not yet been defined.
- `xmlns:xsi`, the namespace of the schema instance which has the fixed value of “`http://www.w3.org/2001/XMLSchema-instance`”.~~-~~
- `xmlns:xlink`, which has a fixed value of `"http://www.w3.org/1999/xlink"`.~~;~~
- `xmlns:xvers`, which has a fixed value of `"http://TBD/XVERSION"` Note that 'TBD' must be defined for the appropriate archive repository.
- `xsi:schemaLocation`, the location of the schema.
  - ~~ALMA note~~
- This value is `"http://TBD/XSDM/sdmbin 1~~0~~/sdmDataObject.xsd"`. Note that 'TBD' must be defined for the appropriate archive repository.
- `xvers:schemaVersion`, the version number of the XML schema to which the XML header elements in the BLOB conform and has the form of `"http://TBD/~~XVERSION~~1.0"`. ~~;~~Note that 'TBD' must be defined for the appropriate archive repository. There is a relationship between the version of this BDF document and the actual format described herein. This document has a version number *MM.mm*, where *MM* is the major version and *mm* is the minor version. The major version will contain substantive changes reflected in schema changes while the minor version will only address document-specific alterations. The *schemaVersion* will track the BDF major version number.
- `xvers:revision` the schema revision number.
  - ~~ALMA note~~
- This value is ~~related~~ equal to the CVS version number of the schema.
- `mainHeaderId`, which has a fixed value of “`sdmDataHeader`”.~~;~~
- `byteOrder`, the endianness of the binary data elements either “Little_Endian” or “Big_Endian”~~;~~

  ALMA note

  This value is “`Little~~IEEE_Low~~_Endian`”
- `projectPath`, a string that identifies the data in the BLOB by execution block number, scan number and sub-scan number (see section 3 for a description of these strings).

The content of the `sdmDataHeader` element is composed of a sequence of the following eight elements.

`startTime`

The time at which the acquisition of the data in the BLOB was started expressed as the (midpoint-interval)/2 of the first data block in an integration sent by the correlator, as an MJD value in nanoseconds.

`dataOID`

The URI of the BLOB; may be used as a reference to the BLOB from the SDM. The element is empty (in the XML sense), and has the following attributes:

- `xlink:type`, which has a fixed value; `"locator"`
- `xlink:href`, the URI of the BLOB (the same as the top-level MIME `X-uid` header value); and
- `xlink:title`, a user-defined, descriptive name for the BLOB.

> *EVLA WIDAR note*
>
> The `xlink:title` attribute has a fixed value of "EVLA WIDAR Visibility Data".

> *ALMA note*
>
> The `xlink:title` attribute has ~~a values~~values of "ALMA BL Correlator Spectral Data", "ALMA BL Correlator Channel Average Data", "ALMA ACA Correlator Spectral Data", ~~"~~and "ALMA ACA Correlator Channel Average Data"

`dimensionality` or `numTimes`

The choice between these two elements depends on the data structure. The `dimensionality` element must be used for data streams that produce a sequence of subsets in a time series of data ~~dump~~sets~~s~~. The use of the `dimensionality` element allows a BLOB to contain a structured hierarchy of data subsets. Alternatively, if the data acquired sequentially in time are blocked into chunks, with one chunk per subset, the `numTimes` element must be used. Note that the `numTimes` element cannot be used when there is more than one subset per BLOB.

When the data that are described by the main data header comprise a series of data subsets, the `dimensionality` element is ~~applicable~~used. In this case, the element specifies the axes along which the data subsets are ordered, with each axis corresponding to a level (or MIME multi-part section) in the high-level structure of the document. As an example, assume that there are two axes, "FOO" and "BAR", such that for every value of (FOO, BAR), one data subset occurs. If the data subsets in the BLOB appear in the order (FOO_0, BAR_0), (FOO_0, BAR_1), …, (FOO_0, BAR_{n0}), (FOO_1, BAR_0), (FOO_1, BAR_1), …, (FOO_1, BAR_{n1}), …, (FOO_m, BAR_0), (FOO_m, BAR_1), …, (FOO_m, BAR_{nm}), then the `dimensionality` element that describes the organization of the data subsets is `<dimensionality axes="FOO BAR">2</dimensionality>`. In this example, the structure of the MIME document would be as follows:

- top-level (content-type: multipart/mixed)
  - main data header
  - FOO_0 (content-type: multipart/mixed)
    - BAR_0 (content-type: multipart/related)
      - data subset header
      - binary component attachments
    - BAR_1 (content-type: multipart/related)
      - ...
    - …
    - BAR_{n1}
      - …
  - FOO_1 (content-type: multipart/mixed)
    - …
  - …
  - FOO_m
    - …
    -

When the data that are described by the main data header comprise a single data subset in which the data are ordered by time, the `numTimes` element is applicable. In this case, the element specifies the size of the time axis in the binary parts. The structure of a MIME document in this case is relatively flat:

- top-level (content-type: multipart/mixed)
  - main data header
  - data subset (content-type: multipart/related)
    - subset data header
    - binary component attachments (each attachment containing data values in a vector of length `numTimes`)

The `dimensionality` element has a single attribute:

- `axes`, the list of names of axes along which the data subsets are ordered (using *AxisName* enumeration values from Table 1).[3]

> *EVLA WIDAR note*
>
> For the EVLA WIDAR format (for sub-band cross- and auto-correlations), this element is fixed to be "`<dimensionality axes='TIM'>1</dimensionality>`".

> *ALMA note*
>
> For the ALMA-B correlator format (for sub-band cross- and auto-correlations), this element is fixed to be "`<dimensionality axes='TIM'>1</dimensionality>`".

[3] Note that, currently, only the TIM axis is supported by the known implementations of the format.

The `numTimes` element has no attributes, and its content is an integer specifying the size along the TIM axis of the data in the binary parts.

`execBlock`

Defines the execution block information for the data in the BLOB. It is the starting point of an outbound link to the execBlock SDM dataset. It exists to help ensure SDM data integrity, and may be used by BDF readers that wish to access the BDF directly without going through the SDM. This element has the following attributes:

- `xlink:href`, the execution block URI is of the form: `"uid://X1/2/3/4"`; and
- `xlink:type`, which has a fixed value. `"simple"`

`numAntenna`

The number of antennas used to obtain data in the BLOB. ~~Typically, this is the number of antennas in the (sub-)array.~~

`correlationMode`

One of the two elements that defines the data structure (or format variant) of the BLOB; it is a required element. Its content is a value of the *CorrelationMode* enumeration type.

> *EVLA WIDAR note*
>
> Content has fixed value of CROSS_AND_AUTO.
>
> *ALMA note*
>
> Content can be AUTO_ONLY or CROSS_AND_AUTO.

`spectralResolution`

One of the two elements that defines the data structure (or format variant) of the BLOB; it is an optional element because data processors are not required to support the concept of *SpectralResolutionType*. Its content is a value of the *SpectralResolutionType* enumeration type.

> *EVLA WIDAR note*
>
> Content has fixed value of FULL_RESOLUTION.
>
> *ALMA note*
>
> Content can be FULL_RESOLUTION or CHANNEL_AVERAGE

`dataStruct`

Provides the description of the data structure, defining the shapes of the binary components. This element must be typed at the instance level by selecting a type in a type hierarchy. The range of element types is constrained by the context set primarily by the `correlationMode` element but also, for refinements, by the `spectralResolution` element.

The contents and attributes of this element depend on the type of the element (defined by the value of the required `xsi:type` attribute). In the following descriptions of all possible child elements and attributes of the `dataStruct` element notations of the context in which each may or must appear are provided.

There are two possible attributes of the `dataStruct` element:

- `xsi:type`, the type declaration; and
- `apc`, the *AtmPhaseCorrection* value(s) of the data on the APC axis (the attribute is present if and only if cross-correlations are in the data stream).

The value of the `xsi:type` attribute must be consistent with the type of the data stream in which the `dataStruct` element appears, as shown in Table 4.

F C B U
C M M M M
A X X X X
B M M M M
apc

correlationMode	spectralResolution			
	FULL_RESOLUTION	CHANNEL_AVERAGE	BASEBAND_WIDE	*undefined*
CROSS_ONLY	CrossDataFull-Resolution	CrossDataChannel-Average	N/A	CrossData
AUTO_ONLY	AutoDataFull-Resolution	AutoDataChannel-Average	AutoDataBaseband-Wide	AutoData
CROSS_AND_AUTO	CrossAndAuto-DataFullResolution	CrossAndAutoData-ChannelAverage	N/A	CrossAndAutoData

Table 4: Values of `dataStruct.xsi:type` attribute as a function of data stream type.

> *EVLA WIDAR note*
>
> The EVLA WIDAR correlator data stream only uses the *CrossAndAutoDataFullResolution* type; the others may be ignored.
>
> *ALMA note*
>
> Data stream can be FULL_RESOLUTION: *AutoDataFullResolution* or *CrossAndAutoDataFullResolution* or CHANNEL_AVERAGE: *AutoDataChannelAverage* or *CrossAndAutoDataChannelAverage*. ~~The ACA correlator~~

The allowed child elements are `baseband`, `flags`, `actualTimes`, `actualDurations`, `crossData`, `autoData`, `zeroLags`, and `weights`. The occurrence of these child elements may depend on the declared type of the `dataStruct` element; see the following sections for details.

baseband

F C B U
C M M M M
A M M M M
B M M M M
numSpectralPoint

A named group of spectral windows. At least one of these elements appears in every `dataStruct` element, and may appear multiple times. Each `baseband` element has a single attribute:

- `name`, the baseband name, a value of type *BasebandName*.

F C B U
C M M M M
A M M M M
B M M M M
numBin

The content of a `baseband` element is a sequence of `spectralWindow` elements.

spectralWindow

F C B U
C M M M M
A X X X X
B M M M M
crossPolProducts

F C B U
C O O O O
A X X X X
B O O O O
scaleFactor

F C B U
C X X X X
A M M M M
B M M M M
sdPolProducts

Metadata describing the binary data structure for a spectral window. The element is empty, and the attributes that may appear depend on the type of the `dataStruct` element of which the `spectralWindow` is a descendant. The possible attributes, together with the context in which they may or must appear, are the following:

- `numSpectralPoint`, the number of spectral points (or channels) (*i.e.*, length of the SPP axis), required;
- `numBin`, the number of bins (*i.e.*, length of the BIN axis), required;
- `crossPolProducts`, the ordered list of polarization products produced for each cross-correlation (*i.e.*, the *StokesParameter* names of the points on the POL axis), the list must be one of “XX”, “YY”, “XX YY”, “XX XY YX YY”, “RR”, “LL”, “RR LL”, or “RR RL LR LL”;
- `scaleFactor`, the scaling factor for visibilities recorded as integers whose definition can be found in [Scott];
- `sdPolProducts`, the ordered list of polarization products produced for single dish data (*i.e.*, the *StokesParameter* names of the points on the POL axis), the list must be one of “XX”, “YY”, “XX YY”, “XX XY YY”, “RR”, “LL”, “RR LL”, “RR RL LL”;

F C B U
C O O O O
A O O O O
B O O O O
id, image

- `id`, a unique identifier for the spectral window across all basebands whose value is of the form `spw_N` where *N* is a~~n~~ positive integer and allow spectral windows can appear in any order;
- `image`, for double sideband receivers, this identifies the *image* sideband of this spectral window whose value is of the form `spw_N` where *N* is an integer; and

F C B U
C M M M M
A M M M M
B M M M M
sideband

- `sideband`, the sideband type whose value is of type *NetSideBand,* required .

EVLA WIDAR note

`id` and `image` are not used, and `sideband` always has the value NOSB.

ALMA note

ALMA has double sideband receivers which separate the upper and lower sidebands by a synchronous 90° phase switching scheme among the LO system and the correlator. These sidebands then appear as "pairs" of spectral windows with a "primary" and "image" spectral window. Thus the elements, *id*, *image*, and *sideband* provide the necessary information to uniquely identify which sideband corresponds to which spectral window.

`flags`, `actualTimes`, `actualDurations`, `crossData`, `autoData`, `zeroLags`, `weights`

Each of these elements provides metadata for the structure of binary components of a particular type in the BLOB[4]. The associations of these elements with the types of binary components is given in section 6.4, and the table in the following section shows which of these elements may appear in a BLOB of a given of data stream type. There is no specific ordering relationship required between these elements and the corresponding binary attachments. Each of these elements has the following attributes:

- `size`, the maximum number of primitive data type values in the binary component within all data subsets (note that this number is not necessarily the number of elements in the component because two primitive data type values are used to represent a complex number); and
- `axes`, the list of axes for the binary component (as described in Section 6.3).

[4] Whereas these elements with a main data header provide metadata for the binary components, the similarly named elements that exist within the subset data headers provide references to particular instances of the binary components for the given data subset.

F C B U
C M M X M
A M M O M
B M M X M
flags

flags

Associated with FLAGS binary component. Attributes as shown above.

F C B U
C M M X M
A M M O M
B M M X M
actualTimes

actualTimes

Associated with ACTUAL_TIMES binary component. Attributes as shown above.

F C B U
C M M X M
A M M O M
B M M X M
actualDurations

actualDurations

Associated with ACTUAL_DURATIONS binary component. Attributes as shown above.

F C B U
C M M X M
A M M O M
B M M X M
crossData

crossData

Associated with CROSS_DATA binary component. Attributes as shown above.

F C B U
C M M X M
A X X X X
B M M X M
autoData

autoData

Associated with AUTO_DATA binary component. Attributes as shown above.

F C B U
C X X X X
A M M M M
B M M X M
zeroLags

zeroLags

Associated with ZERO_LAGS binary component. Attributes as shown above.

F C B U
C M O X O
A M O X O
B M O X O
weights

weights

Associated with WEIGHTS binary component. Attributes as shown above. This element contains the lookup table for the data elements in the binary component. The size of the lookup table must be an integral power of two. The content of this element is the ordered list of lookup table values as single precision floating point numbers.

F C B U
C O O X O
A X X X X

6.1.1 Data stream dependencies

7.1.1

correlationMode	spectralResolution		
	FULL_RESOLUTION	CHANNEL_AVERAGE	BASEBAND_WIDE
CROSS_ONLY	crossData flags actualTimes actualDurations *weights* zeroLags	crossData flags actualTimes actualDurations *weights* zeroLags	
AUTO_ONLY	autoData flags actualTimes actualDurations zeroLags	autoData flags actualTimes actualDurations zeroLags	autoData *flags* *actualTimes* *actualDurations*
CROSS_AND_AUTO	crossData autoData flags actualTimes actualDurations *weights* zeroLags	crossData autoData flags actualTimes actualDurations *weights* zeroLags	

Table 5: Mandatory and optional header elements by data stream. Optional elements are italicized.

7.2 Data subset header

The data subset header appears once for every (sub-)integration in a binary format BLOB.. The XML header element, `sdmDataSubsetHeader`, is also typed at the instance level, with the range of element types being determined by the data stream type.

The contents of this element depend on the type of the element, defined by the value of the required `xsi:type` attribute. The element has the following two attributes:

- `xsi:type`, the type declaration (required); and
- `projectPath`, a string that identifies the data in the (sub-)integration by execution block number, scan number, sub-scan number, integration number, and, if applicable, sub-integration number (see section 3) (required).

The value of the `xsi:type` attribute must be consistent with the type of the data stream in which the `dataRef` element appears, as shown below in Table 6.

correlationMode	spectralResolution			
	FULL_RESOLUTION	CHANNEL_AVERAGE	BASEBAND_WIDE	*undefined*
CROSS_ONLY	BinaryCrossDataFull-Resolution	BinaryCrossData-ChannelAverage	N/A	BinaryCrossData
AUTO_ONLY	BinaryAutoDataFull-Resolution	BinaryAutoData-ChannelAverage	BinaryAutoData-BasebandWide	BinaryAutoData
CROSS_AND_-AUTO	BinaryCrossAnd-AutoData-FullResolution	BinaryCrossAndAuto-DataChannelAverage	N/A	BinaryCrossAnd-AutoData

Table 6: Values of `sdmDataSubsetHeader.xsi:type` attribute as a function of data stream type.

EVLA WIDAR note

The only type used by the EVLA WIDAR correlator is *BinaryCrossAndAutoDataFullResolution*; the others may be ignored.

ALMA note

ALMA correlators support *BinaryCrossAndAutoDataFullResolution*, *BinaryAutoDataFullResolution*, *BinaryCrossAndAutoDataChannelAverage*, *BinaryAutoDataChannelAverage*.

The content of the `sdmDataSubsetHeader` element is composed of a sequence of the following elements. The first two elements described below are required in all cases, whereas the remaining elements are mandatory, optional or excluded depending on the type of the data stream in which the `sdmDataSubsetHeader` element appears.

`schedulePeriodTime`

The scheduled time and duration of the time interval associated with the data in the current data subset, to be used as fallback metadata when the ACTUAL_TIMES and/or ACTUAL_DURATIONS binary components are absent from the data subset. These values are given by the content of the following two child elements of the `schedulePeriodTime` element:

- **`time`**

  the time at the midpoint of the interval, as an MJD value in nanoseconds; and

- **`interval`**

  the duration of the interval, in nanoseconds.

dataStruct

A reference to the main data header that describes the structure of the binary components in the current subset. The element itself is empty, but it has the following attribute:

- `ref`, a reference to the main data header (*i.e.*, the `mainHeaderId` attribute value of the referenced element).
-

F C B U
C O O O O
A O O O O
B O O O O
abortObservation

abortObservation

If a subscan is prematurely ended, this element appears together with the `schedulePeriodTime` and `dataStruct` elements as the only children of the `sdmDataSubsetHeader` element. No `flags`, `actualTimes`, `actualDurations`, `crossData`, `autoData`, `zeroLags`, or `weights` elements should exist in the same sdmDataSubsetHeader element with an `abortObservations` element. An `abortObservation` element is written anytime a subscan is aborted, *i.e.*, by user intervention or by a detected software error. This element contains two required child elements:

- `stopTime`, the time at which the subscan is stopped. This time is as an MJD value in nanoseconds; and
- `reason`, a string providing a description of the reason for the aborted subscan.

F C B U
C O O X O
A O O O O
B O O X O
flags

The remaining seven child elements of the `sdmDataSubsetHeader` element provide the relative URIs of the binary components in the data subset. In all cases, the elements have the attribute `xlink:href`, which specifies the URI of the associated binary component (*i.e.*, the `Content-location` header value of the MIME part containing the binary component); any additional attributes are described below for each element individually.

F C B U
C M M X M
A M M M M
B M M X M
actualTimes

flags

Provides the URI of the FLAGS binary component.

F C B U
C M M X M
A M M M M
B M M X M
actualDurations

actualTimes

Provides the URI of the ACTUAL_TIMES binary component.

actualDurations

Provides the URI of the ACTUAL_DURATIONS binary component

F C B U
C M M X M
A X X X X
B M M X M
crossData

crossData

Provides the URI of the CROSS_DATA binary component. It has one additional required attribute:

- **type**, the primitive data type used to represent the cross-correlation data in the referenced binary component. Its value is one of the following *PrimitiveDataType* enumeration values: SHORT_TYPE, LONG_TYPE, INT16_TYPE, INT32_TYPE, FLOAT32_TYPE.

F C B U
C M O X O
A M O X O
B M O X O
zeroLags

> *ALMA note*
>
> Currently, ALMA only supports SHORT_TYPE and LONG_TYPE.

> *EVLA WIDAR note*
>
> The WIDAR correlator initially will use only FLOAT32_TYPE.

F C B U
C O O X O
A X X X X
B O O X O
weights

autoData

Provides the URI of the AUTO_DATA binary component.

zeroLags

Provides the URI of the ZERO_LAGS binary component.

weights

Provides the URI of the WEIGHTS binary component.

8 Appendices

8.1 XML schemata

XML schemata for the binary data format may be found in the same repository and project under which the master copy of the present document exists.

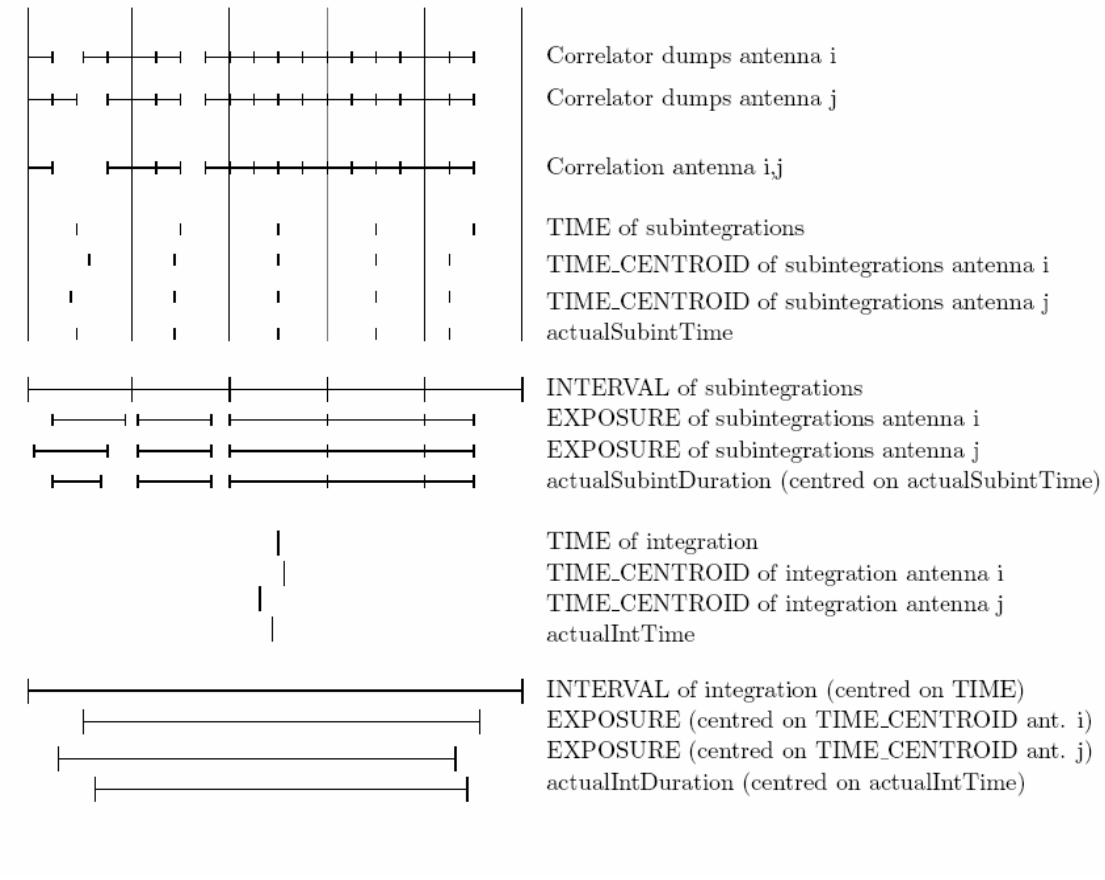
REPOSITORY ACCESS INSTRUCTIONS GO HERE

8.2 MIME format example

```
MIME-Version:1.0
Content-Type: multipart/mixed; boundary=”ABCDE01234”; type="text/xml"
Content-Description: EVLA/CORRELATOR/WIDAR/FULL_RESOLUTION
Content-Location: uid://X1/2/3/4

--ABCDE01234
Content-Type: text/xml; charset=”UTF-8”
Content-Transfer-Encoding: 8bit
Content-Location: sdmDataHeader.xml

[MAIN DATA HEADER]

--ABCDE01234
Content-type: multipart/related; boundary=”abcd”;type="text/xml";
Content-description: data and metadata subset

--abcd
Content-Type: text/xml; charset=”UTF-8”
Content-Location: 1/10/3/1/desc.xml

[DATA SUBSET HEADER]
--abcd
Content-Type: application/octet-stream
Content-Location: 1/10/3/1/actualTimes.bin

[ACTUAL_TIMES binary data]
--abcd
Content-Type: application/octet-stream
Content-Location: 1/10/3/1/actualDurations.bin

[ACTUAL_DURATIONS binary data ]
--abcd
...
--abcd--

--ABCDE01234
Content-type: multipart/relate; boundary=”efgh”
Content-description: data subset

--efgh
Content-Type: text/xml; charset="UTF-8"
Content-Location: 1/10/3/2/desc.xml

[DATA SUBSET HEADER]
--efgh
Content-Type: application/octet-stream
Content-Location: 1/10/3/2/actualTimes.bin

[ACTUAL_TIMES binary data]
--efgh
...
--efgh--
--ABCDE01234--
```

8.3 Schematic representation of times

Data blanking implies that part or all of the data in an integration is ignored. This affects the actual times and durations of the integration. The following diagram schematically presents how actual times and durations for a baseline are determined.



8.4 ALMA implementation details

8.4.1 Spectral data stream

8.4.1.1 Binary component sizes

Here we discuss the sizes of the binary attachments in bytes. The number of basebands, N_{bb}, corresponds to the number of baseband nodes in the sequences; the number of spectral windows per baseband, $N_{sw}(bb)$, corresponds to the number of spectral window nodes, children of a baseband node. These are defined as sequences in the schema with the XML header setting the actual order of the data given common to all baselines.

In the formulas used to determine the size of the arrays, one must take into account the following possible cases as indicated by the value of the SDM item, *correlationMode*:

- $N_{ant} = 0$ and $N_{bl} > 0$ if *correlationMode* = CROSS_ONLY
- $N_{ant} > 0$ and $N_{bl} = 0$ if *correlationMode* = AUTO_ONLY
- $N_{ant} > 0$ and $N_{bl} > 0$ if *correlationMode* = CROSS_AND_ AUTO

When the *correlationMode* = CROSS_AND_ AUTO, the number of polarization products, N_{pp}, may not be the same for the auto-correlation (single-dish) and cross-correlation (interferometric) data. For this reason, a superscript is added. N_{pp}^{o} is used for the auto-correlation data and N_{pp}^{oo} for the cross-correlation data. The following constraints apply:

- if *correlationMode* = CROSS_ONLY: N_{pp}^{oo} = 1, 2, or 4 and N_{pp}^{o} is undefined
- if *correlationMode* = AUTO_ONLY: N_{pp}^{oo} is undefined and N_{pp}^{o} is 1, 2, or 3
- if *correlationMode* = CROSS_AND_ AUTO: N_{pp}^{oo} = 1, 2, or 4 and

$$N_{pp}^{o} = \begin{cases} N_{pp}^{o} \text{ ,if } N_{pp}^{oo} \le 2 \\ 3, \quad \text{if } N_{pp}^{oo} = 4 \text{ (standard mode)} \\ 2, \quad \text{if } N_{pp}^{oo} = 4 \text{ (non-standard mode)} \end{cases}$$

Note that when N_{pp}^{o} = 3, the data are in a set of two real values plus a complex value for the polarization cross products. Else, when $N_{pp}^{o} \le 2$, the data consist of N_{pp}^{o} real values.

The term N_{apc} derives from the sequence of SDM enumerations: AP_UNCORRECTED, AP_CORRECTED, or AP_MIXED. N_{apc} is simply the number of items in this sequence.

Error! Bookmark not defined.FLAGS

These data are long unsigned 32-bit integers and are antenna based. The number of elements is computed as such:

$$N_{ant} \sum_{i=1}^{N_{bb}} N_{bin}(i) N_{pp}^{o}(i)$$

ACTUAL_TIMES

The size of the actual integration time stamp (centroid) for each baseline, bin, polarization product, and baseband. This is the intersection of the 2 antennas (self or cross) of a baseline taking into account any gaps due to blanking in either antenna. The number of elements is computed as such:

$$N_{bl} \sum_{i=1}^{N_{bb}} N_{bin}(i) N_{pp}^{oo}(i) + N_{ant} \sum_{i=1}^{N_{bb}} N_{bin}(i) N_{pp}^{o}(i)$$

ACTUAL_DURATIONS

The actual integration duration (exposure) for each baseline, bin, polarization product, and baseband. This is the intersection of the 2 antennas of a baseline taking into account any gaps due to blanking in either antenna. The number of elements is computed as such:

$$N_{bl} \sum_{i=1}^{N_{bb}} N_{bin}(i) N_{pp}^{oo}(i) + N_{ant} \sum_{i=1}^{N_{bb}} N_{bin}(i) N_{pp}^{o}(i)$$

ZERO_LAGS

The size of the zero lag values for each antenna, spectral window, polarization product and baseband. The number of elements is computed as such:

$$N_{ant}\sum_{i=1}^{N_{bb}} N_{bin}(i)N_{sw}(i)\,f\,(N_{pp})(i)$$

with

$$f\,(N_{pp})=\begin{cases}1\ if\ N_{pp}^{o}(i)=1\ or\ N_{pp}^{oo}(i)=1\\ 2\ if\ N_{pp}^{o}(i)>1\ or\ N_{pp}^{oo}(i)>1\end{cases}$$

CROSS_DATA

The size of the cross correlation spectral data for each cross baseline, spectral window, polarization product, bin, and baseband. These data are complex values. The number of elements is computed as such:

$$2\times N_{apc}N_{bl}\sum_{i=1}^{N_{bb}}\left(N_{bin}(i)N_{pp}^{00}(i)\sum_{j=1}^{N_{sw}(i)}N_{sp}(j)\right)$$

where a factor of 2 is due to complex visibilities.

AUTO_DATA

The size of the auto correlation spectral data. Note that the data can be a mixture of real and complex values due to the cross polarization products when using full polarization. The number of elements is computed as such:

$$N_{ant}\sum_{i=1}^{N_{bb}}\left(N_{bin}(i)f\left(N_{pp}(i)\right)\sum_{j=1}^{N_{sw}(i)}N_{sp}(j)\right)$$

where for pure auto-correlation data, i.e., *correlationMode* = AUTO_ONLY

$$f(N_{pp}(i))=\begin{cases}N_{pp}^{oo}(i),\ \text{if}\ N_{pp}^{o}(i)\le 2\\ 4,\qquad \text{if}\ N_{pp}^{o}(i)=3\end{cases}$$

and for *correlationMode* = CROSS_AND_AUTO

$$f(N_{pp}(i))=\begin{cases}N_{pp}^{oo}(i),\ \text{if}\ N_{pp}^{oo}(i)\le 2\ (i.e.\ 1\ re\ \text{if}\ N_{pp}^{oo}(i)=1,\ 2\ re\ \text{if}\ N_{pp}^{oo}(i)=2)\\ N_{pp}^{oo}(i),\ \text{if}\ N_{pp}^{oo}(i)=4\ (i.e.\ 3\ re+1\ im,\ \text{standard mode})\\ 2,\qquad \text{if}\ N_{pp}^{o}(i)=2\ (i.e.\ 2\ re,\ \text{non-standard mode})\end{cases}$$

Note that in order to support the CROSS_ONLY ~~non-standard~~ mode case, one needs to add a boolean item to the SDM and to the XML header – This is currently a TBD item.

Binary component	Data type	Data Size	Min. Quantity	Max. Quantity (size in bytes)
FLAGS	unsigned long integer	4	1 (4)	133,120 (532,480)
ACTUAL_TIMES	SDM::Time	8	1 (8)	133,120 (1,064,960)
ACTUAL_DURATIONS	SDM::Time	8	1 (8)	133,120 (1,064,960)

Binary component	Data type	Data Size	Min. Quantity	Max. Quantity (size in bytes)
ZERO_LAGS	float	4	1 (4)	131,072 (524,288)
CROSS_DATA	scaled short/long integer	2/4	0 (0)	1,056,964,608 (4,227,858,432)
AUTO_DATA	float	4	256 (1024)	8,388,608 (33,554,432)
XML Headers	Char	1	3433	~6MB
Total size in bytes			**~~1,048~~4,481**	**~4 GB (4,2~~70~~64,599,552)**

Table 7: Full Resolution b~~B~~inary attachment sizes

8.4.1.2 Example of Full Resolution Data

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="MIME_boundary-1"; type="text/xml";
Content-Description: ALMA/CORRELATOR/ALMA_BASELINE/FULL_RESOLUTION
Content-Location: uid://X1/1/0/0

--MIME_boundary-1
Content-Type: text/xml; charset="UTF-8"
Content-Transfer-Encoding: 8bit
Content-Location: sdmDataHeader.xml

<?xml version="1.0" encoding="UTF-8"?>
<sdmDataHeader
  xmlns="http://TBD/XSDM/sdmbin"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xvers="http://TBD/XVERSION"
  xsi:schemaLocation="http://TBD/XSDM/sdmbin 10/sdmDataObject.xsd"
  xvers:schemaVersion="0"
  xvers:revision="0.0.96"
  mainHeaderId="sdmDataHeader"
  byteOrder="LittleIEEE_Low_Endian"
  projectPath="3/1/2/">
  <startTime>4647257068000000000</startTime>
  <dataOID xlink:type="locator"
    xlink:href="uid://X1/1/0/0"
    xlink:title="ALMA BL Correlator Spectral Data"/>
  <dimensionality axes="TIM">1</dimensionality>
  <execBlock xlink:href="uid://X1/1/0/1" xlink:type="simple"/>
  <numAntenna>2</numAntenna>
  <correlationMode>CROSS_AND_AUTO</correlationMode>
  <spectralResolution>FULL_RESOLUTION</spectralResolution>
  <dataStruct xsi:type="CrossAndAutoDataFullResolution" apc="AP_UNCORRECTED">
    <baseband name="BB_1">
      <spectralWindow id="spw_2" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7680~~1~~1920" numBin="1" sideband="N~~O~~LSB"/>
      <spectralWindow id="spw_1" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7680~~1~~1920" numBin="1" sideband="LN~~O~~SB"/>
      <spectralWindow id="spw_3" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7680~~1~~1920" numBin="1" sideband="N~~O~~LSB"/>
      <spectralWindow id="spw_4" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7680~~1~~1920" numBin="1" sideband="LN~~O~~SB"/>
    </baseband>
    <baseband name="BB_3">
      <spectralWindow id="spw_5" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7680~~1~~1920" numBin="1" sideband="LN~~O~~SB"/>
      <spectralWindow id="spw_6" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7680~~1~~1920" numBin="1" sideband="LN~~O~~SB"/>
      <spectralWindow id="spw_7" sdPolProducts="XX" crossPolProducts="XX"
```

```
scaleFactor="3225.523213" numSpectralPoint="76801920" numBin="1" sideband="NOLSB"/>
      <spectralWindow id="spw_8" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="76801920" numBin="1" sideband="LNOSB"/>
    </baseband>
    <flags size="6" axes="BAL ANT BAB"/>
    <actualTimes size="6" axes="BAL ANT BAB"/>
    <actualDurations size="6" axes="BAL ANT BAB"/>
    <crossData size="122880" axes="BAL BAB SPW SPP"/>
    <autoData size="122880" axes="ANT BAB SPW SPP"/>
    <zeroLags size="16" axes="BAL BAB SPW"/>
  </dataStruct>
</sdmDataHeader>
--MIME_boundary-1
Content-Type: multipart/related; boundary="MIME_boundary-2";type="text/xml";
Content-Description: data and metadata subset
--MIME_boundary-2
Content-Type: text/xml; charset="UTF-8"
Content-Location: 3/1/2/1/desc.xml

<sdmDataSubsetHeader
  xsi:type="BinaryCrossAndAutoDataFullResolution"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  projectPath="3/1/2/1/">
  <schedulePeriodTime>
    <time>4647257073120000000</time>
    <interval>10240000000</interval>
  </schedulePeriodTime>
  <dataStruct ref="sdmDataHeader"/>
  <flags xlink:href="3/1/2/1/flags.bin"/>
  <actualTimes xlink:href="3/1/2/1/actualTimes.bin"/>
  <actualDurations xlink:href="3/1/2/1/actualDurations.bin"/>
  <crossData xlink:href="3/1/2/1/crossData.bin" type="SHORT_TYPE"/>
  <autoData xlink:href="3/1/2/1/autoData.bin"/>
  <zeroLags xlink:href="3/1/2/1/zeroLags.bin"/>
</sdmDataSubsetHeader>
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/flags.bin

[BINARY DATA HERE]
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/actualTimes.bin

[BINARY DATA HERE]
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/actualDurations.bin

[BINARY DATA HERE]
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/crossData.bin

[BINARY DATA HERE]
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/autoData.bin

[BINARY DATA HERE]
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/zeroLags.bin

[BINARY DATA HERE]
--MIME_boundary-2--
--MIME_boundary-1--
```

8.4.1.3 Data rate

The quantity of correlator spectral data can vary greatly. A minimal header and data for a single antenna is about 2 KB. For all 64 antennas the data can extend to over 2 GB for all four basebands. The maximum output rate of the ALMA-B correlator to the CDP computers is 1 GB/second across all basebands. Other meta data add about 1% to this. ~~With an integration period of 1 second this exceeds the SSR-specified maximum peak data rates, consequently the SSR values of 6 MB/sec average and 60 MB/sec peak are used.~~

8.4.2 Channel average data stream

8.4.2.1 Binary component sizes

FLAGS

These data are long unsigned integers. The number of elements is computed as such:

$$N_{bl}\sum_{i=1}^{N_{bb}}N_{bin}(i)N_{pp}^{00}(i)+N_{ant}\sum_{i=1}^{N_{bb}}N_{bin}(i)N_{pp}^{0}(i)$$

ACTUAL_TIMES

The size of the actual sub-integration time stamp (centroid) for each baseline, bin, polarization product, and baseband taking into account any blanking, see appendix **Error! Reference source not found.** for details. The number of elements is computed as such:

$$N_{bl}\sum_{i=1}^{N_{bb}}N_{bin}(i)N_{pp}^{00}(i)+N_{ant}\sum_{i=1}^{N_{bb}}N_{bin}(i)N_{pp}^{0}(i)$$

ACTUAL_DURATIONS

The actual sub-integration duration (exposure) for each baseline, bin, polarization product, and baseband. This is the intersection of the 2 antennas of a baseline taking into account any gaps due to blanking in either antenna. **Error! Reference source not found.**The number of elements is computed as such:

$$N_{bl}\sum_{i=1}^{N_{bb}}N_{bin}(i)N_{pp}^{00}(i)+N_{ant}\sum_{i=1}^{N_{bb}}N_{bin}(i)N_{pp}^{0}(i)$$

ZERO_LAGS

The size of the zero lag values for each antenna, spectral window, polarization product and baseband. The number of elements is computed as such:

$$N_{ant}\sum_{i=1}^{N_{bb}} N_{bin}(i)N_{sw}(i)\,f\,(N_{pp})(i)$$

with

$$f(N_{pp})=\begin{Vmatrix} 1\ if\ N^{o}_{pp}(i)=1\ or\ N^{oo}_{pp}(i)=1 \\ 2\ if\ N^{o}_{pp}(i)>1\ or\ N^{oo}_{pp}(i)>1 \end{Vmatrix}$$

CROSS_DATA

The size of the cross correlation channel averages. These data are complex values. The number of elements is computed as such:

$$2\times N_{bl}\sum_{i=1}^{N_{bb}}\left(N_{bin}N^{0}_{pp}(i)\sum_{j=1}^{N_{sw}(i)}N_{sp}(j)\right)$$

where a factor of 2 is due to complex visibilities.

AUTO_DATA

The size of the auto correlation channel averages. Note that the data can be a mixture of real and complex values due to the cross polarization products when using full polarization. The number of elements is computed by the following formula. For full polarization:

$$N_{ant}\sum_{i=1}^{N_{bb}}\left(N_{bin}(i)f\left(N^{0}_{pp}(i)\right)\sum_{j=1}^{N_{sw}(i)}N_{sp}(j)\right)$$

where for pure auto-correlation data, i.e., *correlationMode* = AUTO_ONLY

$$f\left(N_{pp}(i)\right)=\begin{cases} N^{o}_{pp}(i)\ if\ N^{o}_{pp}(i)\le 2 \\ 4 \qquad if\ N^{o}_{pp}(i)=3 \end{cases}$$

and for *correlationMode* = AUTO_AND_CROSS

$$f(N_{pp}(i))=\begin{cases} N^{oo}_{pp}(i)\ \text{if}\ N^{oo}_{pp}(i)\le 2\ (i.e.\ 1\ re\ \text{if}\ N^{oo}_{pp}(i)=1,\ 2\ re\ \text{if}\ N^{oo}_{pp}(i)=2) \\ N^{oo}_{pp}(i)\ \text{if}\ N^{oo}_{pp}(i)=4\ (i.e.\ 3\ re+1\ im,\text{standard mode}) \\ 2 \qquad \text{if}\ N^{o}_{pp}(i)=2\ (i.e.\ 2\ re,\text{non-standard mode}) \end{cases}$$

Using these mapping rules and the header information, the sizes of data are shown in the following table with the number of bytes is shown in parentheses.

Binary component	Data type	Data Size	Min. Quantity	Max. Quantity (size in bytes)
FLAGS	Unsigned long integer	4	1 (4)	133,120 (532,480)
ACTUAL_TIMES	SDM::Time	8	1 (8)	133,120 (1,064,960)
ACTUAL_DURATIONS	SDM::Time	8	1 (8)	133,120 (1,064,960)
ZERO_LAGS	float	4	1 (4)	131,072 (524,288)
CROSS_DATA	scaled short / integer	2/4	0	20,643,840 (82,575,360)
AUTO_DATA	float	4	1 (4)	327,680 (1,310,720)
XML Headers	Char	1	3433	~6MB
Total sizes			**34374 (345724)**	**(~9386 MB) 93,072,768(86,548,480)**

Table 8: Channel average binary data sizes

8.4.2.2 Example of Channel Average Data

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="MIME_boundary-1"; type="text/xml";
Content-Description: ALMA/CORRELATOR/ALMA_BASELINE/CHANNEL_AVERAGE
Content-Location: uid://X1/1/0/0

--MIME_boundary-1
Content-Type: text/xml; charset="UTF-8"
Content-Transfer-Encoding: 8bit
Content-Location: sdmDataHeader.xml

<?xml version="1.0" encoding="UTF-8"?>
<sdmDataHeader xmlns="http://TBD/XSDM/sdmbin"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xvers="http://TBD/XVERSION"
  xsi:schemaLocation="http://TBD/XSDM/sdmbin 10/sdmDataObject.xsd"
  xvers:schemaVersion="0"
  xvers:revision="0.0.96"
  mainHeaderId="sdmDataHeader"
  byteOrder="LittleIEEE_Low_Endian"
  projectPath="3/1/2/">
  <startTime>4647257068000000000</startTime>
  <dataOID xlink:type="locator" xlink:href="uid://X1/1/0/0" xlink:title="ALMA BL
Correlator Channel Average Data"/>
  <dimensionality axes="TIM">1</dimensionality>
  <execBlock xlink:href="uid://X1/1/0/1" xlink:type="simple"/>
  <numAntenna>2</numAntenna>
  <correlationMode>CROSS_AND_AUTO</correlationMode>
  <spectralResolution>CHANNEL_AVERAGE</spectralResolution>
  <dataStruct xsi:type="CrossAndAutoDataChannelAverage" apc="AP_UNCORRECTED">
  <baseband name="BB_1">
    <spectralWindow id="spw_1" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7" numBin="1"/>
    <spectralWindow id="spw_2" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7" numBin="1"/>
    <spectralWindow id="spw_3" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7" numBin="1"/>
    <spectralWindow id="spw_4" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="7" numBin="1"/>
  </baseband>
  <baseband name="BB_3">
    <spectralWindow id="spw_5" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="8" numBin="1"/>
    <spectralWindow id="spw_6" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="8" numBin="1"/>
    <spectralWindow id="spw_7" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="8" numBin="1"/>
    <spectralWindow id="spw_8" sdPolProducts="XX" crossPolProducts="XX"
scaleFactor="3225.523213" numSpectralPoint="8" numBin="1"/>
  </baseband>
  <flags size="6" axes="BAL ANT BAB"/>
```

```
  <actualTimes size="6" axes="BAL ANT BAB"/>
  <actualDurations size="6" axes="BAL ANT BAB"/>
  <crossData size="112" axes="BAL BAB SPW SPP"/>
  <autoData size="112" axes="ANT BAB SPW SPP"/>
  <zeroLags size="16" axes="BAL BAB SPW"/>
  </dataStruct>
</sdmDataHeader>
--MIME_boundary-1
Content-Type: multipart/related; boundary="MIME_boundary-2";type="text/xml";
Content-Description: data and metadata subset
--MIME_boundary-2
Content-Type: text/xml; charset="UTF-8"
Content-Location: 3/1/2/1/1/desc.xml

<sdmDataSubsetHeader xsi:type="BinaryCrossAndAutoDataChannelAverage"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" projectPath="3/1/2/1/1/">
  <schedulePeriodTime>
    <time>4647257068256000000</time>
    <interval>512000000</interval>
  </schedulePeriodTime>
  <dataStruct ref="sdmDataHeader"/>
  <flags xlink:href="3/1/2/1/1/flags.bin"/>
  <actualTimes xlink:href="3/1/2/1/1/actualTimes.bin"/>
  <actualDurations xlink:href="3/1/2/1/1/actualDurations.bin"/>
  <crossData xlink:href="3/1/2/1/1/crossData.bin" type="SHORT_TYPE"/>
  <autoData xlink:href="3/1/2/1/1/autoData.bin"/>
  <zeroLags xlink:href="3/1/2/1/1/zeroLags.bin"/>
</sdmDataSubsetHeader>
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/1/flags.bin

[BINARY DATA HERE]
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/1/actualTimes.bin

[BINARY DATA HERE]
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/1/actualDurations.bin

[BINARY DATA HERE]
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/1/crossData.bin

[BINARY DATA HERE]
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/1/autoData.bin

[BINARY DATA HERE]
--MIME_boundary-2
Content-Type: application/octet-stream
Content-Location: 3/1/2/1/1/zeroLags.bin

[BINARY DATA HERE]
--MIME_boundary-2--
--MIME_boundary-1--
```

8.4.2.3 Data rate

Assume that the channel average integration header is ~2KB bytes. Note that the size of the channel data in the maximum case can be huge. This, of course, violates the maximum data rate specified by the SSR. A typical maximum rate would be ~500KB per sub-integration interval.

Data Description	Data Quantity (bytes)	Instances per transmission	Transmission Interval (seconds)	Data Rate (average/peak)
Channel Aver. Bulk Data (1 antenna)	2KB	1	0.5	2/4 KB/sec
Channel Aver. Bulk Data (64 antennas)	11 MB	1	0.5	11/22 MB/sec

Table 9 –: Channel average data flow

8.4.3 Data capture information

After each subscan, a CORBA binary structure is sent to the Data Capture component for a given subarray. This structure contains identifiers from Control and run-time information from Control and Correlator.

The function is defined in `ICD/OFFLINE/idl/DataCapture.idl` as:

```
void sendSubScanCorrelatorData( in string arrayId,
                                in Correlator::subScanCorrelatorData subScanData )
raises( DataCaptureExceptions::DataErrorEx, DataCaptureExceptions::TableUpdateErrorEx);
```

Where `Correlator::subScanCorrelatorData` is defined in `ICD/CORR/ws/idl/CorrDataCapture.idl` as

```
struct SubScanCorrelatorData
{
   long totalIntegrations;
   long numberSubIntegrationsPerIntegration;
   boolean flagRowIntegrations;
   boolean flagRowSubIntegrations;
};
```

The SDM has one row in the table for all integrations and one row for all sub-integrations. Thus we need to send the total number of integrations in the subscan and the total number of sub-integrations in each integration. Finally a row flag for the integrations and sub-integrations. Note that the data OIDs are delivered to DataCapture via the Control subsystem at the beginning of a subscan.

The data items produced by the correlator are:

- **totalIntegrations**, the total number of integrations in the subscan
- **numberSubIntegrationsPerIntegration**, the total number of sub-integrations in each integration
- **flagRowIntegrations**, general flag set by correlator to flag entire row of integrations with TRUE = 'bad', FALSE = 'good'.
- **flagRowSubIntegrations**, general flag set by correlator to flag entire row of sub-integrations with. TRUE = 'bad', FALSE = 'good'.

Data flow rates are minimal as at most four sets of SubScanCorrelatorData would be sent simultaneously at the end of each subscan resulting in tens of bytes per second.

8.4.4 Data transmission

This section describes how data will be transferred from the CDP master computer which organizes the binary data to the Archive and other data receivers.

8.4.4.1 Overview

The general approach is that for each subscan, the CDP master sends *startSend*() and *send*() commands to the *BulkData Distributor* which contains header information regarding all of the data to be sent in the subscan. At each integration, the CDP master sends the SDM subset headers and binary data via the *send*() command. At the end of the subscan, the CDP master sends a *stopSend*() command signaling the receiver that no more data is coming for this subscan. The current design has a separate pair of flows per subarray, one for the integrations and one for the sub-integrations. Throughout this discussion, the term “sub-integration” can replace “integration”.

8.4.4.2 Subscan start

When a *startSubScan*() command is received by the CDP master, it constructs the SDM data header and calculates the maximum expected data size of text and binary data. This header information is sent via the *startSend*() and *send*(). Note that the actual data may be less than the maximum due to the baselineFlags , actualTimes, and actualDuration attachments not being sent.

8.4.4.3 Sending data

For each integration, the CDP master sends integration subset data as currently defined.

8.4.4.4 Subscan end

At the end of a subscan, the CDP master invokes the *stopSend()* function which signals the receivers that no more binary data for this subscan will be sent. Note that the *stopSend()* function is sent even if the subscan is ended prematurely, i.e., before the expected number of bytes defined in the *startSend()* function are sent. The CDP master also invokes the sendSubScanCorrelatorData() on the DataCapturer.

8.5 EVLA WIDAR implementation

8.5.1 Data stream example

No example is available yet. One will appear in this section shortly after the WIDAR backend software is capable of producing a file in the BDF format.

8.5.2 Data rates

TBD

8.6 ~~TBD~~References

[RFC2045] N. Freed, N. Boorenstein, “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”, IETF RFC 2045, November 1996

[RFC2046] N. Freed, N. Boorenstein, “Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types”, IETF RFC 2046, November 1996

[RFC2387] E. Levinson, “The MIME Multipart/Related Content-type”, IETF RFC 2387, August 1998

[RFC2557] J. Palme, A. Hopmann, N. Shelness, “MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)”, IETF RFC 2557, March 1999

[Scott] S. Scott,”Specifications and Clarifications of ALMA Correlator Details”, February 2003.