PROTOCOL SPECIFICATION

Science Data Model 1.0

Revision: 1.0 (October 17, 2010)

Robert Lucas, François Viallefond, & Michael P. Rupen

Contact author: Michael Rupen
mrupen@nrao.edu

National Radio Astronomy Observatory
1003 Lopezville Road
Socorro, NM 87801

v. 1.1

ABSTRACT

This document describes version 1.0 of the Science Data Model (SDM) used by the Atacama Large Millimeter Array (ALMA) and the Expanded Very Large Array (EVLA).

Contents

1. Introduction

This document, together with the corresponding XML schemata, the binary data format (BDF) specification, and the associated BDF XML schemata, define the Science Data Model (SDM). The SDM is the format in which raw science data (including calibration data) will be archived and provided to astronomical observers by the EVLA and ALMA. A given application (e.g., interferometric observations with the ALMA Compact Array) is expected to implement only a part of this format. The generic SDM format, and consequently its application-specific variants, is supported by the Common Astronomy Software Applications (CASA) post-processing software through a CASA-provided Measurement Set (MS) filler.

An SDM dataset is composed of a set of tables represented as XML documents for data exchange and persistence.[1] The amount of actual data produced by various processors (correlators, 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 are grouped into binary large objects (BLOBs), whose format is described in the Binary Data Format (BDF) protocol specification (5). The current document defines the format in which the associated auxiliary and meta-data are stored. A more abstract view of an earlier version of the SDM may be found in Viallefond (6), while Viallefond & Lucas (7) offers an early look at what was then called the ALMA Export Data Format.

1.1. The role of the XML schemata

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 Appendix A). All the documents required for a complete specification are maintained as a single project under a revision control system. 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.

- The role of the XML schemata in implementations of the SDM format is somewhat peripheral, in that the schemata are not required to read or write the SDM documents. Their role is as a kind of "type checker" that can be used to validate the XML parts of an SDM document. Note that such validation requires that the various elements be written in the order specified in the XML schemata. In the present version of the schema **this order does *not* match the logical order given in the individual table descriptions** (§3).

[1] ALMA has also defined the in-memory representation of the SDM. This is not discussed in the current document, although perhaps some ALMA person will add this as an appendix.

- Currently the XML schemata are *untyped*: all fields are stored as strings. A major feature of the proposed "SDMv2" would be to change over to typed schemata. In the current SDM version 1 the logical types of the individual fields are given in the detailed table descriptions (§3) with the various non-standard types described in other sections in this document.

PROPOSAL:
I propose requiring that the order of elements in the schema match that in the individual table descriptions, in hopes of making the actual documents more readable. This should be done in concert with a re-ordering ot those elements in the descriptions, as they are currently far from uniform.

1.2. Protocol definition

In the ALMA project, the XML schemata are auto-generated from a UML document. Similarly, the ALMA project embeds some table documentation directly in the UML, and auto-generates short table descriptions from those UML comments. These are implementation details; the SDM format is *defined* by this document together with the associated XML schemata. Any differences between the UML and this definition should be resolved in favor of the definition. This is an essential point if the SDM is to be more than an internal format, more similar to the FITS standard than (for example) the current VLA's Export Format.

1.3. Typesetting conventions

This document uses a number of typesetting conventions for various information:

- SDM table names are given in mixed case (e.g., SpectralWindow), while Measurement Set tables are fully capitalized (e.g., the MS' SPECTRAL_WINDOW Table).
- Angle brackets are used for strings which must be replaced with specific values when used: e.g., Cal<Result>.
- Column/field names are given in `the teletype font`, as are Web addresses. References to a field in a particular table are given as <table>:<`field`>; e.g., Main:`time`.
- The word 'table' is capitalized when referring to a specific named table: e.g., "Main Table," but "this table."

- Notes specific to the EVLA are given over a 'Navajo White' background.

- Notes specific to ALMA are given over a 'Pale Turquoise' background.

- Queries to early readers are overlaid on a 'Lavender' background.

- Items early readers should check carefully are given in red.

2. SDM Basics

This section introduces the basic properties and content of the Science Data Model. §2.1 gives the overarching philosophy of the SDM: what the SDM ought (and ought not) to include, and some guiding design principles. The following section introduces the elements/framework of the SDM, the structural building blocks used to construct the model. Next, §2.3 gives an overview of the contents and structure of the SDM, and the purpose of and interactions between its various constituent tables. The following section discusses a number of important details, ranging from the use of time ranges to the specification of units and coordinate systems in the SDM. Finally, §2.6 comments on the relationship between the SDM and the CASA Measurement Set.

2.1. Philosophy

The SDM must contain all the information necessary for the astronomical processing of raw data from the telescope. "Raw data" here is defined as data produced in real time during the observations, including the results of any on-line calibration. These raw data include:

- **Binary data:** data produced by a correlator or other high data-rate processors, this represents (we hope!) the bulk of the data volume. These data are saved in the Binary Data Format (q.v.), but their scientific content and interpretation are defined by the SDM's XML tables.
- **Auxiliary data:** monitor data from the telescope during observing, that will be needed for astronomical data reduction. Examples include antenna coordinates and total power measurements.
- **Metadata:** data needed to describe the observing process as executed, and the resulting binary and auxiliary data. Examples include the spectral setup, source information, and (sub-)array description.

The SDM also includes information required to track the observing project of which these particular data are a part. Thus the SDM contains information on the project code(s), public release date(s), etc., which links to and repeats some aspects of the Project Data Model (PDM).

The SDM is *not* intended to include monitor data which are not normally useful for scientific processing; the primary purpose is to allow useful astronomical analysis, rather than to debug the telescope hardware or software.

2.1.1. Design considerations

The data organization of the SDM was motivated by:

- the efficiency of the data-writing process;
- a desire to organize the data in the most suitable form for data reduction (thus minimizing transpositions);
- the minimization of data volume;
- the efficiency of conversion into the internal format used by the main data reduction package (CASA).

2.1.2. Clarity and completeness

All items in the Science Data Model should be defined clearly enough that they can be interpreted by a reasonable programmer and/or scientist implementing the SDM. Any data that cannot be so defined should not occur in the SDM. Data that are needed, but cannot yet be defined fully, should be noted as "TBD", and are not to be considered part of this version of the SDM. These place-holders should be fleshed out (or disappear) in future versions of this document.

Some flexibility is afforded by the Annotation Table, which allows the user to list arbitrary sets of parameter names and values (including 1- and 2-dimensional arrays). Eventually any truly useful data stored in this table should move to permanent tables.

2.1.3. SDM granularity

The format is able to contain an arbitrary amount of data (a data set).

An SDM written to the archive will comprise at most one Execution Block (see §2.3.2). For both ALMA, and the EVLA much smaller (single scan) SDMs will be used to communicate to and from the real-time calibration system (TelCal); this will also be true for ALMA's Look pipeline.

The smallest possible SDM is set by the EVLA and ALMA implementations of the binary data format (BDF), with one file written every subscan; thus there can be no EVLA or ALMA SDMs referring to data on timescales shorter than a single subscan.

The SDM itself can accommodate a wide variety of data: an entire project, parts of a project, one or more Execution Blocks, a subset of an Execution Block,[2] or data on a source from several different projects. This complexity is intended to allow easy access to data in the archive, so that a single SDM can include the results of a single astronomically-driven search.

ALMA will write one SDM per administrative subarray: i.e., a single instance of Control (a single scientific project) will produce a single SDM. That SDM might include:

- one set of antennas producing interferometric data, looking at source S_1, at frequency ν_1, with integration time δt_1;
- another set producing interferometric data, looking at source S_2, at frequency ν_2, with integration time δt_2;
- yet a third set of antennas operating in single-dish mode, each pointing at a different source, at different frequencies, with different integration times.

Later in the same Execution Block all the antennas might come together to form a single set of antennas in interferometric mode; then go their separate ways again; and back and forth *ad nauseum*. All these data can be stored in a single SDM: at a given time each set of antennas would correspond to a different row in the Main Table, with that row pointing to different rows in the ConfigDescription and Field Tables, and to different binary data files (see §2.3.1).

The EVLA will not use the SDM sub-arraying feature, and will instead produce one SDM per set of antennas, i.e., one SDM for each VLA-style subarray in which all the antennas are cross-correlated. In the above case we would have one SDM for the antennas observing $S_1/\nu_1/\delta t_1$; a second for the antennas observing $S_2/\nu_2/\delta t_2$; and a third for the set of antennas operating as single-dishes.[3]

For this document the important point is that the SDM can handle both cases.

[2] E.g., including only the calibrator data.

[3] This last will seldom obtain for the EVLA in practice, since the EVLA antennas are not optimized for single-dish work.

2.2. Structural elements

2.2.1. Tables, rows, and columns/fields

The SDM is based on the concept of a relational data base. The usage is not quite identical, but this does give a rough idea of how the SDM is intended to work.

The SDM consists of a number of tables. These tables contain data which are grouped together to reflect logical relationships (e.g., receiver parameters) and similar time granularity (some data change with every integration, while others are constant from scan to scan or even over an entire observing run). Some tables are **required** – that is, they must be present in every valid SDM. The majority are **optional**, and are not necessarily present in every SDM. However, every table explicitly referenced by another table in a given SDM, must also exist in that SDM.

The tables are composed of columns (also known as fields) and rows. The columns/fields detail which data may be present in that table; the rows give the actual data. Each table must contain at least one row. The complete list of SDM tables and their fields is given in §3. The tables typically contain information related to:

- **hardware characteristics:** e.g., the size of the antenna, and the characteristics of the backend processor
- **configuration of the array:** e.g., the location of the antennas, the receivers being used, and the frequencies being observed
- **antenna tracking:** e.g., the pointing direction, and details of any Doppler tracking
- **the target:** e.g., the position and flux density of the source
- **monitored auxiliary data:** e.g., system temperatures and weather information
- **the overall project:** e.g., the observer's name
- **post-processing:** e.g., which scans are to be used for phase calibration

Some of this information is static or quasi-static (e.g., configuration setups and hardware characteristics), while other information may vary continuously (e.g., antenna pointing and weather data). Again, the tables are arranged so that these different time granularities occur in different tables, so that each table can be filled as slowly as possible (and hence be as small as possible).

The columns/fields of the tables are grouped into three different sections:

- key ;
- required data ;
- optional data.

2.2.2. Keys

A **key** minimally specifies a unique row in a table.

- Each unique row has a unique key.
- A row is unique if all the Required Data and all fields in the composite key (see below) for that row are unique. Rows thus *cannot* differ only in their Optional Data; at least one Required Data element must be different in every row.
- Specifying only some of the fields in a composite key (see below) does not *guarantee* a unique row in the table; although in practice retrieving all matching rows may result in a single match.

Keys are enormously important in determining the structure of the SDM. In the Main Table, for instance, one is *guaranteed* that there is a separate entry for any combination of {`Time, configDescriptionId, fieldId`}. This implies that there may be several Main Table rows at a given time, corresponding to several different (simultaneous) hardware configurations; a single SDM may thus contain information from several basically independent sets of antennas (often called "subarrays").

To put this another way, keys determine how rows may be added to a table. A table may be *indexed* differently – for instance, one might index the Main Table through the key plus the scan number, for efficiency. Such indexing is an *implementation* detail, not part of the model definition.

Similarly there may be other constraints imposed in addition to the basic uniqueness of a key. The primary additional constraint in the SDM is that, when the **time interval** is one of the fields in a key, there can be *no overlap* between that and any other time intervals in any keys whose other fields are identical. In the Receiver Table, for instance, the composite key is made up of three fields: `sourceId`, `spectralWindowId`, and `timeInterval`. At a given frequency a source can have only one flux density (for instance), so the time intervals for a given `sourceId, spectralWindowId` pair may not overlap. [4]

The Main Table is unique in having only the `time` (not the `interval`) in its key; this is done to show that, in this table, the intervals *may* overlap. For ALMA the typical case would be concurrently-operating subarrays, though these will also differ by another key element (e.g., `configDescriptionId`).

Another constraint is that certain non-key fields are also required to be unique, in some cases across SDM boundaries. Here the obvious examples are the various UIDs, which are guaranteed to be absolutely unique everywhere. Thus one can also specify a unique row in the Main Table by

[4]This "no overlap" constraint applies *only* to time intervals which are key fields; it does *not* apply to times which are required or optional data (see §2.2.3).

giving the `dataUid`. This is not generally an interesting way of indexing a table, so the UID are not used in keys.

Many keys are **composite**, meaning that they consist of multiple fields. Examples include those for the Main and Focus Tables. The ordering of fields in a composite key, as given in the individual table descriptions (§3), is important for efficiency in accessing the table:[5] if one thinks of a table as a multi-dimensional array with the fields of the key as its indices, the last field should correspond to the fastest-varying axis. *The individual fields in a composite key are considered required data when determining uniqueness; that is, it is permissable for two rows in a table to differ only by one or more fields in their composite keys.* Note however that one cannot have two rows which differ only by an integer identified (e.g., `sourceId` in the Source Table), or by a tag (e.g., `antennaId` in the Antenna Table), if all the data columns are identical.

Other keys consist of a single field, as in the ConfigDescription Table. Those single-field keys are referred to as **tags**. [6] The actual value of a tag is unimportant (and ideally should be unknowable); a tag is simply an index into a table. An example is the SpectralWindow Table, where `spectralWindowId` suffices to specify a single, unique row in that table. There must be a unique tag for every row in a given table with a unique set of required fields. Tags are stored on disk as `<TableName>_<unsigned_integer>`. For instance, `Antenna_12` is an example of an `antennaId`.

Note that the Measurement Set has no concept of tags, although they implicitly exist for MS tables such as POLARIZATION which have no explicit keys.

Tags are always named as `<tableName>Id`, with type `<TableName>Tag`. Unfortunately key fields which are *not* tags sometimes follow the same naming convention: e.g., `sourceId` is only one field in the composite key for the Source Table, and is an integer. Tags are pointers to rows; they are unique, and should *only* be used internally. One might say a tag is the moral equivalent of a row number. Integer IDs by contrast are unique "local keys" – they are unique in a local context. For instance, the `feedId` is unique once `antennaId`, `spectralWindowId`, and `timeInterval` are specified.

In the SDM, if an ID (either tag or integer) occurs as a required or key value in one table, the table thus referenced *must* exist, for the first (referring) table to be valid.

The use of IDs in the SDM is fundamentally different from that in the Measurement Set. In the MS an ID is always an integer, not a tag, with numbers ≥ 0 corresponding directly to numbered rows in the corresponding table. Negative numbers are then reserved for interesting "magic values". For instance, in the MS' FIELD Table, `SOURCE_ID`$= -1$ means there is no corresponding source

[5]Note that this is *not* part of the SDM definition, simply a comment on efficient implementation. The order of fields must match that given in the XML schemata, which does *not* match that given in the individual table descriptions in this document.

[6]Note that the single-field keys (tags) of one table, may appear as elements in the *composite* keys of other tables – see for instance the Main Table, whose composite key includes two tags referencing other tables.

defined in the optional SOURCE sub-table; while in the SOURCE Table, `SPECTRAL_WINDOW_ID`$= -1$ means that this row is valid for all spectral windows. In the SDM the actual *values* of even integer IDs are not interesting in themselves; they are to be used only as labels, and "magic values" are not allowed. This means for instance that there is no guarantee that the rows corresponding to higher `sourceId`s actually occur later in the Source Table; nor is it guaranteed that `sourceId=5` exists just because `sourceId=4` and `sourceId=6` do.

2.2.3. Required vs. optional data

As so often in the SDM, **Required Data** means rather more than it says:

- Required fields must have explicit values in order for the corresponding row to be valid.
- *In addition*, a set of required fields specifies a *unique* row in the table. One row may *not* differ from another row in a table, solely in its "Optional Data"; each row *must* differ in one of the required data fields.
- Each such unique row has a corresponding unique key.

Optional Data fields are those which can but do not have to have values for a given row, and which are *not* considered when determining uniqueness. Two rows in a table *cannot* differ only in their optional data – every row must differ in at least one required field. Thus for instance one cannot have two rows in the State Table which differ only in the weights used when combining signal and reference data.

This usage fundamentally links the concepts of "unique" and "required" in the SDM. This is unfortunate but impossible to change at this late date.

The documentation for the individual tables (§3) groups items which are logically related. Note however that the order of the fields must match the XML schemata; and that currently those schemata do *not* match the order given in this document.

2.3. Overview of the SDM

This section gives an overview of the tables in the Science Data Model, beginning with those which describe the structure of the SDM itself.

2.3.1. Basic structure: the SDM File and the Main Table and its associates

For a given instance of the SDM, the tables (and the number of rows in each) are listed in the **SDM File** (sometimes called the **Container File**). The SDM File thus serves as a "table of contents" for a particular SDM.

The **Main Table** is the "master" table for an exported data set. This table gives the links between the binary data and the associated auxiliary and metadata. Each row has three key fields (see § 2.2.2 for the definition of a key): the `time`; the `configDescriptionId`, pointing to a description of the telescope/backend setup; and the `fieldId`, pointing to a description of the position towards which the telescope was pointing. Thus for instance one may use the same hardware configuration for many different scans and sources, by referring to the same `configDescriptionId` in the Main Table.

The binary data themselves are stored in separate files, referenced by the `dataOid`. There is a separate binary data file (BLOB) for each row in the Main Table, corresponding to a single (sub)scan for a given hardware setup and field position. The format of those binary data files is defined by the BDF documentation (5) and BDF XML schemata. The shape of the binary data stored in a file is described in the Processor Table referenced by the **ConfigDescription Table**. [7] The latter table also implicitly gives the ordering of baselines for correlated data stored via the BDF – see the BDF documentation.

The Main Table refers directly or indirectly to a number of other tables; those tables are thus *required* for any SDM used to record an actual observation. Tables not referenced by the Main Table are not required for an exported data set. See Table 1 for a listing of all required and optional tables.

There may be at most one table of a given type per SDM. For example, in a given SDM there can be only one Main Table, one Antenna Table, etc.

2.3.1a. The Calibration Data Model

A given SDM need not contain a Main Table; this is the case for SDMs which describe calibration results rather than an actual observation. The **Calibration Data Model** (CalDM) is used to store calibrations reduced either in real time or in post-processing by observatory science operations. The goal is to allow sharing calibrations between projects, with a secondary aim of providing useful feedback on the state and stability of the instrument to the observatory staff. The CalDM is described in detail in Lucas & Viallefond (2), but is summarized very briefly here to give some feeling for the relation between the CalDM and the SDM. The prose here is taken primarily from §3 of the CalDM document.

[7] Note that the *order* of those axes is significant!

Table 1. Tables referenced by the Main Table

SDM Tables		
Referenced:		
Main		
Antenna	Field	SpectralWindow
ConfigDescription	Pointing	State
DataDescription	PointingModel	Station
ExecBlock	Receiver	Subscan
Feed	Scan	SwitchCycle
Not referenced:		
Annotation	Focus	Source
CalDevice	FocusModel	SysCal
DelayModel	FreqOffset	SysPower
Doppler	GainTracking	WVMCal
Ephemeris	SBSummary	
Flag	Source	
Sometimes referenced:		
AlmaRadiometer	required for radiometers; not allowed for others	
Beam	required for single dish or mosaicked data	
CorrelatorMode	required for correlators; not allowed for others	
Holography	required for holographic observations; not allowed for others	
Polarization	required for astronomical observations; not allowed for holographic observations	
SquareLawDetector	required for total power or noise detectors; not allowed for others	
Mandatory:		
SDM File		

Note. — The 16 tables referenced (directly or indirectly) by the Main Table are **mandatory** for SDMs used to archive astronomical observations. The 16 tables *not* so referenced are optional. Six additional tables are referenced by the Main Table in special circumstances, and hence are sometimes mandatory, and sometimes not allowed. Finally, the SDM File is required for *all* SDMs. CalDM tables are not included in this list – see Table 2.

A calibration result data set is a 'mini-SDM' that includes one or more calibration results, each of which is a set of rows in a Cal<Result>Table. It is similar to a 'standard' SDM dataset containing scientific observations, in the sense that it conforms to the SDM schema; but a CalDM data set need not contain many of the tables mandatory in a 'standard' SDM (e.g., the Main Table and its subsidiaries), and in particular need have no direct links to the actual binary data. There is one Cal<Result>Table for each *type* of calibration result, and each such table will, in general, contain an arbitrary number of results of that type. The actual tables in a calibration data set are:

1. at least one one or several Cal<Result>data tables, describing the actual results of the calibrations.
2. one CalData Table, describing the connection to the data that were obtained from the telescope and reduced by the calibration software to produce the calibration result. The data themselves may be accessed using the `ExecBlockID` field of this table.
3. one CalReduction Table, describing the generic parameters characterizing the reduction and its result.

No other SDM tables are included in the stand-alone calibration results. In this scheme calibration results can be integrated in SDM data sets when these data sets are built (either during observations, through real-time calibration, or in fetching data from the archive). However, there is no direct link through identifiers between the CalDM tables and the rest of the SDM tables. The relationship is rather a simple juxtaposition. The only exception is the insertion of `configDescriptionId` in the CalData Table, to further specify the data that were used to obtain the result (e.g., a specific set of spectral windows).

Although the CalDM document (2) describes the overall philosophy and use of the CalDM tables, those tables are considered part of the SDM, and the detailed table definitions here supercede those given in that document. The tables of the CalDM are listed in Table 2, and noted explicitly in the individual table descriptions in §3.

QUERY:

- Does a particular CalDM, like a 'standard' SDM, require an SDM File to keep track of which tables are present in that CalDM?
- Must all tables in a particular CalDM be present in the same directory, in order to be read by the CASA filler?

I believe the answer to both questions is ***YES***.

Table 2. List of CalDM Tables

CalDM Tables		
CalAmpli	CalFocus	CalPointingModel
CalAtmosphere	CalFocusModel	CalPosition
CalBandpass	CalGain	CalPrimaryBeam
CalCurve	CalHolography	CalSeeing
CalDelay	CalPhase	CalWVR
CalFlux	CalPointing	
CalData		CalReduction

Note. — The tables listed here are all Cal<Result>Tables, except for CalData and CalReduction. All CalDMs **must** include one CalData, one CalReduction, and one or more Cal<Result>Tables.

Note that, despite its name, the CalDevice Table is *not* part of the CalDM.

2.3.2. Temporal hierarchy: projects and SBs and scans, oh my!

The SDM uses a hierarchy of "observing objects" derived from the Scientific Software Requirements documents for ALMA and the EVLA (Fig. 1). These correspond to different granularities of time, and are defined as follows.

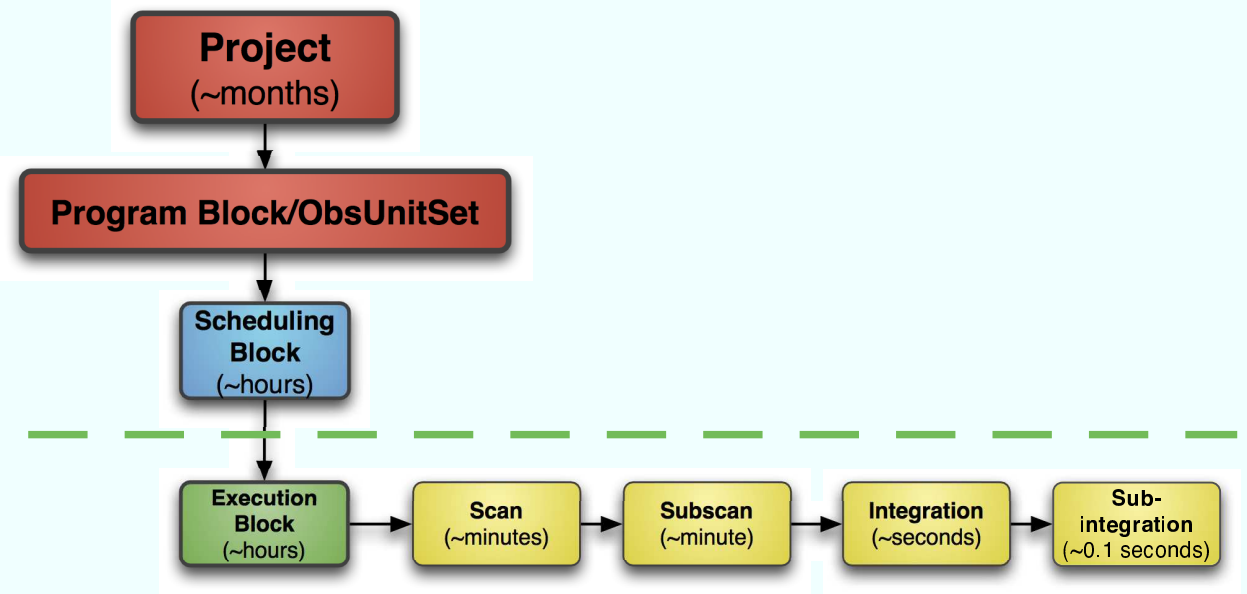

Fig. 1.— Hierarchy of "observing objects" used by the SDM. Each level must include at least one at the next level underneath – for example, a Scan must include at least one Subscan. A typical timescale for a given object is given as well (e.g., $\sim$ 1 second for an Integration). The objects shown above the dashed line exist before observing time; those shown horizontally below that line are created at observing time.

A **Project** is a scientifically independent subset of observations. Generally each Project represents part of a Proposal approved by the observing program committee. The Project is referred to in the SDM through the ExecBlock (`projectId`) and SBSummary (`projectUID`) Tables.

A Project consists of a number of (possibly nested) **ObsUnitSet**s. The ObsUnitSet is referred to through the SDM SBSummary (`obsUnitSetId`) Table. The EVLA has **Program Block**s rather than ObsUnitSets; the only difference is that Program Blocks cannot be nested.

An ObsUnitSet or Program Block is made up of a number of **Scheduling Block**s (SBs). A Scheduling Block is the fundamental, un-interruptible (in the sense that it may be aborted, but not restarted in the middle), atomic unit of observing. SBs are described in the SDM's SBSummary Table.

When a Scheduling Block is actually observed, it is called an **Execution Block**, which is the subject of the SDM's ExecBlock Table. An Execution Block thus represents the actual execution of a Scheduling Block. A single Scheduling Block may be observed multiple times, leading to multiple Execution Blocks; the `execBlockNum` element is intended to keep track of such repeats.

A Scheduling or Execution Block is made up of a number of **scans**. A scan is a sequence of one or more consecutive observations that share a common goal. **Subscans** represent those individual observations. There is always at least one subscan per scan. For example, one might use a 5-point pattern to determine pointing corrections. Individual subscans would refer to individual pointings within the 5-point pattern; the scan would consist of all subscans in that pattern. Scans and subscans are described in the SDM's Scan and Subscan Tables.

ALMA note:
In ALMA a subscan is the minimum amount of data taken by executing a single Control Command Language (CCL) command.

Scan numbers increment starting from 1 in each Execution Block. Subscan numbers increment starting from 1 in each scan. Scan numbers are thus unique within an Execution Block, while subscan numbers are unique within a scan.

An **integration** is the basic recorded unit of data. An integration consists of one or more **sub-integrations**. The number of integrations in a scan, and the number of sub-integrations per integration, is given in the Subscan Table.

ALMA note:
ALMA records channel-averaged data every sub-integration, and "normal" (high spectral resolution) data every integration.

EVLA note:
Currently the EVLA does not write channel-averaged data, and hence the number of sub-integrations per integration is always 1.

2.3.2a. The BIN axis: switched data and pulsar phase bins

Various switching approaches are often used in single dish observations to remove spurious effects resulting from variations in the gain and or in the atmosphere. The Subscan Table (`subscanMode`) is used to keep track of which switching is being used (e.g., frequency switching, nutator switching), while the SwitchCycle Table holds the corresponding position and/or frequency offsets. Slow switching (e.g., position or phase switching) leads to different integrations for the different ON and OFF parts of the switching cycle. Generally this should be done through the use of multiple subscans, as in referenced pointing with the EVLA. Switching on timescales faster than an integration

(e.g., chopper wheel calibration) is handled, where necessary, through the **BIN** axis of the binary data (see 5).

The BIN axis is also used to separate data from different pulsar phase bins, but this is not yet implemented in the SDM.

2.3.3. Arrays and antenna positions

The ConfigDescription Table lists all the antennas being used to obtain science data (including calibration data).[8] The Antenna Table gives basic antenna information (name, make, dish diameter) as well as the position of each antenna's pedestal reference point relative to its station (`position`), and the offset of the antenna feed reference point relative to the pedestal reference point (`offset`). The Antenna Table also gives pointers to the Station Table, which stores the names and ITRF positions of the stations. Finally, the Feed Table gives an optional position offset (`position`) between the feed and the antenna's feed reference point. The final phase reference position for a given feed/antenna combination is therefore:

$$\texttt{Station}:\texttt{position} + \texttt{Antenna}:\texttt{position} + \texttt{Antenna}:\texttt{offset} + \texttt{Feed}:\texttt{position}$$

The idea is that most of these positions and offsets will be stable. Ideally only the `position` in the Antenna Table (the offset of the antenna pedestal from the nominal station position) must be re-measured when the antenna is moved to a new station.

The SDM does **not** store the *(u,v,w)* coordinates for interferometric baselines, which must be calculated by the post-processing system based on the phase reference positions of the station/antenna/feed combinations.

One SDM may store the data from multiple groups of antennas observing at the same time. At a given time, each group would be described by a different row in the Main Table, with a different corresponding row in the ConfigDescription Table, listing all the antennas. The Antenna Table would store the information for antennas in all of the groups; switching antennas from one group to another would simply require a new row in the ConfigDescription Table.

EVLA note:
The EVLA does not currently envision using this capability to the full, but will instead write a separate SDM for each group of antennas. However, the EVLA *will* keep a single SDM as antennas come and go from an EVLA subarray.

Orbiting antennas are not currently handled by the SDM.

[8]The ConfigDescription Table also shows which (if any) of these antennas are being phased up for VLBI or other purposes.

2.3.4. Antenna properties

Currently the SDM holds very limited information on antenna properties, such as the antenna diameter (in the Antenna Table). Forward gains, anntenna efficiencies, and the like, are *not* stored. In the Measurement Set, such information is supposed to be held in the Beam Table, but that has yet to be defined (K. Golap 2009, priv. comm.).

2.3.5. Feeds, receivers, and backends

The Feed and Receiver Tables are used to keep track of the characteristics of the corresponding physical devices on an antenna. Feeds are numbered from 0 on each separate antenna for each spectral window; consequently, `feedId` should be non-zero only in the case of feed arrays, i.e., multiple, simultaneous beams on the sky at the same frequency and polarization.

The State Table, referenced directly from the Main Table, keeps track of the insertion of any calibration devices (e.g., quarter-wave plate, solar attenuator) and whether or not the receiver is getting radiation from the sky. The characteristics of the calibration devices (e.g., temperatures of cold loads) are stored in the CalDevice Table; note that these characteristics are assumed constant over an entire spectral window.

The SDM allows for multiple backends, referred to as **processors**. The Main Table refers to a different row in the ConfigDescription Table for each processor in use at a given time. The ConfigDescription Table in turn references the Processor Table, which gives the type of processor and points to a specific row in the appropriate table (`modeId`). Currently there are three options:

1. CorrelatorMode Table: used for a correlator (interferometric data). This table is particularly important, as it also stores the order of axes in the binary data block (BLOB) (see 5).
2. SquareLawDetector Table: used for total power detectors.
3. AlmaRadiometer Table: used for water vapor radiometers.

2.3.6. Polarization information

The Polarization Table keeps track of which polarization products are recorded, while the Feed Table stores the polarization types and responses of the receptors used in calculating those polarization products. The only explicit polarization calibration data stored in the SDM (apart from the CalDM) is the cross polarization delay in the GainTracking Table.

2.3.7. Directions, pointing, and delay/phase centers

The primary pointing position of each antenna is stored in the Pointing Table. The Pointing Table stores both the commanded pointing direction (`pointingDirection`) and the pointing direction reported by the antenna (`encoder`). The commanded pointing direction is the sum of the global and local (referenced pointing) pointing models, stored in the PointingModel Table, the target direction, and any additional offsets to that target direction (e.g., when mapping the antenna beam):

$$\texttt{pointingDirection} = \texttt{pointingModel} + \sum \texttt{assocPointingModel}_i + \texttt{target} + \texttt{offset}$$

where `pointingModel` is the model referenced directly from the Pointing Table; the $\texttt{assocPointingModel}_i$ are the associated models referenced through `assocPointingModelId` in the PointingModel Table; and `target` and `offset`[9] are given in the Pointing Table. As an example, consider a VLA K-band scan using second-order referenced pointing. There would be four pointing models: the default, global pointing models from array operations, stored separately for the station and for the antenna on that station; the first-order pointing corrections derived at X band; and the second-order corrections derived directly at K band. The Pointing Table's `pointingModelId` would reference the row in the PointingModel Table which contains the second-order (K-band) corrections. That row would in turn use `assocPointingModelId` to reference the row containing the first-order (X-band) corrections; which would have its own `assocPointingModelId`, referencing the row containing the antenna pointing model from array operations; which itself would have yet another `assocPointingModelId`, referencing the station pointing model, again from array operations. The sum of all four models would be added to the `target + offset` to derive the requested `pointingDirection`. If the antenna is on-source, the `encoder` position will closely match this commanded `pointingDirection`.

QUERY:

- Nuria points out that PointingModels were originally supposed to be user-defined, and relevant only to the current SB. If this is true I do not know how to ensure one can recover the pointingDirection, which includes terms from the model used by array operations.

Pointing offsets on timescales less than a subscan, generally used for switching on- and off-source for single dish work, are recorded as separate subscans. For very fast switching (e.g., when using a nutating subreflector) the ON and OFF data are stored in different bins of the binary data using the BIN axis; see §2.3.2a. In cases where the correction used is interesting but the ON and

[9] `offset` is an offset in horizontal (Az-El) coordinates. The Pointing Table also has a `sourceOffset` field, to allow specifying offsets in other coordinate systems (e.g., equatorial) for on-the-fly mapping and the like.

OFF data are not stored separately, the switching mode can still be recorded in the `subscanMode` elemenent of the Subscan Table.

Note that each antenna is assigned a separate row in the Pointing Table. Thus the SDM can handle the ALMA case where many telescopes are commanded to observe different positions at the same time (e.g., to facilitate single-dish observations of a large area on the sky).

The individual feeds on a given antenna may give pointings offset from this antenna pointing direction. These individual feed pointing offsets are stored in the Feed Table via `beamOffset`. `beamOffset` itself is an array, with one offset given for each receiver/receptor used with that feed.

The directions and offsets given in the Pointing, PointingModel, and Feed Tables represent those actually used during the observations. Pointing results derived from the observations – which may or may not have been applied in real time – are given in the CalPointing and CalPointingModel Tables, which are part of the CalDM.

The PointingModel Table gives the pointing model in terms of named coefficients.

ALMA note:
ALMA uses coefficient names and meanings taken from the TPOINT pointing analysis program, and described in Mangum (3).

EVLA note:
The EVLA will likely *not* use the TPOINT naming conventions, but the actual coefficient names and meanings are TBD.

The Field Table gives the delay and phase center, and a reference direction (`delayDir, phaseDir, referenceDir`). These need not be the same as the pointing direction discussed above. For single-dish observations the reference direction should give the reference direction for position-switching. For interferometers the reference direction is the correlated field center, and is likely the same as the delay and/or phase center.

The Field Table optionally refers to an astronomical source described in the Source Table. The Source Table gives the direction (sky position) and proper motion of the source, and optionally its three-dimensional position (e.g., for satellites). The Field Table describes how the array observed; the Source Table describes an astronomical object. Put another way, the Field Table tells you about the *data*, while the Source Table tells you about the *sky*. The Field Table is mandatory for all SDMs use to archive data, while the Source Table is optional.

2.3.8. Frequencies and tuning

The frequency range covered by a receiver is given in the Receiver Table (`frequencyBand`). This frequency band is broken up into one or more **basebands** for processing by the backend(s). These

basebands are described (if at all) in the specific processor tables; currently the most developed in this sense is the CorrelatorMode Table, which records information about the filter used in creating the baseband. These basebands need not be contiguous. The basebands themselves are split into **spectral windows**, which are contiguous sets of frequency **channels** which share common physical properties and make sense to calibrate together.[10] These spectral windows are described in the SpectralWindow Table, primarily in terms of the net sideband, the conversion between channels and frequencies, and the windowing (spectral smoothing) function (e.g, Hanning). One may optionally record the frequency resolution and effective bandwidth of the frequency channels as well. The SpectralWindow Table may also refer to the Doppler Table, meaning that the instrument is Doppler tracking on a given line and velocity (given in the Source Table). The FreqOffset Table gives any additional antenna-dependent frequency offsets to the observing frequency, as might be used e.g., in Doppler tracking with a very large array.

The SwitchCycle Table records any frequency steps used as part of a switching cycle. These are added to the observing frequencies described above. For very fast frequency switching the ON and OFF data are stored in different bins of the binary data using the BIN axis; see §2.3.2a.

There is currently no way to describe the storage of anything other than spectra (e.g., lag spectra).

FIGURE: receiver band, baseband, spectral window, subband

Receiver tuning information is split amongst several tables. The Receiver Table records the nominal local oscillator (LO) setups and the receiver sideband. Antenna-based LO frequency offsets (as might be used for anti-aliasing and sideband rejection) are stored in the GainTracking Table.

2.3.9. Calibrations applied during an observation

The SDM distinguishes between calibrations which are *measured* during an observation, and those which are actually *applied* in real time. Measured values are stored in the CalDM tables. Calibrations actually applied in real time are spread throughout various 'standard' SDM tables, as described in the next sections.

2.3.9a. Delays, rates, and the interferometer model

The SDM stores a number of different delays in the DelayModel and GainTracking Tables. The split between tables is set by the physical origin of the delays. Delays which should be the same

[10] For some detectors (notably the ALMA and EVLA correlators) the spectral windows are formed from adjacent, possibly overlapping **subbands** stitched together to make the spectral window. Eventually the SDM should track this information. Initially the EVLA will produce a separate spectral window for each subband – i.e., each EVLA spectral window will initially consist of one and only one subband.

for all LOs, receivers, feeds, backends, and frequencies, are stored in the DelayModel Table, which thus includes atmospheric, geometric, and clock delays. These delays are stored as polynomials in time. The GainTracking Table stores those delays which vary with electronics or frequency – LO propagation delays, cable delays, antenna-based delay offsets, and the like. These delays, for a given row in the table, are constant with time.

Both the DelayModel and GainTracking Tables record the delays actually applied at the time of the observations. Delays *measured* during an observation are given in the CalDelay Table, which is part of the CalDM. In some cases of course those measured delays will be applied in real time, thus leading to new rows in the DelayModel and/or GainTracking Tables.

The total delay applied is the sum of all the delays given in the DelayModel and GainTracking Tables. Both tables are referenced *implicitly* – i.e., given a set of hardware, one must search these tables directly to find out which delay entries apply.

The *origin* of these delays – for instance, the version of and inputs to CALC, or the weather model – is not currently recorded in the SDM.

2.3.9b. Water vapor radiometry and real-time atmospheric phase corrections

The WVMCal Table gives the coefficients used to determine pathlength variations based on water vapor radiometer (WVR) data. The ConfigDescription Table records whether these atmospheric phase corrections (APCs) have actually been applied to some or all baselines, and allows for the possibility of saving both corrected and uncorrected data.

As usual, real-time determinations of the conversion between observed WVR values and the pathlength are stored in the CalDM (CalWVR Table).

2.3.10. Other data measured during an observation

In addition to the actual data, instrument characteristics, and calibration information, the SDM records a variety of auxiliary data taken during an observing run.

- **Temperatures**: Antenna, receiver, sky, and calibration (noise tube) temperatures, measured through the regular feeds, are stored as spectra in the SysCal Table. The measurement times are stored in this table as well, so they need not be related to the times attached to the main data, and may in fact be different for the different temperature data, or for different antennas/feeds.
- **Water vapor radiometers**: WVR data are stored as *binary* data in a separate binary data stream (BLOB), with the `processorType` in the ConfigDescription Table set to RADIOMETER. The radiometer itself is described in the AlmaRadiometer Table.

- **Auto-correlations**: Some correlators (e.g., EVLA's WIDAR) produce auto-correlations (either narrow or wideband) which are independent of the 'standard' correlations which form the bulk of the data. Thes auto-correlations should be stored in separate binary data streams (BLOBs) described by independent rows in the Main Table.
- **Weather data**: Pressure, humidity, and other weather information are stored in the Weatber Table, which again allows time stamping independent of that of the main data. This table also gives the `stationId` of the originating weather station, which may or may not correspond to an astronomical antenna.
- **Sampler information**: Various information about the sampler(s) is needed for the quantization (Van Vleck) correction for a digital correlator. Currently the SDM does not store this information, but simply tracks whether the quantization correction was done in real time (confusingly stored in the SpectralWindow Table, as the optional Boolean field `quantization`).

2.3.11. Blanking and flagging

The newly-defined Flag Table contains flags from the on-line system; additional blanking and flagging information is carried with the binary data, as described in (5). A few SDM tables still use the `flagRow` field to denote rows which should be ignored; this is gradually being phased out, in favor of simply not writing those rows in the first place. The SysCal and Weather Tables also allow flagging individual fields as meaningless.

2.3.12. Observing logs

Operator comments and other logging information should be entered in the ObservingLog Table, which has not yet been adopted.

2.3.13. Reaching outside the SDM: purposes and intents

The Science Data Model must contain some information beyond the data collected and hardware setup used in real time. In particular, the SDM must reference the project of which these observations are a part, and pass information about the observer's intentions on to post-processing.

The first requirement relates to the **Project Data Model** (PDM). The Science Data Model is one of several models used in the ALMA project, and at least partly shared by the EVLA. The *Project* Data Model describes an astronomical observing project, covering the proposal and observing preparation stages (in ALMA referred to as Phase 1 and Phase 2), and the intermediate stage of reviewing and approving the project. The SDM contains some information taken from the

corresponding PDM, as well as providing direct links in the ExecBlock and SBSummary Tables. A document describing the PDM is being developed currently.

QUERY:
Does such a document now exist? What is the appropriate reference?

The main method for passing information to the post-processing software is the use of **intents**. These indicate the intended use of the data: this scan should be used for gain calibration, this subscan is a particular part of a tipping scan, and so forth. At the highest level the basic type and scientific goals of a scheduling block are stored in the SBSummary Table. Scan and subscan intents are stored in the corresponding Scan and Subscan Tables. These are all enumerations (see §4) to ensure they can be interpreted by the post-processing software and associated pipelines.

2.4. Missing and extraneous information

The SDM has yet to be used regularly for a wide range of observations, and is almost certainly missing some information which is vital for some observers. The Annotation Table is intended to store necessary information which has not yet been incorporated in the standard SDM tables.

The opposite point is that the SDM can store a great deal of data which may not actually be needed. Table 1 shows which tables are actually *required* for an SDM used to hold observational (as compared to simply calibration) data; clearly many will be used only when needed for a specific observation. One may also store rows in an individual table which are never accessed – for instance, one may store information about all the antennas in an array in the Antenna Table, but only observe with a small subset of those. There is no requirement that every row – or even every table – of an SDM be used; tables and/or fields may exist which are never referred to, directly or indirectly, by other tables.

2.5. SDM details

2.5.1. Time ranges in the SDM

Time[11] and time intervals are referenced frequently throughout the SDM (Table 3). Unfortunately the names and usages of the various time-related fields are not entirely consistent. Some field names are redundant – for instance, the (`startTime, stopTime`) pair, the (`time, interval`) pair, and `timeInterval` all carry the same information, in slightly different ways. Further, the

[11]Note that this documentation often simply uses 'time' to mean 'time and date', usually stored in a specifically defined data type.

field names are not used consistently – for example, the (`startTime, stopTime`) pair is used for both planned and actual observed times. This is a vestige of the *ad hoc* development of the SDM, and the rather independent development of the CalDM tables, which tend to use more descriptive field names (e.g., `startValidTime, startTimeObserved`).

Time intervals in the SDM are considered "closed" on the early side, and "open" on the late side: i.e., a time range from 12 to 14 nanoseconds includes 12.000 nanoseconds but does not include 14.000 nanoseconds. This is important because time intervals in keys may not overlap (see §2.2.2).

2.5.1a. Planned vs. observing time

The SDM maintains a distinction between the planned observing time, as given in the schedule, and the actual observing time, as measured during the actual observation. The planned times are given in the Main, ExecBlock, and SBSummary Tables; the actual observed times are stored in the Scan and Subscan Tables.

The most accurate and reliable times are stored with the binary data using the BDF, which gives `ACTUAL_TIMES` (the time centroid associated with the data) and `ACTUAL_DURATIONS` (the total amount of time used in deriving the data), as well as `schedulePeriodTime` (the midpoint and duration of the interval, *not* accounting for blanking, to be used as a fallback if `ACTUAL`$*$ don't exist). The times given in the Scan and Subscan Tables may differ from those in the binary data, because the BDF allows storage of different times for different antennas, baselines, spectral windows, etc. The time ranges stored in a row of the Scan or Subscan Table should correspond to the period from the earliest start to the latest end time for the (sub-)scan, as given in the binary data.[12]

2.5.1b. Time ranges with unknown end times

Time ranges in the SDM are given either as a central time plus a duration (`time, interval`; `timeInterval`) or as a start plus an end time (`startTime, endTime`). In some cases however the ending time is not known. The convention for the SDM in these cases is as follows.

1. The end time for any time range which logically terminates at the end of a Scheduling Block (e.g., the last scan) should be set to the (observed) end time of that Scheduling Block.
2. In cases where the ending time is truly not known:

[12]The sdmDataHeader of the BDF also has the element `startTime`, which gives the time at which acquisition of the data in this BLOB was started (i.e., the beginning of the first data block in the first (sub-)integration). This is an observed (as contrast with planned) time, and should match the starting time given in the row of the Subscan Table corresponding to the BLOB.

Table 3. The names and usages of fields related to time in the SDM.

Meaning	Field name	Tables
Nominal (planned) period covered by this row	`time` [key], `interval` [required]	Main
	`startTime, EndTime` [required]	ExecBlock
	`sbDuration` [required]	SBSummary
True (observed) period covered by this row	`startTime, endTime` [required]	Scan, Subscan
Time at which a quantity was measured	`time` [required]	Antenna
Time over which data were collected for this row	`startTimeObserved, endTimeObserved` [required]	CalData
Time over which this row is valid (intrinsic to measurement or report)	`time` [required], `interval` [optional]	Annotation
	`timeInterval` [key]	CalDevice, DelayModel, Feed, Focus, FreqOffset, GainTracking, Pointing, Receiver, Source, SysCal, WVMCal, Weather
Time over which this row is valid (assigned by user)	`startValidTime, endValidTime` [required]	CalAmpli, CalAtmosphere, CalBandpass, CalCurve, CalDelay, CalFlux, CalFocus, CalFocusModel, CalGain, CalHolography, CalPhase, CalPointing, CalPointingModel, CalPosition, CalPrimaryBeam, CalSeeing, CalWVR
Temporal boundaries of Chebyshev polynomial	`startValidTime, endValidTime` [required]	CalCurve
Origin time for polynomial expansion	`timeOrigin` [required]	DelayModel
	`time` [required]	Field
Time over which rms noise was calculated	`integrationTime` [required]	CalPhase, CalSeeing

- For (`startTime, endTime`) pairs, `endTime` should be set to the greatest long long integer.
- For `timeInterval` and (`time, interval`) pairs, the "central" time should be set to the *starting* time, while the duration should be set so that the starting time plus the duration equals the greatest long long integer.

These conventions contrast with those of the Measurement Set, where intervals may take two special values:

- 0: a constant value with no time dependence
- < 0: valid until re-defined (time-stamped but with undefined or unknown period of validity)

The CASA SDM filler must translate the SDM into the MS convention.

2.5.2. Units and coordinate systems

The SDM stores data in a variety of units, using a variety of data types. The default units are given in Table 4. Coordinate systems are specified in the individual table descriptions (§3); generally the default may be over-ridden for a given row by filling in an optional field (e.g., `directionCode` and `directionEquinox` in the Source Table).

Several unusual data types are used in the SDM:

ArrayTime Time in nanoseconds since 17 November 1858 00:00:00 UTC, the beginning of the modified Julian Day (MJD). All times are International Atomic Time (TAI). A specific TAI time differs from the corresponding UTC time by an offset that is an integral number of seconds.

ArrayTimeInterval This specifies a time range, written in the XML as:
`<midPoint> <interval>`
where `<midPoint>` is an ArrayTime specifying the mid-point of the time range, and `<interval>` is the duration of the time range in nanoseconds. See §2.5.1b for a discussion of open-ended intervals.

Interval An interval is simply a duration, in nanoseconds.

Multi-dimensional arrays These are written in the XML as:
`<NumDimen> <Dim1> <Dim2> ...<DimN> <Value1> <Value2> ...<ValueM>`
where `NumDimen` is the number of dimensions, `M` is the product of all the dimensions (`Dim1` $\times$ `Dim2` $\times$ ...$\times$ `DimN`), and the `Values` are given with the last axis varying fastest. Thus an array `X` given as `3 1 2 3 A B C D E F` would have elements

`X[0][0][0]= A, X[0][0][1]= B, X[0][0][2]= C,`
`X[0][1][0]= D, X[0][1][1]= E, X[0][1][2]= F`
assuming 0-based indexing.

Tag Written as `<TableName>_<unsigned_integer>`. E.g., `Antenna_12` is an example of an `antennaId`.

2.5.3. Strings and enumerations

The SDM enforces consistent usages for various field values by the use of **enumerations**, which in the SDM are basically strings which can only take on certain restricted values. These enumerations are given in §4. These enumerations are *global*, in the sense that they are shared by all tables in the SDM.

NOTE: Many of the same enumerations are used in the other data models, and throughout the ALMA and EVLA software. Changing the possible values of an enumeration requires careful consultation.

Some fields are strings rather than enumerations. In this case the field may consistent of any string – spaces, odd characters, and the like are allowed, if they do not disrupt the XML (e.g., one should avoid strings like "</catalog>" in a `catalog` field).

2.5.4. Defaults and special values

Where important, default values for optional fields are given in the individual table descriptions. Required and key fields have no defaults: they must be given explicitly.

Currently the SDM defines no "special" values (e.g., blank or 0 means XXX). This contrasts with the MS usage (§2.6). The only exception is the specification of time ranges with unknown end times, discussed in §2.5.1b.

2.5.5. OIDs and UIDs

UID stands for *unique identifier*; OID, for *Object ID*.[13] The SDM is moving to using UIDs throughout, to provide unique identifiers for Projects, ObsUnitSets, Execution Blocks, Scheduling Blocks, individual SDM tables, and binary data.

UIDs are guaranteed to be globally unique – e.g., no two archives can assign the same UID to different data. UIDs are based on the Uniform Resource Identifier (URI) syntax (RFC3986).

[13]The discussion in this section is based primarily on Wicenec (8) and Meuss (4).

- A URI is a sequence of characters from a very limited set, i.e., the letters of the basic Latin alphabet, digits, and a few special characters.
- The general layout of a URI looks like: `<first>/<second>#<third>?<fourth>`

The hierarchical structure of URIs is used in the UIDs to discriminate more global from more local parts. The prefix `uid://` is used to distinguish the UIDs from other URIs.

ALMA note:
For ALMA, the UIDs look like

`uid://<archiveId>/<global>/<local>`

Currently the `<archiveId>` is composed of a letter followed by a 2- to 4-digit decimal number, with differing letters for different users, and different numbers for each site or machine (e.g., 01=OSF, 02=AOS, 11=Asian ARC; while individual machines have a unique number generated from their MAC addresses). `<global>` and `<local>` are hexadecimal numbers. `<global>` is centrally controlled and assigned by the ALMA archive, while `<local>` is locally controlled and assigned by using the UID libraries. So for instance `uid://A01/X2a/Xd` refers to an object created by user `A` (i.e., `alma`) in the `01` archive (i.e., the OSF); it is the `0x2a`'th (42nd) object thus created.

EVLA note:
For the EVLA currently, the UIDs look like

`uid://evla/<type>/X<time>`

where `<type>` refers to the type of the corresponding object (e.g., `bdf` for a binary data file, `sdm` for an SDM table), and `<time>` is the current time in milliseconds since the local computer's reference time.[14] So for example `uid://evla/bdf/X1238096948463` refers to an EVLA BDF object created at some time which is intrinsically uninteresting, but likely to be unique.

2.5.6. Array shapes and sizes

All information needed to specify the shape of the entries in a given table, should appear in that table. Thus for instance the Source Table may list a number of spectral transitions for a given source, and the number of those transitions as well as the actual list must appear in the row for that source. Note that this is an *additional constraint* beyond the regular required fields.

This approach leads to the duplication of some information between tables. For instance, the number of antennas (`numAnt`) appears in both the ConfigDescription and the Main Tables, because there are arrays with that dimension in both those tables. In such cases the field name will be the same in both tables. Note however that the mere fact that a field name is the same in two tables

does *not* mean that the field is used in the same fashion (and hence should have the same value) in those two tables! One must consult the individual table descriptions to see whether two such fields must in fact agree.

2.5.7. Associations and groups

In some cases one would like to group multiple rows together within a table. For instance, it is interesting to know that row A refers to the channel-averaged data, while row B contains the same data at full spectral resolution. Such groupings in the SDM are done through **associations**.

In associations, the associated row(s) are referred to via a tag or identifier (`assoc<TableName>Id`), with the nature(s) of the association given by the corresponding `assocNature`. Taking the tables in alphabetical order, associations are used as follows:

- Antenna Table: associates positions measured for the same antenna at different times.
- ConfigDescription Table: associates channel-averaged and full spectral resolution data.
- Field Table: associates fields which are astronomically related (e.g., different pointings on the same galaxy).
- FocusModel and PointingModel Tables: associates rows which should be summed to give the total model.
- SpectralWindow Table: associates channel-averaged and full spectral resolution data. `imageSpectralWindow` is similar, pointing to the row corresponding to the image sideband of the current spectral window.

The Scan and SpectralWindow Tables also allow grouping rows together for calibration purposes. In the Scan Table this is done through `calibrationSet` (giving the type of calibration, e.g., amplitude gain) and `calibrationFunction` (identifying the first or last scan in the calibration set). In the SpectralWindow Table, `freqGroup` and `freqGroupName` can be used to group together, e.g., line-free channels for use in amplitude calibration.

2.6. Relationship between the SDM and the Measurement Set

Both ALMA and the EVLA have adopted CASA as their main post-processing package. The SDM is derived from CASA's Measurement Set model (1), taking much of the structure of the MS' auxiliary tables. There are however several significant differences.

1. The binary data in the SDM are located in separate BLOBs referenced from the Main Table, while the binary data in the MS are located in the Main Table itself.

2. The SDM adds a ConfigDescription Table to identify the basic hardware configuration: the sets of antennas, frequency windows, and backends used. These change rarely during an observation and this new table simplifies the Main Table considerably.

3. The SDM contains a great deal of information that is not stored in the MS. Eventually this information will either prove interesting (in which case the MS and filler should be modified to use it) or boring (in which case it should be dropped from the SDM).

4. Several items and tables have been renamed to match ALMA terminology (scans, observations, integrations).

5. The SDM has no keywords (variables attached to a table which characterize the table as a whole and do not vary from row to row).

6. The SDM uses tags and keys to identify rows, with integer IDs in some cases being part of complex keys. The MS uses integer IDs rather differently (see §2.2.2) and also has the concept of a row number, which does not exist in the SDM.

The correspondence between data in the SDM and data in the MS is discussed in §C. This is not part of the definition of the SDM (which is entirely independent of the MS) but is often useful in understanding what is going on.

2.7. Relationship between the SDM and the BDF

Each row in the SDM's Main Table references a unique BDF file through `dataUid`. There is necessarily some association between the contents of that BDF file and the corresponding entries in the SDM; and the definitions of the BDF and the SDM *jointly* determine what can be stored in a single binary data file. This section discusses these inter-dependencies.

2.7.1. Spectral Windows in the BDF

The BDF's `sdmDataHeader` defines the axes, dimensionality, and ordering of the visibility data. Here is an example of how this works:

```
<crossData size="540672" axes="BAL BAB SPW BIN SPP POL"/>
<numAntenna>12</numAntenna>
<baseband name="BB_1">
  <spectralWindow sw="1"
    swbb="BB_1"
    crossPolProducts="RR RL LR LL"
```

```
      numSpectralPoint="256"
      numBin="1"
      scaleFactor="1.000000" sideband="NOSB"/>
    <spectralWindow sw="3"
      swbb="BB_1"
      crossPolProducts="RR"
      numSpectralPoint="1024"
      numBin="1"
      scaleFactor="1.000000" sideband="NOSB"/>
  </baseband>
  <baseband name="BB_2">
    <spectralWindow sw="1"
      swbb="BB_2"
      crossPolProducts="RR LL"
      numSpectralPoint="512"
      numBin="2"
      scaleFactor="1.000000" sideband="NOSB"/>
</baseband>
```

- The values of the `<baseband name>` and `swbb` attributes must be identical. This redundancy is required (1) to match the APDM, and (2) for ease of implementation (validation, code generation, and use of query languages).
- The ordering of the `baseband` and `spectralWindow` elements in the `sdmDataHeader` defines the order in which the actual binary data appear. In this example the order of the binary data will be:
  `BB_1/sw=1 BB_1/sw=3 BB_2/sw=1`
  - The `baseband` and `swbb` attributes must be in *ascending* order: i.e., `swbb="BB_4"` must occur after `swbb="BB_2"`. The actual values of these attributes are not important, although it would be good practice to match those used in the corresponding SDM.
  - The `sw` attributes must be in *ascending* order: i.e., `sw="3"` must occur after `sw="1"`. Note that this attribute does not correspond in any direct way to the `spectralWindowId` tag in the SDM (for one thing, that's a tag, not an integer).
  - The `sw` values are important only when there are sideband associations.

EVLA note:
Since the EVLA uses only single-sideband receivers the values of the `sw` attributes are not important, and the `sideband` must be set to `"NOSB"`.

ALMA note:

The `sw` values are used by ALMA to track sideband associations: the `image` attribute references a local `sw` id.

– The BDF does not require any particular ordering of these elements. However, consistency with the SDM requires that the ordering of these elements in the BDF must match the order of the elements in the `dataDescriptionId` array in the corresponding row of the SDM's ConfigDescription Table. This is discussed further below.

ALMA note:

For double-sideband receivers, spectral windows which are image sidebands of each other must be defined contiguously, with the lower sideband (LSB) first, and the upper sideband (USB) following.

For the ACA correlator, the spectral windows should be arranged in ascending order of the first channel of the spectral window in terms of 3.8 kHz channels.

- For this example the dimensions of the binary data are as follows:
  - `BAL`: `numAntenna*(numAntenna-1)/2= 66`
  - `BAB`: number of `baseband` elements= 2
  - `SPW`: number of `spectralWindow` elements:
    first `BAB` (set by order of `baseband` elements) → 2
    second `BAB` (set by order of `baseband` elements) → 1
  - `BIN`: given explicitly for each spectral window
  - `SPP`: given explicitly for each spectral window
  - `POL`: given implicitly for each spectral window by the number of `crossPolProducts`

2.7.2. Correspondences between spectral windows in the BDF and in the SDM

The SDM's Main Table refers to a BDF file through `dataUid`. Each `dataUid` occurs in a Main Table row which includes a `configDescriptionId`. The `configDescriptionId` is a tag and refers to a unique row in the ConfigDescription Table. That ConfigDescription row references a number of rows in the DataDescription Table through an array, `dataDescriptionId`, which has `numDataDescription` elements.

- `numDataDescription` is the total number of Spectral Windows, summed over all basebands. For the above BDF example this is 3.
- The order of the elements in the `dataDescriptionId` array is vital: the ith spectral window in the BDF's `sdmDataHeader` corresponds to the ith element in `dataDescriptionId`, and the

corresponding row in the DataDescription Table points to the header information for that Spectral Window in the SpectralWindow and Polarization Tables (ignoring holography for now!).
So in the above example, the header information for:
`BB_1/sw=1` comes from `dataDescriptionId[0]`
`BB_1/sw=3` comes from `dataDescriptionId[1]`
`BB_2/sw=1` comes from `dataDescriptionId[2]`

2.7.3. Multiple simultaneous BDF files

Each row of the Main Table refers to a unique binary data file. This implies that the number of integrations (`numIntegration`) and the nominal data interval (scheduled duration; `interval`) must be the same for all data in a single binary data file. Modern correlators allow considerably more flexibility – one might for instance employ recirculation to increase the spectral resolution for one spectral window, at the expense of poorer time sampling. In this case one would write multiple binary data files at the same time; each would be referenced by a different row in the Main Table; and each such row could refer to a different row in the ConfigDescription Table, and hence (through the `dataDescriptionId` array) to a different set of spectral windows.

3. SDM Tables

TO BE DONE!

- Add ASDM Table
- separate CalDM tables
- Name
- Status: last modified & by whom
- ‘‘references" & ‘‘referenced by"
- Meaning of special values (e.g., blank or non-existent)
- Notes a la MS

Table 4. Data Types and units in the SDM.

Data type	Unit
`Angle`	radians
`ArrayTime`	MJD in nanoseconds
`ArrayTimeInterval`	ArrayTime + Interval
`Flux`	janskys (should be flux density, sigh)
`Frequency`	Hertz
`Humidity`	per cent
`Interval`	nanoseconds
`Length`	meters
`Pressure`	pascals
`Speed`	meters per second
`Temperature`	kelvins

Note. — The SDM uses SI units throughout, apart from using nanoseconds (instead of seconds) for times.

Table 5. Field names and the tables in which they appear in the SDM.

Field	Tables
aborted	ExecBlock
accumMode	CorrelatorMode
almaRadiometerId	AlmaRadiometer
alphaSpectrum	CalAtmosphere
ambientTemperature	CalFocus, CalHolography, CalPointing
ampli	CalPhase
annotationId	Annotation
antennaId	Annotation, Antenna, CalDevice, ConfigDescription, DelayModel, ExecBlock, Feed, Focus, FocusModel, FreqOffset, GainTracking, Pointing, PointingModel, SysCal, WVMCal
antennaMake	Antenna, CalFocusModel, CalHolography, CalPointing, CalPointingModel, CalPrimaryBeam
antennaName	CalAmpli, CalAtmosphere, CalDelay, CalFocus, CalFocusModel, CalHolography, CalPointing, CalPointingModel, CalPosition, CalPrimaryBeam, CalWVR
antennaNames	CalBandpass, CalCurve, CalPhase
antennaPosition	CalPosition
antennaType	Antenna
apertureEfficiency	CalAmpli
apertureEfficiencyError	CalAmpli
appliedCalibrations	CalReduction
assocAntennaId	Antenna
assocCalDataId	CalData
assocCalNature	CalData
assocConfigDescriptionId	ConfigDescription
assocFieldId	Field
assocFocusModelId	FocusModel
assocNature	ConfigDescription, Field, FocusModel, PointingModel, SpectralWindow
assocPointingModelId	PointingModel
assocSpectralWindowId	SpectralWindow
atmDryDelay	DelayModel
atmPhaseCorrection	CalAmpli, CalBandpass, CalCurve, CalDelay, CalFocus, CalPhase, CalPointing, CalPosition, CalSeeing, ConfigDescription
atmWetDelay	DelayModel
attenuator	GainTracking

Table 5—Continued

Field	Tables
attFreq	GainTracking
attSpectrum	GainTracking
averagedPolarizations	CalPointing
axesOffset	CalPosition
axesOffsetErr	CalPosition
axesOffsetFixed	CalPosition
axesOrderArray	CorrelatorMode
azimuthRMS	CalPointingModel
bandType	SquareLawDetector
basebandConfig	CorrelatorMode
basebandName	Annotation, CalBandpass, CalDelay, CalPhase, SpectralWindow
basebandNames	CorrelatorMode
baselineLengths	CalPhase, CalSeeing
basePa	ExecBlock
baseRangeMax	ExecBlock
baseRangeMin	ExecBlock
baseRmsMajor	ExecBlock
baseRmsMinor	ExecBlock
beamId	Beam, Feed
beamMapUID	CalHolography, CalPrimaryBeam
beamOffset	Feed
beamPA	CalPointing
beamPAError	CalPointing
beamPAWasFixed	CalPointing
beamWidth	CalPointing
beamWidthError	CalPointing
beamWidthWasFixed	CalPointing
binMode	CorrelatorMode
cableDelay	GainTracking
calDataId	CalAmpli, CalAtmosphere, CalBandpass, CalCurve, CalData, CalDelay, CalFlux, CalFocus, CalFocusModel, CalGain, CalHolography, CalPhase, CalPointing, CalPointingModel, CalPosition, CalPrimaryBeam, CalSeeing, CalWVR
calDataType	CalData, Scan

Table 5—Continued

Field	Tables
calDeviceName	State
calEff	CalDevice
calibrationFunction	Scan
calibrationGroup	Source
calibrationOnLine	Scan
calibrationSet	Scan
calLoadNames	CalDevice
calPattern	Scan
calReductionId	CalAmpli, CalAtmosphere, CalBandpass, CalCurve, CalDelay, CalFlux, CalFocus, CalFocusModel, CalGain, CalHolography, CalPhase, CalPointing, CalPointingModel, CalPosition, CalPrimaryBeam, CalReduction, CalSeeing, CalWVR
calType	CalData
catalog	Source
centerDirection	SBSummary
centerDirectionCode	SBSummary
centerDirectionEquinox	SBSummary
chanFreq	CalWVR
chanFreqArray	SpectralWindow
chanFreqStart	SpectralWindow
chanFreqStep	SpectralWindow
chanWidth	CalWVR, SpectralWindow
chanWidthArray	SpectralWindow
clockDelay	DelayModel
code	Field, Source
coeffError	CalFocusModel, CalPointingModel
coeffFixed	CalFocusModel, CalPointingModel
coeffFormula	CalFocusModel, CalPointingModel, FocusModel, PointingModel
coeffName	CalFocusModel, CalPointingModel, FocusModel, PointingModel
coeffVal	CalPointingModel, FocusModel, PointingModel
coeffValue	CalFocusModel
collError	CalPointing
collOffsetAbsolute	CalPointing
collOffsetRelative	CalPointing
collOffsetTied	CalPointing
configDescriptionId	ConfigDescription, Main

Table 5—Continued

Field	Tables
configName	ExecBlock
correctionValidity	CalAmpli, CalPhase
correlationBit	SpectralWindow
correlationMode	ConfigDescription
correlatorCalibration	Subscan
correlatorModeId	CorrelatorMode
correlatorName	CorrelatorMode
corrProduct	Polarization
corrType	Polarization
crossDelayOffset	CalDelay
crossDelayOffsetError	CalDelay
crossPolarizationDelay	GainTracking
curve	CalBandpass, CalCurve
dataDescriptionId	ConfigDescription, DataDescription
dataOid	Main
dataSize	Main
decorrelationFactor	CalPhase
delayDir	Field
delayError	CalDelay
delayOffset	CalDelay, GainTracking
delayRms	CalPosition
deltaVel	Source
details	Annotation
dewPoint	Weather
dewPointFlag	Weather
direction	CalFlux, CalHolography, CalPhase, CalPointing, Source
directionCode	CalFlux, Field, Source, SwitchCycle
directionEquinox	CalFlux, Field, Source, SwitchCycle
dirOffsetArray	SwitchCycle
dishDiameter	Antenna
dispDelay	DelayModel
distance	Holography

Table 5—Continued

Field	Tables
dopplerId	Doppler, SpectralWindow
dValue	Annotation
effectiveBw	SpectralWindow
effectiveBwArray	SpectralWindow
elevationRms	CalPointingModel
encoder	Pointing
endTime	ExecBlock, Scan, Subscan
endTimeObserved	CalData
endValidTime	CalAmpli, CalAtmosphere, CalBandpass, CalCurve, CalDelay, CalFlux, CalFocus, CalFocusModel, CalGain, CalHolography, CalPhase, CalPointing, CalPointingModel, CalPosition, CalPrimaryBeam, CalSeeing, CalWVR
ephemerisId	Ephemeris, Field
execBlockId	ExecBlock, Main, Scan, Subscan
execBlockNum	ExecBlock
execBlockUID	CalData, ExecBlock
exponent	CalSeeing
feedId	CalDevice, ConfigDescription, Feed, FreqOffset, GainTracking, SysCal
feedNum	Feed
fieldId	Field, Main
fieldName	CalData, Field, Scan, Subscan
filterMode	CorrelatorMode
fit	CalGain
fitWeight	CalGain
flagRow	ExecBlock, Main, Polarization, Scan, Subscan
flux	CalFlux, Source
fluxErr	Source
fluxError	CalFlux
fluxMethod	CalFlux
focus	Holography
focusCurveWasFixed	CalFocus
focusCurveWidth	CalFocus
focusCurveWidthError	CalFocus
focusMethod	CalFocus

Table 5—Continued

Field	Tables
focusModel	CalFocusModel
focusModelId	Focus, FocusModel
focusOffset	Focus
focusPosition	CalHolography
focusReference	Feed
focusRMS	CalFocusModel
focusTracking	Focus
forwardEfficiency	CalAtmosphere
forwardEfficiencyError	CalAtmosphere
forwardEffSpectrum	CalAtmosphere
freqGroup	SpectralWindow
freqGroupName	SpectralWindow
freqLimits	CalBandpass
freqLO	Receiver
freqOffset	GainTracking
freqOffsetArray	SwitchCycle
frequency	SBSummary, Source
frequencyBand	Receiver, SBSummary
frequencyInterval	Source
frequencyRange	CalAmpli, CalAtmosphere, CalCurve, CalFocus, CalHolography, CalPhase, CalPointing, CalPrimaryBeam, CalSeeing
frequencyRanges	CalFlux
frequencyRefCode	Source
frequencySpectrum	CalAtmosphere
gain	CalGain
gainValid	CalGain
geomDelay	DelayModel
gravCorrection	CalHolography
gravOptRange	CalHolography
groundPressure	CalAtmosphere
groundRelHumidity	CalAtmosphere
groundTemperature	CalAtmosphere
groupDelay	DelayModel

Table 5—Continued

Field	Tables
holographyId	Holography
illuminationTaper	CalHolography
illumOffset	Feed
imageSpectralWindowId	SpectralWindow
inputAntennaNames	CalWVR
integrationTime	CalPhase, CalSeeing
interval	Annotation, Main
invalidConditions	CalReduction
issue	Annotation
lineArray	SpectralWindow
llValue	Annotation
loPropagationDelay	GainTracking
mainBeamEfficiency	CalPrimaryBeam
measFreqRef	SpectralWindow
measuredFocusPosition	Focus
messages	CalReduction
modeId	Processor
name	Antenna, Receiver, SpectralWindow, Station
netSideband	SpectralWindow
noiseCal	CalDevice
numAntenna	AlmaRadiometer, Annotation, CalBandpass, CalCurve, CalPosition, ConfigDescription, ExecBlock, Main
numApplied	CalReduction
numAssocValues	ConfigDescription, SpectralWindow
numAtmPhaseCorrection	ConfigDescription
numAttFreq	GainTracking
numAxes	CorrelatorMode
numBand	SquareLawDetector
numBaseband	Annotation, CorrelatorMode

Table 5—Continued

Field	Tables
numBaseLengths	CalSeeing
numBaseline	CalBandpass, CalCurve, CalPhase
numberIntegration	Subscan
numberRepeats	SBSummary
numberSubintegration	Subscan
numCalload	CalDevice
numChan	CalWVR, SpectralWindow, SysCal, WVMCal
numCoeff	CalFocusModel, CalPointingModel, FocusModel, PointingModel
numCorr	Holography, Polarization
numDataDescription	ConfigDescription
numFeed	ConfigDescription
numField	Scan
numFreq	CalAtmosphere, Source
numFrequencyRanges	CalFlux
numInputAntennas	CalWVR
numIntegration	Main
numIntent	Scan
numInvalidConditions	CalReduction
numLines	Source
numLO	GainTracking, Receiver
numLoad	CalAtmosphere
numObs	CalPointingModel
numObservingMode	SBSummary
numPanelModes	CalHolography
numParam	CalReduction
numPoly	CalBandpass, CalCurve, CalWVR, DelayModel, Field, WVMCal
numReceptor	CalAmpli, CalAtmosphere, CalBandpass, CalCurve, CalDelay, CalDevice, CalFocus, CalHolography, CalPhase, CalPointing, CalPrimaryBeam, Feed, GainTracking, SysCal
numSample	Pointing
numScan	CalData
numScienceGoal	SBSummary
numScrew	CalHolography
numSideband	CalDelay
numSourceObs	CalFocusModel

Table 5—Continued

Field	Tables
numStep	SwitchCycle
numStokes	CalFlux, Source
numSubScan	Scan
numTerm	Pointing
numWeatherConstraint	SBSummary
observerName	ExecBlock
observingLog	ExecBlock
observingMode	SBSummary
obsUnitSetId	SBSummary
offIntensity	CalFocus, CalPointing
offIntensityError	CalFocus, CalPointing
offIntensityWasFixed	CalFocus, CalPointing
offset	Antenna, CalFocus, FreqOffset, Pointing
offsetError	CalFocus
offsetWasTied	CalFocus
onSky	State
outerScale	CalSeeing
outerScaleRMS	CalSeeing
oversampling	SpectralWindow
overTheTop	Pointing
PA	CalFlux
PAError	CalFlux
paramSet	CalReduction
pathCoeff	CalWVR, WVMCal
peakIntensity	CalFocus, CalPointing
peakIntensityError	CalFocus, CalPointing
peakIntensityWasFixed	CalFocus, CalPointing
phase	CalPhase
phasedArrayList	ConfigDescription
phaseDelay	DelayModel

Table 5—Continued

Field	Tables
phaseDiffFlag	SysCal
phaseDiffSpectrum	SysCal
phaseDir	Field
phaseOffset	GainTracking
phaseRms	CalPosition
phaseRMS	CalPhase, CalSeeing
pointingDirection	CalFocus, Pointing
pointingMethod	CalPointing
pointingModelId	Pointing, PointingModel
pointingModelMode	CalPointing, CalPointingModel
pointingTracking	Pointing
polarizationId	Polarization
polarizationsAveraged	CalFocus
polarizationType	CalFocusModel, CalPointingModel, FocusModel, PointingModel
polarizationTypes	CalAmpli, CalAtmosphere, CalBandpass, CalCurve, CalDelay, CalFocus, CalHolography, CalPhase, CalPointing, CalPrimaryBeam, Feed, GainTracking
polOrHoloId	DataDescription
polResponse	Feed
polyFreqLimits	CalWVR, WVMCal
position	Antenna, Feed, Source, Station
positionAngle	Source
positionAngleErr	Source
positionErr	CalPosition
positionMethod	CalPosition
positionOffset	CalPosition
powerLoadSpectrum	CalAtmosphere
powerSkySpectrum	CalAtmosphere
pressure	Weather
pressureFlag	Weather
processorId	ConfigDescription, Processor
processorSubType	Processor
processorType	ConfigDescription, Processor
projectId	ExecBlock
projectUID	SBSummary

Table 5—Continued

Field	Tables
properMotion	Source
quantization	SpectralWindow
rangeVel	Source
rawRMS	CalHolography
receiverBand	CalAmpli, CalAtmosphere, CalBandpass, CalCurve, CalDelay, CalFocus, CalFocusModel, CalHolography, CalPhase, CalPointing, CalPointingModel, CalPosition, CalPrimaryBeam, FocusModel, PointingModel
receiverDelay	GainTracking
receiverId	Feed, Receiver
receiverSideband	Receiver
receptorAngle	Feed
reducedChiSquared	CalBandpass, CalCurve, CalDelay, CalFocus, CalFocusModel, CalPointing, CalPointingModel, CalPosition
ref	State
refAntennaName	CalBandpass, CalCurve, CalDelay
refAntennaNames	CalPosition
refChan	SpectralWindow
referenceDir	Field
refFreq	CalDelay, SpectralWindow
refFreqPhase	CalDelay
refTemp	CalWVR, WVMCal
relativeAmplitudeRms	CalPrimaryBeam
releaseDate	ExecBlock
relHumidity	Weather
relHumidityFlag	Weather
resolution	SpectralWindow
resolutionArray	SpectralWindow
restFrequency	Source
rms	CalBandpass, CalCurve

Table 5—Continued

Field	Tables
samplingLevel	GainTracking
sbDuration	SBSummary
sbGain	CalAtmosphere
sbGainError	CalAtmosphere
sbGainSpectrum	CalAtmosphere
sbSummary	ExecBlock
sBSummaryId	ExecBlock, SBSummary
sbSummaryUID	SBSummary
sbType	SBSummary
scanIntent	CalData, Scan
scanNumber	Main, Scan, Subscan
scanSet	CalData
schedulerMode	ExecBlock
scienceGoal	SBSummary
screwMotion	CalHolography
screwMotionError	CalHolography
screwName	CalHolography
seeing	CalSeeing
seeingError	CalSeeing
sessionReference	ExecBlock
sideband	CalBandpass
sidebandLO	Receiver
sidebandProcessingMode	SpectralWindow
sidebands	CalDelay
sig	State
siteAltitude	ExecBlock
siteLatitude	ExecBlock
siteLongitude	ExecBlock
size	CalFlux, Source
sizeErr	Source
sizeError	CalFlux
skyRMS	CalPointingModel
software	CalReduction

Table 5—Continued

Field	Tables
softwareVersion	CalReduction
sourceCode	CalData
sourceId	Doppler, Field, Source
sourceModel	CalFlux, Source
sourceName	CalData, CalFlux, Scan, Source
sourceOffset	Pointing
sourceOffsetEquinox	Pointing
sourceOffsetReferenceCode	Pointing
spectralType	ConfigDescription
spectralWindowId	AlmaRadiometer, CalDevice, DataDescription, Feed, FreqOffset, GainTracking, Receiver, Source, SpectralWindow, SysCal, WVMCal
squareLawDetectorId	SquareLawDetector
startTime	ExecBlock, Scan, Subscan
startTimeObserved	CalData
startValidTime	CalAmpli, CalAtmosphere, CalBandpass, CalCurve, CalDelay, CalFlux, CalFocus, CalFocusModel, CalGain, CalHolography, CalPhase, CalPointing, CalPointingModel, CalPosition, CalPrimaryBeam, CalSeeing, CalWVR
stateId	Main, State
stationId	Antenna, Station, Weather
stationName	CalPosition
stationPosition	CalPosition
statPhaseRMS	CalPhase
stepDurationArray	SwitchCycle
stokes	CalFlux
stokesParameter	Source
subscanIntent	Subscan
subscanMode	Subscan
subscanNumber	Main, Subscan
surfaceMapUID	CalHolography
switchCycleId	ConfigDescription, SwitchCycle
syscalType	CalAtmosphere
sysVel	Source

Table 5—Continued

Field	Tables
tantFlag	SysCal
tantSpectrum	SysCal
tantTsysFlag	SysCal
tantTsysSpectrum	SysCal
target	Pointing
tAtm	CalAtmosphere
tAtmSpectrum	CalAtmosphere
tau	CalAtmosphere
tauSpectrum	CalAtmosphere
tcalFlag	SysCal
tcalSpectrum	SysCal
telescopeName	ExecBlock
tempCorrection	CalHolography
temperature	Weather
temperatureFlag	Weather
temperatureLoad	CalDevice
tempOptRange	CalHolography
time	Annotation, Antenna, Field, Main
timeInterval	CalDevice, DelayModel, Feed, Focus, FreqOffset, GainTracking, Pointing, Receiver, Source, SysCal, Weather, WVMCal
timeOrigin	DelayModel, Pointing
timeReduced	CalReduction
timeSampling	Main
totalFit	CalGain
totalFitWeight	CalGain
totalGainValid	CalGain
totBandwidth	SpectralWindow
transition	Source
transitionIndex	Doppler
tRec	CalAtmosphere
tRecSpectrum	CalAtmosphere
trxFlag	SysCal
trxSpectrum	SysCal
tskyFlag	SysCal

Table 5—Continued

Field	Tables
tskySpectrum	SysCal
tSys	CalAtmosphere
tsysFlag	SysCal
tsysSpectrum	SysCal
tSysSpectrum	CalAtmosphere
type	Holography, Station
typeCurve	CalBandpass, CalCurve
usePolynomials	Pointing
vdValue	Annotation
velDef	Doppler
vllValue	Annotation
vvdValues	Annotation
vvllValue	Annotation
water	CalAtmosphere
waterError	CalAtmosphere
weatherConstraint	SBSummary
weight	State
weightArray	SwitchCycle
weightedRMS	CalHolography
wereFixed	CalFocus
windDirection	Weather
windDirectionFlag	Weather
windMax	Weather
windMaxFlag	Weather
windowFunction	SpectralWindow
windSpeed	Weather
windSpeedFlag	Weather
wvrMethod	CalWVR, WVMCal

4. Enumerations

TO BE DONE!

5. Implementation

5.1. XML

QUERY:
Do we need anything here apart from pointers to the schemata? E.g., required headers or whatever.

5.2. The SDM on disk

These requirements stem primarily from the SDM-to-MS filler, and may be relaxed over time.

1. All SDM tables except the BLOBs (binary data) must be stored in the same directory on disk. The BLOBs (binary data) must be stored in a sub-directory of this main directory, named `SDMBinary`.

2. Each SDM table (except the BLOBs) must be stored in a separate file, named `<TableName>.xml`. So for instance the Feed Table should be stored in the file `Feed.xml` on disk.

3. The binary data must be stored in separate files in the `SDMBinary` directory. Each file corresponds to a `dataUid` referenced by a row in the Main Table. The name of the binary file should correspond to the `dataUid`, with "_" (underscore) substituted for all ":" (colons) and "/" (foward slashes) in the `dataUid`. So for instance if the `dataUid` is
   `"uid://evla/bdf/1238096948503"`
   then the file name on disk should be
   `uid__evla_bdf_1238096948503`

6. Use cases

TBD later.

6.1. Storage of Tsys and Tcal

6.2. Standard interferometry

6.3. Mosaicking

6.4. Pulsars

Binning, gating, timing

6.5. Astrometry

- accounting (CALC)

6.6. VLBI

- phased array
- single dish

6.7. Wideband autocorrelations

6.8. Burst mode

6.9. Holography

6.10. Doppler tracking

Can one Doppler track on one source while observing another?

How does one say one only changes frequencies at scan boundaries, vs. continuously?

7. Evolution of the SDM

The SDM is primarily intended for use by two telescopes which do not yet exist: ALMA and the EVLA. Major design changes to the SDM will have correspondingly major impacts on the schedules of those two construction projects. Those impacts must be weighed heavily before introducing changes motivated more by aesthetics than by necessity.

That said, the SDM will undoubtedly evolve as it is used. The design should be flexible enough to respond to our growing understanding of the instruments and their use.

7.1. Versioning

7.2. Revisions

- expected approach
- timescale

7.3. Change history

A. XML schemata

XML schemata for the Science Data Model may be found in the same repository and project under which the master copy of the present document exists.

A.1. Repository access

The ALMA CVS repository contains the SDM specification documents and schemata. Instructions for obtaining a CVS account for the repository can be found at:
`http://www.eso.org/projects/alma/develop/alma-se/reference/CVS.html#login`
To check out a copy of the documents and schemata from CVS, follow these instructions:

1. Set the CVSROOT environment variable to
   `:pserver:_YOUR_ACCOUNT_ID_@cvssrv.hq.eso.rog:/project2/CVS`
   where `_YOUR_ACCOUNT_ID` is you CVS login account ID.
2. Log in to the CVS server with the command
   `cvs login`
3. Check out a copy of the project with the command
   `cvs co -r sdm_1_0 ICD/HLA/SDM`

A.2. Schema validation

A simple schema validation tool is available in `ICD/HLA/SDM/test`. This tool validates the XML parts of an SDM document against the XML schemata. To run the tool, follow these steps:

1. `cd ICD/HLA/SDM/src`
2. `make all install`
   ... this builds and installs the validation tool

3. `validateSDM <sdm_file>`
   ... this executes the validation tool on file `<sdm_file>`

B. Glossary of terms

array

baseband

BDF Binary Data Format

BLOB The contents of a single document in the Binary Data Format which includes MIME, XML and binary parts.

CalDM Calibration Data Model

CASA The Common Astronomy Software Applications (CASA) package. The primary data processing and analysis package for both ALMA and the EVLA.

cell

channel

channel average A group of spectral channels averaged together; these data (if present) are recorded every sub-integration.

EVLA note:

Channel average data are not currently written by the EVLA.

column

composite key

enumeration

Execution Block

field

integration The basic unit of data produced by a data processor in the Binary Data Format at the highest available spectral resolution. Generally recorded at a lower time resolution and higher spectral resolution than a sub-integration.

ITRF International Terrestrial Reference Frame

key

LO Local oscillator

MIME Multipurpose Internet Mail Extensions

MS Measurement Set; format is defined in Kemball & Wieringa (1).

ObsUnitSet

OID Object identifier

Project

Program

Program Block

row

SB Scheduling Block

scan A sequence of one or more consecutive observations that share a common goal.

Scheduling Block

SDM Science Data Model

simple key

spectral window

subarray The complete set of antennas producing the data of a BLOB.

sub-integration A data set of channel-averaged data from a correlator. Generally exists at a higher time resolution and lower spectral resolution than an integration.

subband

subscan A sequence of data (integrations or sub-integrations) comprising a single BLOB.

table

tag

UID Unique identifier

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

XML

C. Measurement Set equivalences

```
- two-way Excel spreadsheet
- action of filler
```

D. SDM data origins & fill rates

```
--> graph showing origins, fill rates, etc. for both ALMA & EVLA
```

E. ALMA notes, examples, and sizes

F. EVLA notes, examples, and sizes

G. Planned enhancements and outstanding questions

1. Order of fields in XML schemata should be changed to match the individual table documentation.
2. Should keep track of how spectral windows are formed (constituent subbands and stitching method).
3. Lag spectra

1. Station Table: should this have `time` as a required field?

Missing information:

1. The version number of SDM/BDF
   This seems to be embedded in the XML.
2. The units of the raw (binary) data (corr. coefficient, K in T_A^*, Jy, ...)
   MPR suggestion: add this to each processor table (ALMARadiometer, SquareLawDetector, CorrelatorMode). Should be required.
3. Storage of lags (rather than spectra)
   MPR suggestion: add enumerated list of possibles (if there's anything other than LAG and SPECTRA) to SpectralWindow Table. If these are the only options, make it a boolean (lagSet). Should be optional (not given means spectra).

4. Whether Doppler tracking is done "continuously" or just at the beginning of each scan
   MPR: Add this to Doppler Table. Should be required I suppose (if we think a given telescope might employ both methods).

5. BDF2.0.1, p. 21: "The time basis (UTC, IAT, local, etc.) is defined in the SDM, and the times recorded in this component are expressed in that basis."
   . . . Currently this is only recorded implicitly (see §2.5.2).

6. Antenna gains
   Also seems not to be stored in the MS: sensitivity (K/Jy) and changes with AzEl

7. Array geometry
   Compare FITS-IDI: pole; rotation rate

8. Sampler information

9. Spacecraft orbit

10. Pulsar gating and binning

11. CALC parameters

G.1. Consistency

Naming conventions:

1. Tags: Could we perhaps change the naming convention for tags (since the MS has IDs which are integers; see below)? I suggest we choose a convention from among the following:
   - `<tableName>Ptr`
   - `<tableName>Tag` (with the type being `<TableName>TagType`$*$)
   - `<tableName>Row`
   - `<tableName>UniqId`

2. UIDs should be `<name>UID`.

3. Times:
   - Field:`time`: change to `timeOrigin`
     . . . more descriptive
   - CalCurve:`startValidTime, endValidTime`: change to `startTime, endTime`
     . . . time here serves both to show the validity interval, and to specify the temporal boundaries of the Chebyshev polynomial; the current name emphasizes the former at the expense of the latter

- Field:`time`: change to `timeOrigin`
- Cal:`startValidTime, endValidTime`: change to `startTimeValid, endTimeValid`
  ... to match `startTimeObserved, endTimeObserved`, and to correctly imply that "Valid" modified "time" rather than the reverse
- Main:`time, interval`: change to `timePlanned, intervalPlanned`
  ExecBlock:`startTime, endTime`: change to `startTimePlanned, endTimePlanned`
  ... to clarify that this is planned rather than observed time
  ... to match `timeValid` and `timeObserved`
- Scan, Subscan:`startTime, endTime`: change to `startTimeObserved, endTimeObserved`
  ... to clarify that this is observed rather than planned time
  ... to match `timeValid` and `timeObserved` elsewhere

4. Frequencies:
   - in the Receiver Table we have `freqLO` and `frequencyBand`. I would like to see either `freq` or `frequency` used, but not both.
5. `sideBand` vs. `sideband`
6. `temp`, `t`, and `temperature`

- Order of keys and fields
- Naming of ALMA- and EVLA-specific tables and fields
- What should go in the SDM
- Binary vs. XML tables

G.2. CALC

G.3. Samplers etc.

G.4. Ephemeris

Acknowledgements

Joe McMullin
Allen Farris
Ken Sowinski
Lindsey Davis

v. 1.1

Jeff Kern
Michel Caillat
Rich Moeser
CASA folks for MS

REFERENCES

Berners-Lee, T., Rielding, R., & Masinter, L. 2006, *Uniform Resource Identifier (URI): Generic Syntax* [RFC 3986]

Freed, N. & Boorenstein, N. 1996a, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, IETF RFC 2045 [RFC 2045]

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

Kemball, A.J. & Wieringa, M.H. 2000, *MeasurementSet definition version 2.0 (AIPS++ Note 229)*

Levinson, E. 1998, *The MIME Multipart/Related Content-type*, IETF RFC 2387 [RFC 2387]

Lucas, R. & Viallefond, F. 2007, *ALMA Calibration Data Model*

Mangum, J. 2001, *A Telescope Pointing Algorithm for ALMA* (ALMA Memo #366)

Meuss, H. 2008, *Archive ID Assignment*, http://almasw.hq.eso.org/almasw/bin/view/Archive/ArchiveIdAssignment

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

Pokorny, M. & Pisano, J. 2009, *Science Data Model Binary Data Format* version 2.0.1 (14 April 2009)

Viallefond, F. 2006, *The ALMA Science Data Model*, ADASS XV (ASP Conference Series Vol. 351), 627

Viallefond, F. & Lucas, R. 2004, *ALMA Export Data Format*

Wicenec, A. 2005, *The Archive UID Concept*, http://almasw.hq.eso.org/almasw/bin/view/Archive/ArchiveUids

This preprint was prepared with the AAS LaTeX macros v5.2.