FSTS and ADV File Formats

Version 2.1 Specification

09-Feb-2020

The ADV Standard Working Group

Scope

The Flexible Stream Transport Format (FSTF) defines a file format for storing streams of data such as measurements from scientific equipment. The format is built as a generic and extensible binary file format which is fitted for storing of large amounts of data such as 16 bit video frames.

The Astronomical Digital Video (ADV) file format is built on the top of the FSTF and is designed to particularly store astronomical time-stamped video observations.

Purpose

This document formally defines the FSTS and ADV formats for data structuring and exchange. It specifies the organisation and content of FSTS and ADV data sets. A list of standard metadata tags RECOMMENDED for use in the ADV format is given.

Definitions and notations

Conformance

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Numeric values

Decimal and hexadecimal are used within this specification. Decimal numbers are most frequently used to represent quantities or counts. Addresses and sample binary data is represented by hexadecimal numbers.

Byte ordering

The ordering convention of bytes within the 16, 32 and 64 bit integer numbers in the FSTS binary data is little-endian. Images received by cameras CAN have and MAY be saved using both little-endian and big-endian byte order. The exact format is defined in the corresponding data sections.

Format development

The FSTS and ADV open file formats are developed by the ADV Standard Working Group. The following people have made contributions to the project : Hristo Pavlov (Tangra Observatory,

Tangra, OccultWatcher, OccuRec, ADVS, Tony Barry (IOTA-VTI, ADVS) , James Fidell (oaCapture), Chris Garry (SER Player, PIPP), Dave Gault (Kuriwa Observatory, ADVS) , Robin Glover (SharpCap), Bruce Holenstein, Frederic Jabet (AiryLab), Emil Kraaikamp (AutoStakkert)), Stefan Meister (DVTI), Thierry Midavaine, Andreas Schweizer (DVTI)

A set of libraries to read and write in ADV format is maintained at https://github.com/AstroDigitalVideo and is available in **C++** and other programming languages.

Revision History

Version	Date	Notes
2.0	26-Sep-2016	Initial Version
2.1	09-Feb-2020	Update after a review by Andreas Schweizer. Added specification for recording a list of multiple non-overlapping ROIs in a single image layout. Added numbering to all tables. Removed the second entry (repeated stream id at byte position 4) from Table 15 to match the current implementation in AdvLib.
	04-May-2020	Added **BGGR** to the list of supported Bayer patterns in Table 12 after a review by Robin Glover.

FSTF File Format

Overview

The scientific data is saved in *Data Blocks* which are part of logical *Data Streams*. *Data Blocks* from the same *Data Stream* are not necessarily placed sequentially in the file. *Metadata* are name/value pairs used to describe specific properties of a section of the file or the saved data in the *Data Blocks*.

File Structure

The FSTF file is organized into the following 6 blocks:

The Header identifies the file format and defines the data streams and the offsets of the blocks inside the file. The Data Streams Definition block defines the number, type and settings of the data streams and data sections that are saved for each data frame. The Data Blocks contain the actual recorded data frames in all data streams.

The Index Table contains the offsets of each data frame in each data stream in the file for a faster access and the Metadata table contain generic name/value pairs of metadata related to the current recording. There are two Metadata tables – System Metadata Table which was created and populated at the time of the recording and a User Metadata Table, which is created blank and can used by data reduction software to add extra information to the file as necessary.

Data Types

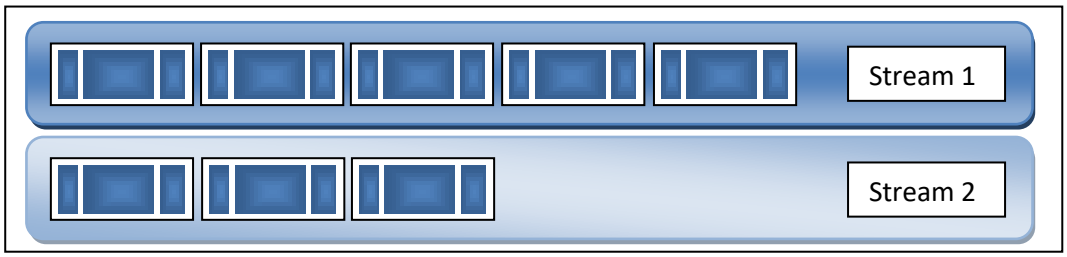
Throughout this document a number of data type abbreviations are used to indicate the type of data of the fields. The following data types are used:

Type	Definition
UInt8	An 8 bit unsigned integer number.
UInt16	A 16 bit unsigned integer number.
UInt32	A 32 bit unsigned integer number.
UInt64	A 64 bit unsigned integer number.
Int64	A 64 bit signed integer number.
UTF8String	A length-prefixed UTF8 string. The first two bytes represent the length of the string in bytes as a 16 bit number. It is followed by the UTF8 bytes of the string literal. The string is not zero byte terminated.
Bytes16	A block of 16 bytes.
Bytes	A block of bytes with variable length.

Table 1. Data Types

Data Streams

The information recorded in the file is saved in *data streams. Data streams* have the same *data structure* but different *data streams* can be used for storing different types of *data frames* or *data frames* acquired at separate time periods. *Data streams* present a way for adding an additional structure to the recording. They are defined by their names and optionally associated metadata.



Data Sections

The information within the *data streams* is stored as sequential *data frames.* Each *data frame* contains one or more sections of data linked together - representing information recorded at the same time. The meaning of the data stored in each section is defined separately in the data stream definition header and is the contract between the recorder and the Reader Software. All *data streams* in the file use the same set of *data sections* for all recorded *data frames*.

Metadata Values

The metadata that can be defined for the *data streams* is saved as key/value pairs of UTF8Strings. When the values have a meaning of a floating point number the decimal separator MUST be a dot (.) and no grouping separators should be used. For example *1029341.23* and 1.02934123E06 are correct floating point values presented as a string while *1,029,341.23* and *1029341,23* are incorrect representations.

Glossary

Name	Definition
Data Frame	An individual piece of data saved in the file. It always consists of one or more sub-items saved together. Each sub-item is the payload for the corresponding data section defined in the file.
Data Section	A of chunk of data (section) that may be saved in a Data Frame. The actual definition and the interpretation of the section are dependent on the software that created the file. The purpose of the section is defined in its header using metadata and the section name.
Data Stream	A group of data frames is saved in a chronological order in a data stream. One file may contain one or more data streams.
Reader Software	Any software that reads the file format for example to do data reduction.

Table 2. Glossary

The format of all file blocks is specified next.

Header

The first section of the FSTF file is the Header. It is positioned at the beginning of the file and its binary content is as follows:

Offset	Type	Description
0x00	UInt32	Magic value - this is the number 0x46545346 (the characters ‘FSTF’) which

		identifies the file format.
0x04	UInt8	The revision of the file format. This document describes revision 02.
0x05	UInt32	MUST be zero (0x00000000).
0x09	UInt64	The offset of the index table from the beginning of the file.
0x11	UInt64	The offset of the system metadata table from the beginning of the file.
0x19	UInt64	The offset of the user metadata table from the beginning of the file.

Table 3. FSTF Header

The offsets to the Index Table and Metadata Tables are 64-bit and a Reader Software MUST be able to handle 64-bit addressing.

Data Streams Definition

The Data Streams Definition block is placed immediately after the Header. Its binary format is presented in the table below. The offsets in the first columns are from the beginning of the file as the Data Streams Definition block can be viewed as a continuation of the Header.

Offset	Type	Description
0x21	UInt8	Number of data streams saved in the file.
0x22	UTF8String	Name of the first data stream.
	UInt32	Number of data frames saved in the first data stream.
	UInt64	The frequency in Hz of the clock used to time the start and end moment of the recorded data frames in the first data stream.
	UInt32	Accuracy of the start and end timestamps in clock ticks for the timings of data frames recorded in the first data stream.
	UInt64	Offset from the beginning of the file of an optional metadata header for the first data stream. If there is no metadata associated with this data stream the value MUST be Zero.
	UTF8String	Name of the second data stream.
	UInt32	Number of data frames saved in the second data stream.
	UInt64	The frequency in Hz of the clock used to time the start and end moment of the recorded data frames in the second data stream.
	UInt32	Accuracy of the start and end timestamps in clock ticks for the timings of data frames recorded in the second data stream.
	UInt64	Offset from the beginning of the file of an optional metadata header for the second data stream. If there is no metadata associated with this data stream the value MUST be Zero.
…	…	…
	UTF8String	Name of the N-th data stream.
	UInt32	Number of data frames saved in the N-th data stream.
	UInt64	The frequency in Hz of the clock used to time the start and end moment of the recorded data frames in the N-th data stream.
	UInt32	Accuracy of the start and end timestamps in clock ticks for the timings of data frames recorded in the N-th data stream.
	UInt64	Offset from the beginning of the file of an optional metadata header for the N-th data stream. If there is no metadata associated with this data stream the value MUST be Zero.
	UInt8	Total number of data sections.
	UTF8String	Name of the first section.
	UInt64	Offset from the beginning of the file of the configuration header for the

		first section.
	UTF8String	Name of the second section.
	UInt64	Offset from the beginning of the file of the configuration header for the second section.
…	…	…
	UTF8String	Name of the N-th section.
	UInt64	Offset from the beginning of the file of the configuration header for the N-th section.

Table 4. Data Streams Definition

The *data stream* and *data section* names SHOULD contain only alpha numerical characters.

It is RECOMMENDED the configuration headers of the individual *data streams* and *data sections* to be placed immediately after the Data Section Definition and before the System Metadata Table. The data layout of the configuration headers of the individual data sections is content specific and is not defined in the FSTF format specification. The data section configuration header formats for the ADV file format is defined later in this document.

The frequency of the clocks together with the timestamps recorded as clock ticks can be used to determine the relative time of the start and end of each data frame. This time is not linked to UTC and it is acceptable during the recording the clock to experience small drifts related to temperature changes and/or other factors. A UTC timestamp may be recorded separately in the data frames. The ADV file format defines a separate absolute UTC timestamp.

The estimated timestamp precision should take into account the timing error of the recording hardware and software including various random delays such as the clock read time and others. This value has a meaning of an error, not a bias.

Data Stream Metadata

If the data streams have metadata associated with them, then the format of their metadata table is as follows:

Offset	Type	Description
0x00	UInt8	Number of entries in the metadata table.
0x01	UTF8String	Name of the first metadata entry.
	UTF8String	Value of the first metadata entry.
	UTF8String	Name of the second metadata entry.
	UTF8String	Value of the third metadata entry.
…	…	…
	UTF8String	Name of the N-th metadata entry.
	UTF8String	Value of the N-th metadata entry.

Table 5. Data Stream Metadata

Here is an example of a Header and Data Stream Definition blocks that define two data streams called MAIN (with 163 data frames) and CALIBRATION (with 4 data frames). The data streams have no metadata and the clocks used to timestamp the data frames have a frequency of 76900 Hz and system has a timestamping accuracy of 77 ticks (1 ms). The data frames saved in the streams have 2

sections called IMAGE and STATUS. The offset of the Index Table is 0x000000000028C463, the offset of the System Metadata Table is 0x000000000028C507 and the offset of the User Metadata Table is 0x0000000000320A35.

There are 2 metadata entries for the MAIN streams with names Name1 and Name2 and values of Христо (in Cyrillic) and Frédéric. There is 1 metadata entry for the CALIBRATION stream with a name Name1 and a value of 好的茶 (in Chinese Simplified).

The IMAGE section header is at 0x00000000000000C7, the STATUS section header is at 0x000000000000011C, the MAIN stream metadata table is at 0x0000000000000085 and the CALIBRATION stream metadata table is at 0x00000000000000B1.

All values in the first table below are hexadecimal and the second table displays the corresponding ASCII codes for clarity where non-displayable characters are shown with a dot.

46	53	54	46	02	00	00	00	00	63	C4	28	00	00	00	00
00	07	C5	28	00	00	00	00	00	35	0A	32	00	00	00	00
00	02	04	00	4D	41	49	4E	A3	00	00	00	64	2C	01	00
00	00	00	00	4D	00	00	00	85	00	00	00	00	00	00	00
0B	00	43	41	4C	49	42	52	41	54	49	4F	4E	04	00	00
00	64	2C	01	00	00	00	00	00	4D	00	00	00	B1	00	00
00	00	00	00	00	02	05	00	49	4D	41	47	45	C7	00	00
00	00	00	00	00	06	00	53	54	41	54	55	53	1C	01	00
00	00	00	00	00	02	00	00	00	05	00	4E	61	6D	65	31
0C	00	D0	A5	D1	80	D0	B8	D1	81	D1	82	D0	BE	05	00
4E	61	6D	65	32	0A	00	46	72	C3	A9	64	C3	A9	72	69
63	01	00	00	00	05	00	4E	61	6D	65	33	09	00	E5	A5
BD	E7	9A	84	E8	8C	B6									

F	S	T	F	.	.	.	.	.	C	.	(	.	.	.	.
.	.	.	(	.	.	.	.	.	5	.	2	.	.	.	.
.	.	.	.	M	A	I	N	.	.	.	.	d	,	.	.
.	.	.	.	M	.	.	.	.	.	.	.	.	.	.	.
.	.	C	A	L	I	B	R	A	T	I	O	N	.	.	.
.	d	,	.	.	.	.	.	M	.	.	.	.	.	.	.
.	.	.	.	.	.	.	I	M	A	G	E	.	.	.	.
.	.	.	.	.	.	S	T	A	T	U	S	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	N	a	m	e	1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
N	a	M	e	2	.	.	F	r	.	.	d	.	.	r	I
c	.	.	.	.	.	.	N	a	m	e	1	.	.	.	.
.	.	.	.	.											

Data Block

The obtained data is saved in the Data Blocks as sequence of frames. There is one frame for each interval where data was obtained. Each frame consists of one or more sections of data. A frame MAY contain more than one section of data and each frame MUST have exactly the same number and order of sections of data. A section could contain for example the image bytes and/or other information such as a collection of measurements including timestamps, settings of the equipment and any other information that can be useful and goes with the frame. All data sections must be present in the file for each and every frame. If for a given frame there is no data in a specific section then the section is still present but blank.

Data Blocks section SHOULD be placed immediately or close after the System Metadata Table. The FSTF file doesn't provide the offset of the Data Blocks sections in the Header however the offset is given in the Index Table for each data frame (see below).

Data frames SHOULD be placed sequentially in the Data Blocks and a usage of padding between the data frames is OPTIONAL.

The entries in the Index Table are used to find the position of each data frame. Further to this each data frame starts with the same magic number which makes it possible to find the data frames even when the Index Table hasn't been saved in the file for example due to a power outage during the recording.

It is RECOMMENDED all the *data frames* to be recorded in the order they are received and processed by the recording software regardless of which data stream they belong to. The Index Table at the end of the file is structured per *data stream* and is used to find the position of sequential *data frames* in a *data stream*.

The format of the individual data frames is presented below:

Offset	Type	Description
0x00	UInt32	Magic value - this is the word 0xEE0122FF which identifies the beginning of a data frame.
0x04	UInt8	The Id of the data stream to which this data frame belongs to.
0x05	Int64	Reference Timestamp associated with the beginning of the data frame represented as clock ticks elapsed since the clock reference time (usually system start).
0x0D	Int64	Reference Timestamp associated with the end of the data frame represented as clock ticks elapsed since the clock reference time (usually system start).
0x15	Bytes	Contents of the first section of the data frame.
	Bytes	Contents of the second section of the data frame.
	...	...
	Bytes	Contents of the N-th section of the data frame.

Table 6. Data frame format

The offsets given in the table above are from the beginning of the individual data frame. The offset of the data frame and the total length of the data frame bytes are given in the Index Table. The four bytes used for the magic value identifying the start of the frame are not included in the bytes counts saved in the Index Table.

Padding bytes CAN be used between data blocks for example to achieve particular boundary alignment of the data.

Index Table

The Index Table contains the positions and sizes for each data frame saved in the file and is structured per data stream. The offset of the Index Table from the beginning of the file is given in the Header. The Index Table has the following format:

Offset	Type	Description
0x00	UInt8	Number of data streams.
0x01	UInt32	Offset from the beginning of the Index Table of the index block corresponding to the first data stream.
0x05	UInt32	Offset from the beginning of the Index Table of the index block corresponding to the second data stream.
	…	…
	UInt32	Offset from the beginning of the Index Table of the index block corresponding to the N-th data stream.

Table 7. Index table

Each Index Block has the following format:

Offset	Type	Description
0x00	UInt32	Number of index entries.
0x04	Bytes16	First frame index entry.
0x14	Bytes16	Second frame index entry.
0x24	Bytes16	Third frame index entry.
…	…	…
	Bytes16	N-th frame index entry.

Table 8. Index block

Where each frame index entry contains 20 bytes in the following format:

Offset	Type	Description
0x00	UInt64	Elapsed time in data stream clock ticks since the first frame in the data stream.
0x08	UInt64	The offset of the data frame from the beginning of the file.
0x10	UInt32	The length of the data frame in bytes excluding the data frame MAGIC id.

Table 9. Index entry

The elapsed time SHOULD NOT be used to calculate the timestamp of a frame based on the first frame timestamp. The timestamp value given at the beginning of each data frame should be used instead. The elapsed time is provided so Reader Software can display visually the relative position of any frame in the file and also to do a search for a frame based on the time elapsed from the beginning of the recording.

Metadata Tables

The System and User Metadata tables contain series of name/value pairs referred to in this specification as ***Tags***. The type of the information saved as System Metadata Tags is controlled by the recorder that created the file. Examples of such information include the name and version of the recoding software, version of the hardware, geographical coordinates and others. The System

Metadata Table SHOULD be placed after the Data Section Definition block and before the Data Blocks.

The User Metadata Table MUST be at the very end of the file so any Reader Software can easily add new Tags and update existing Tags. Reader Software SHOULD NOT update any of the entries in the System Metadata Table.

The two Metadata Tables have the following format:

Offset	Type	Description
0x00	UInt32	Number of Metadata tags.
0x04	UTF8String	Name of the first tag.
	UTF8String	Value of the first tag.
	UTF8String	Name of the second tag.
	UTF8String	Value of the second tag.
	...	...
	UTF8String	Name of the N-th tag.
	UTF8String	Value of the N-th tag.

Table 10. Metadata entries

Reconstruction of a Corrupted File

In cases the Index Table and User Metadata Table MAY be missing. This could happen if the software didn't complete the recording due to an interruption for example caused by a power failure. In such a case it could be possible for a Reader Software to restore the Index Table by searching the FSTF file for the data frame magic number (0xEE0122FF). Data frames contain the timestamp and duration of the exposure and this information is sufficient to build the Index Table entries.

In a case of an interrupted recording the User Metadata Table MAY be missing as well however it SHOULD be empty for a new file.

Any software that reads the data frames SHOULD handle gracefully the case where a data frame hasn't been flushed completely to the disk as a result of an interrupted recording.

To determine whether the recording has been interrupted a Reader Software could use the following tests:

- The offset of the Index Table or User Metadata Table specified in the Header are either not specified (are zero) or appear to be after the end of the file
- The Index Table could not be read correctly

If the software reconstructs the Index Table of a file it MAY also add an entry into the User Metadata Table about this reconstruction adding information such as:

- Date when the reconstruction was done
- Reason for reconstructing the file
- Identification of the software and/or user that has reconstructed the file

ADV File Format

The FSTF file format defines a flexible and extensible data layout for storing scientific data. However to be more useful the file format needs to be extended and the data formats of the Data Stream Definition and Data Blocks need to be defined for a specific usage.

Astronomical video recording involves the use of CCD and video cameras to record a sequence of images (which we will call a video) in more than 8bit images. Often the individual images (which we will call video frames) are time-stamped using a UTC time reference (usually GPS time). The ADV file format is defined next with a purpose to improve the information storage capabilities in astronomical videos.

The ADV file uses two data streams called *MAIN* and *CALIBRATION* and two data sections called *IMAGE* and *STATUS*. The *CALIBRATION* data stream is used to store optional calibration frames such as bias, dark and flat frames or others, while the *MAIN* data stream is used for the main video frames.

The *IMAGE* section stores the pixels from each video frame and the *STATUS* sections stores any additional measurements/information associated with the corresponding image from the same data frame. The ADV file format was created with the intension for it to allow a number of different use cases, including: very fast recording of uncompressed images at high frame rates and recording compressed data that would minimize the total file size for long records at lower frame rates. There are two lossless compression algorithms defined for use in the current specification.

The ADV format is also extensible allowing the format to define new data structures in the future for example when better cameras or better data compression becomes available or when a particular need arises.

IMAGE Section

The name of the first section is IMAGE in capital letters. The UTF8String hexadecimal bytes of the name of the section are:

05	00	49	4D	41	47	45

.	.	I	M	A	G	E

The configuration of the IMAGE section is saved immediately after the Data Stream Definition block. The offset of the configuration from the beginning of the file is given in the Data Section Definition for the *IMAGE* section. The configuration has the following format:

Offset	Type	Description
0x00	UInt8	Version of the *IMAGE* section. This specification defined version 02.
0x01	UInt32	The width of the video images in pixels.
0x05	UInt32	The height of the video images in pixels.
0x09	UInt8	The native bits per pixels (BPP) of the data acquired from the camera. This may be different from the BPP used to save individual data frames. For example an SDK used to control a particular video camera may return images in 16 bit format but the values in the pixels may be 12 bit with a

		maximum pixel value of 4095. At the same time the camera may use a 14 bit A/D converter internally. In this case the BPP saved in the Image Section will be 12, the BPP saved in the Image Layout will be 16 and a separate and optional file metadata entry could be added to indicate that the camera A/D converter is 14 bit.
0x0A	UInt8	Number of Image Layouts defined in the file.
0x0B	UInt8	ID of the first defined Image Layout.
	Bytes	Configuration of the first Image Layout.
	UInt8	ID of the second defined Image Layout.
	Bytes	Configuration of the second Image Layout.
	...	...
	UInt8	ID of the N-th defined Image Layout.
	Bytes	Configuration of the N-th Image Layout.
	UInt8	Number of configuration name/value Tags.
	UTF8String	The name of the first *IMAGE* section tag.
	UTF8String	The value of the first *IMAGE* section tag.
	UTF8String	The name of the second *IMAGE* section tag.
	UTF8String	The value of the second *IMAGE* section tag.
	...	...
	UTF8String	The name of the N-th *IMAGE* section tag.
	UTF8String	The value of the N-th *IMAGE* section tag.

Table 11. Image section

The *IMAGE* section configuration defines some generic parameters such as the size and bit depth of the images and then defines specific Image Layouts that are used as format definitions when storing the video frames. Image Layouts allow different compression and pixel formats to be used for the video frames and with this to either achieve fast recording time or smaller file size. The *IMAGE* section also defines its own Tags that give extra information about the recording. The following Tags are defined:

Tag Name	Tag Value
IMAGE-BYTE-ORDER	Indicates the byte order of the pixel bytes. Possible values are the **LITTLE-ENDIAN** and **BIG-ENDIAN**. If this tag is missing the default value is **LITTLE-ENDIAN**.
IMAGE-MAX-PIXEL-VALUE	When present this tag indicates the maximum pixel value of the pixels of the saved images. For example a 16 bit image may have a maximum pixel value different than 65535. This could be a result of image stacking, binning or other pre-processing of the images before they are saved. When this tag is missing the maximum pixel value is determined by the BPP saved in the Image Section header.
IMAGE-BAYER-PATTERN	The type of the colour image array for raw colour camera images. If this tag is missing its assumed value is **MONOCHROME**. Other predefined values include **RGGB**, **GRBG**, **GBRG**, **BGGR, CMYG**, **CMYG2**, **YCMY**, **YMCY**, **MYYC** and **LRGB**. For more information see the Colour Coding section below.
SECTION-DATA-REDUNDANCY-CHECK	When a data redundancy check is used the value

	indicates the algorithm being used. When this Tag is present its value can be only **CRC32**. If the tag is missing then no data redundancy check is used.

Table 12. Image section tags

Each of the Image Layout configurations has the following format:

Offset	Type	Description
0x00	UInt8	Version of the Image Layout configuration. This specification defines version 02.
0x01	UInt8	Bits per pixel (BPP) for each pixel saved in the Image Layout. This value indicates the number of bytes used to save a pixel in the layout. This may differ from the BPP of camera data saved in the Image Section but needs to be at least as big. For example the Image Section may define a BPP of 12 but if the 12 bit pixels are saved as 16 bit numbers then the BPP of the Image Layout will be 16.
0x02	UInt8	Number of name/value Tags of the Image Layout.
	UTF8String	Name of the first tag.
	UTF8String	Value of the first tag.
	UTF8String	Name of the second tag.
	UTF8String	Value of the second tag.
	...	...
	UTF8String	Name of the N-th tag.
	UTF8String	Value of the N-th tag.

Table 13. Image layout

All specifics of an Image Layout are given in Tag values and the following Tags are used:

Tag Name	Tag Value
DATA-LAYOUT	The type of the data layout used. Currently supported layouts are **FULL-IMAGE-RAW**, **12BIT-IMAGE-PACKED** , **8BIT-COLOR-IMAGE** and **IMAGE-ROIS.** More formats may be defined in the future for example to allow differential coding using key frames. The three supported layouts will be described below. This Tag MUST be present.
SECTION-DATA-COMPRESSION	Specifies the used compression algorithm. Possible values are **UNCOMPRESSED**, **LAGARITH16** and **QUICKLZ**. This Tag MUST be present.
ROI-COUNT	When this Tag is present and the value is greater than zero it indicates that one or more Region-Of-Interests (ROI) have been recorded in the image layout. This Tag is OPTIONAL.
ROI-WIDTH-n	If **ROI-COUNT** is specified then this Tag MUST be present and it defines the width of the **n**-th recorded ROI, where **n** is zero based.
ROI-HEIGHT-n	If **ROI-COUNT** is specified then this Tag MUST be present and it defines the height of the **n**-th recorded ROI, where **n** is zero based.

ROI-TOP-n	If **ROI-COUNT** is specified then this Tag MUST be present and it defines the zero-based vertical position of the first row of pixels within the **n**-th recorded ROI, where **n** is zero based.
ROI-LEFT-n	If **ROI-COUNT** is specified then this Tag MUST be present and it defines the zero-based horizontal position of the first column of pixels within the **n**-th recorded ROI, where **n** is zero based.

Table 14. Common image layout tags

When each data frame is saved in the ADV file, in accordance with the generic FSTF format described above, the data frame begins with the 4 byte magic number followed by the Id of the data stream and then by the two 8 byte start and end clock tick timestamps. After that follows the data from the IMAGE and STATUS sections, which are specific for the ADV format and are defined next:

Offset	Type	Description
0x00	UInt32	The size of the *IMAGE* data block starting from the next byte (following immediately).
0x04	UInt8	The Image Layout Id used to save the pixels in the current image.
0x05	UInt8	The type of the recorded frame. This field is reserved for future use and currently MUST have a value of 0.
0x06	Bytes	Pixel values according to the Data Layout used (see below).
	UInt32	The size of the *STATUS* data block starting from the next byte (following immediately).
	Bytes	Data associated with the STATUS data block.

Table 15. Image layout data block

Colour Coding

Colour images can be saved directly in a FULL-IMAGE-RAW or 12BIT-IMAGE-PACKED data layouts in their respective Bayer Pattern as specified in the IMAGE-BAYER-PATTERN Image Layout tag. They can be also saved in the 8BIT-COLOR-IMAGE layout in RGB or BGR pattern where each red, green and blue colour is saved as an 8bit value. An RGB and BGR value for the IMAGE-BAYER-PATTERN Image Layout tag is only valid for an 8BIT-COLOR-IMAGE data layout.

Compression

If the specified Image Layout uses compression (the value of the SECTION-DATA-COMPRESSION tag is not UNCOMPRESSED) then the bytes need to be decompressed first. The bytes that are compressed are only the actual pixels saved in the layout. Here is an example of the binary content of a compressed ADV video frame:

FF	22	01	EE	00	84	0B	2A	AC	BC	0B	00	00	0B	0D	E7
BC	0B	00	00	10	06	00	00	02	00	XX	XX	XX	XX	XX	XX
XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

The data block starts with the magic value 0xEE0122FF followed by the stream id, in this case 0x00. Next follow the two 16-bit tick-stamps of the start and end of the frame – in this case 0x00000BBCAC2A and 0x00000BBCE70D. The data block length is 0x00000610 bytes long and the layout id is 0x02 followed by a reserved byte with a value of 0x00. Next is the compressed image which bytes are shown with 'XX'. These are the bytes that need to be decompressed with the

corresponding algorithm. The compressed bytes MAY contain information such as the size of the decompressed or compressed block or other information specific to the used compression algorithm.

Currently the ADV standard supports two methods of compression 'QUICKLZ ' and 'LAGARITH16' but more methods could be added in the future.

The QuickLZ algorithm is an open source fast compression library developed by Lasse Mikkel Reinhold. The library is available free for open source projects under GPL 1, 2 and 3 or under commercial license for closed source or commercial projects. The source code of the library in a number of different languages is available at http://www.quicklz.com/.

The Lagarith16 algorithm has been developed by Ben Greenwood specifically for the ADV file format and is released under MPL 2.0 license as a part of the **AdvCore** library.

After a decompression (where required) the pixels are interpreted according to the specified image layout pixel data format. The data format for the FULL-IMAGE-RAW, 12BIT-IMAGE-PACKED and 8BIT-COLOR-IMAGE types is given below. If SECTION-DATA-REDUNDANCY-CHECK is specified in the Image Section tags at the end of the pixel data section a 4 byte CRC32 value is present.

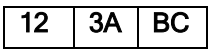
FULL-IMAGE-RAW

This image layout is supported for 16bit and 8bit pixel values. The data block contains 2 * Width * Height bytes for 16bit pixel values (i.e. 2 bytes per pixels) and Width * Height bytes for 8bit pixels (i.e. 1 byte per pixel). Pixels in a row are given in order from left to right and rows are added to the data block from top to bottom. The order of the bytes in the 16 bit pixel values is determined by the IMAGE-BYTE-ORDER Image Layout tag value. The default byte order, when IMAGE-BYTE-ORDER is not specified, is little-endian.

12BIT-IMAGE-PACKED

This image layout packs 12bit data using 3 pixels per two 12 bit values. The data block contains 3 * Width * Height / 2 bytes. Pixels in a row are given in order from left to right and rows are added to the data block from top to bottom. The IMAGE-BYTE-ORDER Image Layout tag value is ignored for this image layout and the encoded bytes always use the following format:

The two 12 bit values: 0x0123 and 0x0ABC are encoded as:



8BIT-COLOR-IMAGE

This image layout is used to store 24 bit colour images (8 bits for each colour). The data block contains 3 * Width * Height bytes where each pixel is represented by 3 bytes which could be either in order RGB or BGR, depending on the IMAGE-BAYER-PATTERN Tag of the Image Section which must be present if this Image Layout is used to store images and must have a value of either RGB or BGR. Pixels in a row are given in order from left to right and rows are added to the data block from top to bottom. The IMAGE-BYTE-ORDER Image Layout tag value is ignored for this image layout.

Image ROIs

When the **ROI-COUNT** and the other four **ROI-** related Tags are defined the image layout data block consists of the concatenated blocks of the individual ROI data in order with no gaps between them.

The pixels of each of the ROIs are saved in a block according to the specified **DATA-LAYOUT** for the Image Layout (see above).

STATUS Section

The name of the second section is STATUS in capital letters. The UTF8String hexadecimal bytes of the name of the section are:

06	00	53	54	41	54	55	53

.	.	S	T	A	T	U	S

The configuration of the *STATUS* section is saved immediately after the Data Section Definition block. The offset of the configuration is given in the Data Section Definition for the *STATUS* section. The configuration has the following format:

Offset	Type	Description
0x00	UInt8	Version of the *STATUS* section configuration. This specification defines version 02.
0x01	UInt64	Absolute accuracy of the UTC time-stamping in nanoseconds. This should take into account possible random delays and other timing issues of the recording system. The value together with the exposure duration can be used by a Reader Software to derive the timing error of the UTC timestamps associated with the data frames. This value has a meaning of an error, not a bias.
0x05	UInt8	Number of status entries in the section.
	UTF8String	Name of the first status entry.
	UInt8	Data type of the first status entry.
	UTF8String	Name of the second status entry.
	UInt8	Data type of the second status entry.
	...	...
	UTF8String	Name of the N-th status entry.
	UInt8	Data type of the N-th status entry.

Table 16. Status section

The names of the status entries SHOULD be short but descriptive and indicate the type of the value being saved in this entry. The data type is given with the Data type configuration entry. The following data types are defined:

Data Type	Data Type Code	Description
0	Int8	Signed 8 bit number.
1	Int16	Signed 16 bit number.
2	Int32	Signed 32 bit number.
3	Int64	Signed 64 bit number.
4	Real	A 32 bit floating point single precision number (IEEE 745).
5	UTF8String	A UTF8 string with a maximum length of 65535 characters and two leading bytes indicating the number of bytes in the UTF8 string literal.

Table 17. Status section entries data types

The data block representing the *STATUS* section has the following format:

Offset	Type	Description
0x00	UInt32	The size of the STATUS section data block for the current frame.
0x04	UInt64	UTC timestamp for the middle of the exposure represented as the number of nanoseconds elapsed since 1 Jan 2010, 00:00:00.000 Universal Time (JD 2455197.5).
0x0C	UInt32	Exposure duration in nanoseconds.
0x10	UInt8	Number of status entries given in the current data block. This value can be zero if no status entries are given.
0x11	UInt8	The 0-based index of the first recorded status entry.
	UTF8String	The value of the first recorded status entry.
	UInt8	The 0-based index of the second recorded status entry.
	UTF8String	The value of the second recorded status entry.
	...	...
	UTF8String	The 0-based index of the N-th recorded status entry.
	UInt8	The value of the N-th recorded status entry.

Table 18. Status section data block

The values of some or all of the defined status entries MAY be omitted. The “blank” status section data block, which only contains the UTC time stamp (X) and exposure (Y), has a length of 17 bytes and looks like this:

01	00	00	00	XX	XX	XX	XX	XX	XX	XX	XX	YY	YY	YY	YY	00

The following standardized list of status entries may appear in the ADV file. Additional status entries may be defined as needed. In order to support a maximum interoperability of ADV files across systems if a status entry that needs to be saved in the file is given in the table below then the exact Name and Type MUST be used as per the table.

Status Entry Name	Data Type	Value Units and Meaning
Gain	Real	Used gain in decibels by the camera.
Gamma	Real	Used gamma correction by the camera.
Shutter	Real	The shutter value in milliseconds used by the camera.
Offset	Real	An image offset level in percentages used by the camera.
SystemTime	Int64	A low precision local time of the system when the frame was saved presented as the number of nanoseconds elapsed since 1 Jan 2010, 00:00:00.000 Universal Time.
VideoCameraFrameId	Int32	An internal sequential number corresponding to the current video frame as provided by some cameras.
HardwareTimerFrameId	Int32	An internal sequential number assigned to the current frame and provided by some timing devices.
TrackedSatellites	Int8	The number of satellites tracked by the timing module.
AlmanacStatus	Int8	A status flag indicating if the leap second correction to

		UTC is known/certain. Possible values are '0' = uncertain and '1' = certain.
AlmanacOffset	Int8	The almanac offset in seconds. This is the difference in leap seconds between the satellite UTC time and the receiver known amount of leap seconds.
SatelliteFixStatus	Int8	A status flag for the satellite fix. It could be '0' for no fix, '1' for internal time keeping in a condition of a no fix after establishing a previous time or position fix, '2' for a time fix or '3' for time and position fix.
Error	UTF8String	A human readable error message.

Table 19. Standard ADV status entries

Metadata Tags

The following is a standardized list of System Metadata Tags that may appear in the ADV file. In order to support a maximum interoperability of ADV files across systems if a metadata tag that needs to be saved in the file is given in the table below then the exact Name and value format MUST be used as per the table. Additional tags can be defined as needed using FITS header keys whenever.

As the binary format of all System Metadata tag values is an UTF8String an additional Type which indicates how the value should be interpreted is defined. There are three possible types – *string*, *integer* and *real*. All values of type *real* MUST use a dot as a decimal separator and MUST NOT have a group separator. Exponential form (e.g. -3.12342E04) is allowed for *real* numbers.

Tag Name	Type	Tag Value
RECORDER-SOFTWARE	string	The name of the recording system.
RECORDER-SOFTWARE-VERSION	string	The version of the recording software running on the system.
RECORDER-HARDWARE	string	The name of the used hardware system.
RECORDER-HARDWARE-VERSION	string	The version of the used hardware system.
CAMERA-MODEL	string	The model of the used video camera.
CAMERA-SERIAL-NO	string	The serial number of the used video camera.
CAMERA-VENDOR-NAME	string	The vendor name of the video camera.
CAMERA-SENSOR-INFO	string	Information about the sensor of the video camera as a free text. It may include model, dimensions, pixel size, etc.
CAMERA-FIRMWARE-VERSION	string	The version of the firmware running on the video camera.
CAMERA-DRIVER-NAME	string	The name of the software driver used to control the video camera.
CAMERA-DRIVER-VERSION	string	The version of the software driver used to control the video camera.
LONGITUDE	real	The Geographical Longitude in degrees where the observation was made.
LATITUDE	real	The Geographical Latitude in degrees where the observation was made.
ALTITUDE	real	The altitude above the sea level in meters as a floating point number.
NTP-SERVER-LIST	string	A comma delimited list of the names of NTP servers that have been used for NTP timing.

NTP-SERVER-INFO	string	Free text information about the servers used for NTP timing which may include for example response times.
BINNING-X	integer	Camera horizontal pixel binning as an integer number.
BINNING-Y	integer	Camera vertical pixel binning as an integer number.
AUTHOR	string	Author of the file.
COMMENT	string	Generic descriptive comment.
EPOCH	real	Equinox of celestial coordinate system. Same as EQUINOX.
EQUINOX	real	Equinox of celestial coordinate system. Same as EPOCH.
INSTRUMENT	string	Name and details about the instrument .
OBSERVER	string	Name of the observer.
TELESCOPE	string	Name and details about the telescope.
RA	real	Right ascension of the observed object in hours.
DEC	real	Declination of the observed object in degrees.
OBJNAME	string	Name of the observed object.

Table 20. Standard ADV metadata entries