A COMMUNITY FORMAT FOR ELECTRO-OPTICAL SPACE SITUATIONAL AWARENESS (EOSSA) DATA PRODUCTS

Tamara Payne, Shaylah Mutschler, Neal Shine
Applied Optimization
Daniel Meiser, Roberto Crespo
NASIC/GSMS
Elizabeth Beecher, Lawrence Schmitt
AFRL/RYWW

ABSTRACT

In this paper, we present a flexible format for compiling radiometry/photometry data with pertinent information about the collections into a file for use by the Space Situational Awareness (SSA) community. With the increase in the number of Electro-Optical (EO) sensors collecting photometric, radiometric, and spectroscopic data on man-made Resident Space Objects (RSOs) for SSA purposes, the EO SSA community of interest and stakeholders in SSA require a file format protocol for reporting the extracted information used for SSA from these datasets.

This EOSSA file format provides a foundation to enable data providers to format their processed data. The objective of this format is to handle a variety of photometric measurements from multiple sensors and provide fields for specific parameters containing crucial data about the object, the sensor, the collection, and the processing.

The chosen formatting type for EOSSA is the Flexible Image Transport System (FITS). It is maintained by the International Astronomical Union and NASA/GSFC. FITS is the standard data format used in astronomy and has extensions and features that make it easy to transport and archive large scientific data sets. There are types of FITS files for multi-dimensional arrays, such as images, or hyperspectral image cubes, and headers and tables for data extracted from the images, and descriptive information about the data and sensor. The FITS binary table extension is the most efficient data structure to use for the purposes of SSA with respect to ease of programming, computational speed, and storage space. A hierarchical data format (HDF5) has many of these features; however, its biggest drawback to our purpose is that the files are large and require a lot of storage space. Secondly, no standardized HDF5 file structure has been developed and there is no high level application programming interface (API).

1. INTRODUCTION

With the increase in the number of Electro-Optical (EO) sensors collecting photometric, radiometric, and spectroscopic data on man-made Resident Space Objects (RSOs) for Space Situational Awareness (SSA) purposes, the EO SSA community of interest and stakeholders in SSA require a file format protocol for reporting the extracted information used for SSA from these datasets. We have developed a sensor-independent file format standard to meet the community's current and evolving needs to ingest these datasets and provide a common data format for analysis tool developers. This file format for EO SSA data products is thusly named EOSSA.

There are various sensors producing photometric data products of various types from various missions. As such, a format that is consistent, contains required information for pedigree, captures observing conditions, and yet is flexible is required. Since EO observations vary both in size and type depending on the sensor and/or collection mode, a standardized and extensible format must be able to handle the variability. The EOSSA format has been developed to accommodate any of these varieties. With a standardized format that includes all the relevant information on RSO collections, such as time, sensor location, target location, calibrations, etc., as well as the EO measurements themselves, analysis tools can be more readily developed and tested for RSO characterization with reduced cost.

The file specification document for EOSSA provides a foundation to enable data providers to format their processed data into the EOSSA format (Ref. 1). The objective of this format is to handle a variety of photometric measurements from multiple sensors and provide fields for specific parameters containing crucial data about the object, the sensor, the collection, and the processing. These parameters are essential for applying the EO phenomenology to identify and classify RSOs, detect and resolve anomalies, and detect and track RSO status changes.

The chosen format [Flexible Image Transport System (FITS)] is maintained by the International Astronomical Union and NASA/GSFC (Ref. 2). FITS is an open standard data format used in astronomy and has extensions and features that make it easy to transport and archive large scientific data sets. There are types of FITS files for multi-dimensional arrays, such as images, or hyperspectral image cubes, and headers and tables for data extracted from the images, and descriptive information about the data and sensor. The FITS binary table extension is the most efficient data structure to use for our purposes, both with respect to ease of programming, computational speed, and storage space (Ref. 2).

From the FITS Support Office web site, the following quote describes the motivation behind the development of FITS. "The Flexible Image Transport System (FITS) evolved out of the recognition that a standard format was needed for transferring astronomical data from one installation to another. The original form, or Basic FITS, was designed for the transfer of images and consisted of a binary array, usually multidimensional, preceded by an ASCII text header with information describing the organization and contents of the array. The FITS concept was later expanded to accommodate more complex data formats. A new format for image transfer, random groups, was defined in which the data would consist of a series of arrays, with each array accompanied by a set of associated parameters."[1]

FITS is organized in a series of blocks. A FITS block is a sequence of 2880 bytes aligned on 2880 byte boundaries in the FITS file; blocks are most commonly either a header block or a data block. Each FITS structure consists of an integral number of FITS blocks. A FITS file is composed of a primary Header and Data Unit (HDU), which is a required feature of every FITS file. The primary HDU starts with the first FITS block of the FITS file. A FITS file may also contain conforming extensions and other special records, both of which are optional (e.g., an EOSSA file must contain a BINTABLE extension). A FITS file containing one or more extensions following the primary HDU is sometimes referred to as a Multi-Extension FITS (MEF) file. The first FITS block of each subsequent FITS structure shall be the FITS block immediately following the last FITS block of the preceding FITS structure. For example, in an EOSSA file, the first FITS block of the BINTABLE extension FITS structure is the FITS block immediately following the FITS block containing the primary HDU (Ref. 2). For longer descriptions and more details, see the FITS Standard document and the User's Guide.[2]

2. EOSSA FORMAT STRUCTURE

An EOSSA file contains two structures: a primary header and a binary table extension. Both of these structures contain fields that are required by the FITS format as shown in Table 1 andTable 2, respectively; however, the binary table extension contains additional fields that distinguish it from a more commonly known FITS image file.

Table 1. FITS Primary Header Keywords

#	Keyword	Description	Example
1	SIMPLE	The value field shall contain a logical constant with the value T if the file conforms to FITS standard. This keyword is mandatory for the primary header and must not appear in extension headers. A value F signifies that the file does not conform to FITS standard.	SIMPLE = T
2	BITPIX	The value field shall contain an integer. The absolute value is used in computing the sizes of data structures. It shall specify the number of bits that represent a data value in the associated data array. For valid values of BITPIX see Ref. 1. Writers of FITS arrays should select a BITPIX data type	BITPIX = 16

[1] http://fits.gsfc.nasa.gov/fits_overview.html
[2] http://fits.gsfc.nasa.gov/standard30/fits_standard30aa.pdf and http://fits.gsfc.nasa.gov/users_guide/usersguide.pdf

		appropriate to the form, range of values, and accuracy of the data in the array.	
3	NAXIS	The value field shall contain a non-negative integer no greater than 999 representing the number of axes in the associated data array. A value of zero signifies that no data follow the header in the HDU.	NAXIS = 2

4	NAXISn, n = 1,…,NAXIS	The NAXISn keywords must be present for all values n = 1,…, NAXIS, in incrementing order of n, and for no other values of n. The value field of this indexed keyword shall contain a non-negative integer representing the number of elements along axis n of a data array. A value of zero for any of the NAXISn signifies that no data follow the header in the HDU. If NAXIS is equal to 0, there shall not be any NAXISn keywords.	NAXIS1 = 250
			NAXIS2 = 300

	Custom	Placeholder for other non-required fields within FITS file format or fields that the data provider might want to define. Such as those defined in order to create an EOSSA FITS document.	DATE = '2006-10-22'
last	END	This keyword has no associated value. Bytes 9 through 80 shall be filled with ASCII spaces. The END keyword marks the logical end of the header and must occur in the last 2880-byte FITS block of the header.	END

Table 2. FITS Binary Table Extension Header Keywords

#	Keyword	Description	Example
1	XTENSION	The value field shall contain a character string giving the name of the extension type. This keyword is mandatory for an extension header and must not appear in the primary header.	XTENSION=␣'BINTABLE'[3]
2	BITPIX	(same description as Table 1)	BITPIX = 8
3	NAXIS	(same description as Table 1)	NAXIS = 2

4	NAXISn, n = 1,…, NAXIS	(same description as Table 1)
		NAXIS1 is the number of bytes in each row for a BINTABLE extension
		NAXIS2 is the number of rows in data table for a BINTABLE extension
			NAXIS1 = 410
			NAXIS2 = 10 (for a 10 row table)

5	PCOUNT	The value field shall contain an integer that shall be used in any way appropriate to define the data structure, consistent with Eq. 1. In IMAGE extensions this keyword must have the value 0; in BINTABLE extensions it is used to specify the number of bytes that follow the main data table in the supplemental data area called the heap.	PCOUNT = 0
6	GCOUNT	This keyword must have the value 1 in the IMAGE and BINTABLE extensions.	GCOUNT = 1

[3] The ␣ and single quotes do not need to be specified by the provider. This is done by the library that aids in writing the FITS/EOSSA file. See the FITS Standard for more information on the XTENSION keyword and the formatting shown (Ref. 1).

8	TFIELDS	The value field shall contain a non-negative integer representing the number of fields (columns) in each row of the data table.	TFIELDS = 1
9	TFORMn	The value field of this indexed keyword shall contain a character string of the form *r*T. It describes the value which will be in the nth field/column of the data table. Reference below for details.	TFORM1 = 27A
10	TTYPEn	The value field of this indexed keyword shall contain a character string giving the name of field n.	TTYPE1 = UTC_Begin_Exp
11	TUNITn	The value field shall contain a character string describing the physical units in which the quantity in field n is expressed.	TUNIT1 = char
	Custom	Placeholder for other non-required fields within FITS file format or fields that the data provider might want to define. Such as those defined in order to create an EOSSA FITS document.	(EXTNAME, CLASSIF, VERS,..etc.)
last	END	(same description as Table 1)	END

The total number of bits in the extension data array (exclusive of fill that is needed after the data to complete the last 2880-byte data block) is given by the following expression:

$$N_{bits} = |BITPIX| \times GCOUNT \times (PCOUNT + NAXIS1 \times NAXIS2 \times \ldots \times NAXIS_m), \qquad \textbf{Eq. 1}$$

where N_{bits} must be non-negative and is the number of bits excluding fill; m is the value of NAXIS; and BITPIX, GCOUNT, PCOUNT, and the NAXISn represent the values associated with those keywords. If $N_{bits}>0$, then the data array shall be contained in an integral number of 2880-byte FITS data blocks. The header of the next FITS extension in the file, if any, shall start with the first FITS block following the data block that contains the last bit of the current extension data array (Ref. 1).

The binary table extension structure is composed of two parts: a header and a data unit (or table data). The binary table extension header contains both FITS specific keywords and EOSSA specific keywords; the EOSSA specific keywords are shown in Table 3. The EOSSA specific keywords within the binary table extension header are the metadata of the data contained in the actual table of the binary table extension. The metadata in the header is used to describe the data within the table in its entirety. For example, header keywords include the security classification of the table data (CLASSIF), the telescope site name for which the data was collected from (TELESCOP), and the TLE of the target object for which has been collected for (TLELN1 and TLELN2).

Table 3. EOSSA Binary Table Extension Header Required Keywords

#	Req'd.	Keyword	Description	Units	Format	Example(s)
1	All	EXTNAME	Filename of the FITS binary extension table file		A	
2	All	CLASSIF	Security classification level of the data contained in the file		A	'UNCLASS', 'CONF', 'SECRET', etc.
3	All	VERS	Version number of the EOSSA data format		A	'1.0', '2.5', '3.0'

4	All	OBSEPH	Observer type, i.e. ground-based, space-based with a TLE for the sensor, or space-based with a state vector for the sensor		A	‘GROUND’, ‘TLE’, ‘STATE’
5	All	TELESCOP	Telescope site name or SSN sensor identifier.		A	‘AMOS’, ‘RME’, ‘SENSOR510’
6	G	TELLAT	Geographical latitude of the telescope	degrees N	D	
7	G	TELLONG	Geographical longitude of the telescope	degrees E	D	
8	G	TELALT	Distance above sea level of the telescope	m	D	
9	All	OBSNAME	Telescope name		A	‘SENSOR510’, ‘AEOS’, ‘MT16’, ‘RMERaven’,etc.
10	All	OBJEPH	Target object ephemeris source		A	'STATE', 'TLE'
11	All	OBJTYPE	If the target has a space catalog number, the string ‘SCN’ is the value. If another catalog is used to identify the object, the name of that catalog is the value. Otherwise ‘NULLSTRING’ is the value.		A	'SCN', ‘NULLSTRING’
12	All	OBJNUM	If OBJTYPE = ‘NULLSTRING’, -2147483648 is the value. Otherwise, the identification number of the object from the catalog in OBJTYPE is the value.		J	12345
13	All	OBJECT	Name of target object		A	‘GALAXY14’
14	All	TLELN1	Target Object Truncated TLE Line 1. First line of TLE without preceding '1' (67 characters). ‘NULLSTRING’ for UCTs.		A	
15	All	TLELN2	Target Object Truncated TLE Line 2. Second line of TLE without preceding '2' (67 characters). ‘NULLSTRING’ for UCTs.		A	

The binary table extension data unit is comprised solely of EOSSA specific fields. These fields are defined below in Table 4. This data unit can be thought of as a table comprised of rows and columns. Table 4 simply defines the columns of the table, and the each row represents a single observation of the target object. Examples of columns of the data table are start time of the exposure for that observation (UTC_Begin_Exp), the exposure time for that observation (Exp_Duration), and the range normalized magnitude of the object for that observation (Mag_Range_Norm).

Table 4. EOSSA Binary Table Extension Data Column Descriptions

#	**Req’d.**	**TTYPEn values** (data columns)	**Description**	**Units**	**Format** (part T of TFORM)	**Example(s)**
1	All	UTC_Begin_exp	The start time of the exposure in UTC	yyyy-mm-ddThh:mm:ss{.sssssss}	A	‘2013-08-01T12:13:14.000’

2	All	UTC_End_exp	The end time of the exposure	yyyy-mm-ddThh:mm:ss{.sssssss}	A	‘2013-08-01T12:14:14.000’
3	All	JD_Mid_Exp	The time of mid-exposure in JD. Be sure that the values for this keyword have enough significant digits to express the time to the appropriate fraction of a second		D	2456506.00937
4	All	Exp_Duration	Length of the integration or exposure time	sec	D	30.0
5	All	Cur_Spec_Filt_Num	The reference number, n, in CALFILn for the currently used spectral filter		J	1
6	All	Cur_ND_Filt_Num	The reference number n, in NDFNAMn for the currently used neutral density filter		J	2
7	All	Mag_Exo_Atm	Exo-atmospheric magnitude is the magnitude of the target on the standard scale with atmospheric effects removed.	mag	D	13.21
8	All	Mag_Range_Norm	Range-normalized magnitude (Mag in #7) using 1000 km	mag	D	3.45
9	All	Eph_RA_DE	Predicted Right Ascension and Declination of the Target object from the frame of reference of the sensor (J2000, geocentric velocity aberration). SGP4 and VCMs produce geocentric origin and velocity aberration and subtracting the sensor geocentric position of the sensor places in its reference frame.	deg	2D	[75.33; -5.001]
10	All	Met_RA_DE	Measured Right Ascension and Declination of the Target object from the frame of reference of the sensor.	deg	2D	[75.31; -5.201]
11	G	Eph_AZ_EL	Predicted Azimuth and elevation angles of the target object from a ground-based sensor (no atmospheric refraction correction required). AZ_EL implies apparent	deg	2D	[92.89; 32.12]

			topocentric place in true of date reference frame as seen from the observer with aberration due to the observer velocity and light travel time applied.			
12	G	Met_AZ_EL	Measured azimuth and elevation angles of the target object from a ground-based sensor (no atmospheric refraction correction required)	deg	2D	[92.91;32.76]
13	G	Sun_AZ_EL	Azimuth and elevation angles of the sun from a ground-based telescope (no atmospheric refraction correction required)	deg	2D	[273.91; -25.77]
14	All	Tel_Obj_Range	Distance from the telescope to the target object during the observation	m	D	3.578683E7

3. BUILDING YOUR OWN EOSSA FILE

In this section, we discuss the process of building an EOSSA file from the sensor data. The details for assembling the metadata of the collections and calculating the additional parameters to fully populate all the EOSSA fields are described in Reference 1.

3.1 Data Collection Considerations and Image Data

The creation of an EOSSA file actually starts from the collection of images at the telescope; typically FITS is used to capture the image and metadata pertinent to the collection, e.g. time, location, filter, exposure time, array size, etc. Many of the fields in the original raw image FITS file should be captured by the image processing software for copying into the EOSSA output file. Many sensors, however, use HDF5 format to capture images and store the metadata. Again, many of the fields captured at the time of collection merely need to be read and stored by the image processing software to be written into the EOSSA file. EOSSA is for assembling and storing the extracted radiometric/photometric and astrometric information from the sensor data collection. Therefore, it contains the output of the specific image processing/extraction software used to calibrate and process the sensor data.

3.2 Library Choices

There are several open-source FITS libraries available for various programming languages. Below is a brief and compressed table showing some of the library options available for C, C++, Java, and MATLAB development. This is not a comprehensive list; this table, in its entirety, and more information can be found on the FITS website[4]. Table 5 shows available tools that will allow the creation of FITS binary extension tables and thus, can be used to build software to read and write EOSSA files.

Table 5. FITS Libraries available in certain Programming Languages for Writing EOSSA Files

Language and Library Name	Read	Write	Other capabilities: R(read), W(write)

[4] http://fits.gsfc.nasa.gov/fits_libraries.html

C++ & C; CFITSIO	yes	**yes**	R&W Images, Groups, ASCII Table, and Var. Len. Arrays
Java; nom.tam.fits	yes	**yes**	R&W Images, Groups, ASCII Table, and Var. Len. Arrays
MATLAB; built-in	yes	**no**	R&W Images, only R ASCII and Binary Table
Python; PYFITS	yes	**yes**	R&W Images, Groups, ASCII Table, and Var. Len. Arrays

As shown above in the table, C++, C, Java, and Python all have extensive FITS libraries that allow the user to read and write EOSSA files. However, MATLAB currently only has the capability to read an EOSSA file, not write one.

3.3 Putting It All Together

When implementing software to write an EOSSA file, there is a certain flow of thought and code organization that one should follow in order to simplify the process. It is recommended that the data that is going to be written into an EOSSA file be stored in an organized structure that emulates the structure of an EOSSA file. This structure should be created in the language in which the image processing is executed so that the metadata and extracted radiometry can be easily transferred into the structure. This structure should contain binary table header fields that are grouped together and consist of only scalar variables. Similarly, the binary table data should be grouped together and consist of only array or vector variables. It is not necessary to include the primary header and the FITS specific fields within the structure because these fields are automatically calculated and populated by most FITS writer libraries. The FITS library chosen is primarily dependent on the language that is performing the image processing and then reliant on which of those libraries support writing a FITS binary table extension. A condensed list of libraries can be found in Table 5; this list covers the major programming languages, which includes languages used for IRAF and Source Extractor: Java, C, C++, Python, and MATLAB. There are FITS libraries available for other various programming languages as well. After selecting a library, utilize this library in order to write your data structure to a FITS file. If all the EOSSA required fields are within this FITS file, an EOSSA file has been successfully created.

4. SUMMARY

In this paper, we presented a flexible format, EOSSA, for compiling radiometry/photometry data with pertinent information about the collections into a file for use by the Space Situational Awareness (SSA) community. This format provides a foundation to enable data providers to format their processed data and provides a standard format for analysis tools. The EOSSA format structure was presented and how to build an EOSSA file was discussed.

5. REFERENCES

1. Electro-Optical Space Situational Awareness (EOSSA) File Format Description Document, Version 3.1.1. Release 2, July 2014
2. Pence , W.D., Chiappetti , L., Page, C. G., Shaw, R. A., Stoble, E., 2010, A&A, 524, A42