SAND2014-17270R

GFF Specification for SAR Systems

Sandia National Laboratories
Albuquerque, NM 87185-0519

REVISION I 08/13/2014

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

This page intentionally left blank.

Revision History

Date	Rev.	Description	Author
2-20-08	A	Creation	JDM/TB
7-02-08	B	Cleanup to match the code	JDM
2-16-10	C	Bumped minor revision for momeas extension additions.	JDM
7-01-12	D	Clarified endian flag	HJC
8-22-13	E	Added the compression enumerations	LZP
9-26-13	F	Add the undocumented enumerations values and updated the minor version to 5. Add the pixelDataType field to the spec.	LZP
4-09-14	G	Added the pointers written into the header blocks (unused, but present in all GFF files)	HJC
8-05-14	H	Removed OUO/ECI markings.	NLS
8-13-14	I	Added Sandia and DOE logos; added Sandia funding statement.	NLS

Table of Contents

Acronyms and Terms

Acronym	Description
SAR	Synthetic Aperture Radar
GSAT	Ground-based SAR Applications Test bed
GFF	GSAT File Format

1 Introduction

1.1 Purpose

To provide a well documented common image format for SAR systems such that images and this document can be distributed to interested observers so the format can be well understood.

1.2 Scope

This document provides the basic GFF format definition for a GFF file. This document covers the contents of the main header and the basic GFF file structure. The contents of specific extensions are covered in a separate document.

2 GFF File Format

A GFF file consists of a main header, zero or more header extensions and image data.

The main header contains all the information necessary to correctly read and display the image data.

Header extensions contain metadata that is not necessary to display the image data but could be useful for further image exploitation. All header extensions are optional.

The image data is the last block of data in the file and the main header will indicate how the data is stored.

Version numbers are very important in GFF as they indicate which readers/writers are valid for this file. The main header and header extensions contain their own version numbers and are independent of one another.

3 Basic building blocks

The main header, all header extensions and the image data are preceded by the common GFF tag structure shown in Table 1. As shown, the GFF common tag structure contains fields that indicate the identity of the data, the version number (major and minor) and the size of the data NOT including the size of the GFF common tag structure. The information in the GFF tag structure allows GFF file readers to skip over a header extension if the reader does not understand or care what type of data is in the header extension.

Table 1. GFF Common Tag Structure.

Field name	Type	Elements	Description
systemID	char	16	Unique ID indicating the type of data stored in the block
versMajor	unsigned short	1	Major version of the data in the block
versMinor	unsigned short	1	Minor version of the data in the block
unused	uint32	1	*Reserved*
numBytes	int32	1	Size of the data in the block
unused	uint32	1	*Reserved*

4 GFF Header Definition

4.1 Main Header

The contents of the main header of the GFF specification are defined below. All GFF files must contain at least the main header and image data. The main header contains all the information necessary to correctly read and display the image data. The contents of the main header are shown below divided up into sections for clarity. In the actual GFF file, there are no divisions between the data in the different sections. The different sections of the main header are:

- basic GFF file information

- image size, type and compression
- pixel size and type
- pixel scaling

The main header is preceded by the GFF Common Tag Structure with the default values shown in Table 2.

Table 2. GFF Common Tag Structure Values for the Main Header.

Field name	Default	Comments
systemID	"GSATIMG"	This field must match exactly or the file is an invalid GFF file.
versMajor	2	Main header major version
versMinor	5	Main header minor version
unused	0	*Reserved*
numBytes	82	The size of the main header structure. sizeof(GFFHeader_t)
unused	0	*Reserved*

4.1.1 Basic GFF File Information

This section of the main header contains the name of the image, the number of bytes from the start of the GFF file to the image data and the endian (big or little) of all data in the GFF file. Table 3 shows the detailed contents of this section.

Table 3. Basic GFF file information in the main header.

Field name	Type	Elements	Description
endian	enum (int32)	1	Indicates how the data is stored – bit 0 indicates endian-ness (0=big, 1=little), bit 1 indicates word size (0=32-bit, 1=64-bit). i.e. endian: 0=big endian, 32-bit; 1=little endian, 32-bit; 2=big endian, 64-bit; 3=little endian, 64-bit
imageCreatorLen	unsigned short	1	Length of the image creator field
imageCreator	char	24 (IMAGE_CREATOR_LEN)	Text representing the system that created this image. For example it could be software project and version (i.e. "Project 1.01")

4.1.2 Image size, type and compression

This section contains information describing the size of the image data in pixels, the order of the pixels (row major or column major), the size of the image data in bytes, and the image compression scheme.

Table 4. Image size, type and compression information in the main header.

Field name	Type	Elements	Description
rangePixels	unsigned int32	1	Number of rows in the image
azPixels	unsigned int32	1	Number of columns in the image
pixOrder	unsigned int32	1	Specifies the pixel order, azimuth or range major: eRANGE_CONSECUTIVE = 0 eAZ_CONSECUTIVE = 1
imageLengthBytes	int32	1	The number of bytes that the attached image data occupies. This field is needed because the image data may be compressed.

imageCompressionScheme	enum (int32)	1	The type of compression used on the attached image (the header is never compressed). JPEG2000 uses the licensed Kakadu SDK.
			`eNO_COMPRESSION (0)`
			`eJPEG (1)`
			`eZLIB (2)`
			`eJPEG2000 (3)`

The pixel order requires more explanation due to the amount of confusion encountered during development. Image pixels can be oriented in two different ways in the GFF specification either azimuth consecutive (Figure 1) or range consecutive (Figure 2). Azimuth consecutive refers to pixels stored in memory consecutively in rows where range consecutive indicates that pixels are stored in memory consecutively in rows. Data once retrieved may have to be transposed to get the data in the desired format for use.

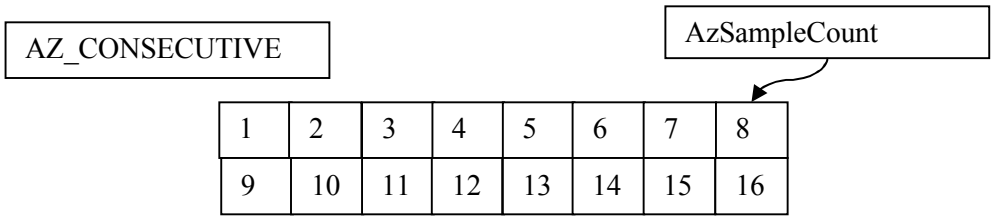

Figure 1 Azimuth Consecutive

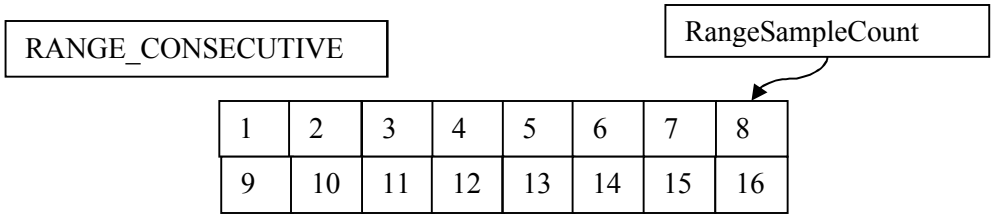

Figure 2 Range Consecutive

4.1.3 Pixel Size and Type

This section contains information about the individual pixels of the image data. The information in this section indicates the data type of the pixels (float, int), the information contained in each pixel (magnitude, complex), how the information in each pixel is stored (phase first, magnitude first), and the number of bits representing the data in each pixel.

4.1.3.1 Pixel Data Type

Table 5 Pixel Data Type Enumeration Definition

Field name	Type	Elements	Description
pixDataType	Enum (in32)	1	Stores the pixel data type. Each enumeration indicates the data type of the pixel and the information of each pixel (complex or magnitude) eMAG_UCHAR = 0 eMAG_PHASE_USHORT = 1 eCOMPLEX_USHORT = 2 eCOMPLEX_UINT = 3 eCOMPLEX_ULONG = 4 eMAG_CHAR = 5 eMAG_PHASE_SHORT = 6 eCOMPLEX_SHORT = 7 eCOMPLEX_INT = 8 eCOMPLEX_LONG = 9 eCOMPLEX_SINGLE = 10 eCOMPLEX_DOUBLE = 11 eMAG_PHASE_UCHAR = 12 eMAG_PHASE_CHAR = 13 eUNDEFINED_PIXEL_TYPE = 14

4.1.3.2 Pixel Format

Table 6 Pixel Data Type Definition

Field name	Type	Elements	Description
bitSize	unsigned short	1	Size of the pixel component in bits. This field will be used more once compression is added to the library, but

			for now it is redundant information.
dataType	enum (int32)	1	Specifies the way the data is stored eUCHAR8 = 0 eUSHORT16 = 1 eUINT32 = 2 eULONG64 = 3 eCHAR8 = 4 eSHORT16 = 5 eINT32 = 6 eLONG64 = 7 eFLOAT32 = 8 eDOUBLE64 = 9

Field name	Type	Elements	Description
compFormat	Pixel Data Type	numComponents Structure is allocated to MAX_PIX_COMPONENTS (Currently set to two)	Array of numComponents **Pixel type elements. Each element is defined in Table 5 Pixel Data Type Enumeration Definition**

Field name	Type	Elements	Description
pixDataType	Enum (in32)	1	Stores the pixel data type. Each enumeration indicates the data type of the pixel and the information of each pixel (complex or magnitude) eMAG_UCHAR = 0 eMAG_PHASE_USHORT = 1 eCOMPLEX_USHORT = 2 eCOMPLEX_UINT = 3 eCOMPLEX_ULONG = 4 eMAG_CHAR = 5 eMAG_PHASE_SHORT = 6 eCOMPLEX_SHORT = 7 eCOMPLEX_INT = 8 eCOMPLEX_LONG = 9 eCOMPLEX_SINGLE = 10 eCOMPLEX_DOUBLE = 11

						eMAG_PHASE_UCHAR = 12 eMAG_PHASE_CHAR = 13 eUNDEFINED_PIXEL_TYPE = 14

Table 5 Pixel Data Type Enumeration Definition

Field name	Type	Elements	Description
pixDataType	Enum (in32)	1	Stores the pixel data type. Each enumeration indicates the data type of the pixel and the information of each pixel (complex or magnitude) eMAG_UCHAR = 0 eMAG_PHASE_USHORT = 1 eCOMPLEX_USHORT = 2 eCOMPLEX_UINT = 3 eCOMPLEX_ULONG = 4 eMAG_CHAR = 5 eMAG_PHASE_SHORT = 6 eCOMPLEX_SHORT = 7 eCOMPLEX_INT = 8 eCOMPLEX_LONG = 9 eCOMPLEX_SINGLE = 10 eCOMPLEX_DOUBLE = 11 eMAG_PHASE_UCHAR = 12 eMAG_PHASE_CHAR = 13 eUNDEFINED_PIXEL_TYPE = 14

Table 7. Pixel size and type information in the main header.

Field name	Type	Elements	Description

cmplxDomain	enum (int32)	1	Defines whether the two components form a rectangular (IQ) or polar (MP) coordinate system and which components are present in this image. Allowable values: IQ, QI, MP, I1Q2, Q1I2, M1P2, P1M2, M, P.
			IQ, QI, MP indicate interleaved data.
			I1Q2, Q1I2, M1P2 and P1M2 indicate that components are stored in bands.
			`eIQ = 0`
			`eQI = 1`
			`eMP = 2`
			`eI1Q2 = 3`
			`eQ1I2 = 4`
			`eM1P2 = 5`
			`eP1M2 = 6`
			`eM = 7`
			`eP = 8`

numComponents	int32	1	Defines the number of cmplxDomain components in this structure. Should follow the Description definitions for possible cmplxDomain values. (IQ numComponents=2)

4.1.4 Pixel Scaling

This section describes the scaling of the pixel data.

Table 8. Pixel scaling information in the main header.

Field name	Type	Elements	Description
pixValLin	int32	1	Describes the operation the pixels of this image have undergone (magnitude, sqrt(magnitude), etc.)
autoScaleFac	float	1	Value that input values are multiplied by. Can be used to back out original values.

4.2 Image Data

The image data section of the GFF header contains the actual pixel data for the image. Every GFF file must have an image data section. Table 9 shows the required values of the GFF common tag structure for image data.

Table 9. GFF common tag structure values for image data.

Field name	Default	Comments
systemID	"IMAGEDATA"	Tag identifying this extension
versMajor	2	Image data major version
versMinor	0	Image data minor version
p_data	0	Unused
numBytes	azPixels * rngPixels (unless compressed)	Number of bytes in the image data not including this header.
p_next	0	Unused

5 Header Extensions

Header extensions are added to the GFF file by adding header extensions to the file and filling in the GFF common tag structure as appropriate for the extension. Adding header extensions should have no impact on the basic GFF functionality.

6 References

GFF Data Extensions Reference
GFF Developers Guide