LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

- LIGO –

CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
&
VIRGO EXPERIMENT

CNRS-INFN

Document Type Technical Note	LIGO-T970130-v4 VIR-1129B-21	November 18, 2022

Specification of a Common Data Frame Format for Interferometric Gravitational Wave Detectors (IGWD)
(Version 9)

LIGO Data and Computing Group
Virgo Data Acquisition Group

VIRGO CNRS/INFN
Traversa H di Via Macerata
56021 S. Stefano a Macerata
Cascina (PI), Italy
Phone (39) 050 752 521
Fax (39) 050 752 550

California Institute of Technology
LIGO Laboratory - MS 100-36
Pasadena CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Laboratory - MS NW17-161
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

www: http://www.ligo.caltech.edu/

www: http://www.virgo.infn.it

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

The LIGO/VIRGO Data Frame Format for interferometric gravitational wave detectors (IGWD) is a collaborative effort that has evolved out of a frame format design originated within the VIRGO Project. This specification has evolved out of the recognition for the need of a standard definition of this frame format that can be used by individual (international) projects wishing to adhere to a common representation of data produced by IGWD. It is hoped that by using a standard design for data, future collaborative analyses of data taken by different projects can be promoted more easily.

The predominant type of data stored in frames is time series data of arbitrary duration. It is possible, however, to encapsulate in frame structures other types of data, e.g., spectra, lists, vectors or arrays, etc. However, the primary purpose of this specification is to address how (raw) data are written into frame structures.

It is the intent of the Projects collaborating internationally on this frame definition and standardization to promote a continued evolution of the standard through formal configuration control, scheduled revisions and releases, and code maintenance. A Consortium or Working Group with representatives from each of the participating projects will be formed and will have formal control over the contents of this specification as it evolves.

1.1 Purpose

The primary intent of the Frame Format is to capture the informational content of real-time data acquisition systems associated with interferometric gravitational wave detectors to efficiently archive that information.

As experience in using frame data within the gravitational wave community develops, the informational content of Frames will need to grow to support newly identified needs through the addition of new structures. It is the intent of this specification to present a foundation to frame-based data which will only be modified to remove errors and to fill in missing components; new ways of organizing future data needs will be accommodated through the addition of new structures, rather than the evolution of existing (and working) structures. In doing this, it will be more easily possible to support older frame formats at the same time while accommodating newer ones within the same frame libraries.

1.2 Scope

Frames are written assuming IEEE/ASCII compliant hardware and software are used to read/write data.

This standard specifies the organization and content of IGWD Frame data sets, including the C structures which create a frame. It is only a specification of the frame storage and not a library specification of how precisely data will be stored in memory.

This specification also defines rules to which new extensions and revisions are required to conform.

1.3 Applicability

LIGO and VIRGO will work to ensure that all developed hardware and software systems will support IGWD Frames ("frames") for the interchange of binary data. All participating projects will acquire their data in Frames and make their data available, when and if data exchanges occur, in Frame formatted media. There is no restriction in media type. Reduced data still containing time-series representation of IGWD datastreams shall be made available in Frames. The Frame format shall be available in the public domain, subject only to the standards and controls defined herein.

Chapter 2

Applicable Documents

Table 1: Applicable Documents

Document Identifier	**Description**	**Comments**
VIRGO-MAN-LAP-5400-103	Frame Library Users Manual	
T970100	LIGO System Software Design Issues	
T970140	LIGO Systems Software Specifications and Design Requirements	In process

Chapter 3

List of Definitions, Acronyms and Symbols

Table 2: Acronyms, definitions and symbols used in this document

Acronym	Definition
ASCII	American Standard Code Information Interchange
ANSI	American National Standards Institute
C/C++	Programming languages
GPS[1]	Global Positioning System Time
IEEE	Institute of Electrical and Electronic Engineers
IGWD	Interferometric Gravitational Wave Detector(s)
LIGO	Laser Interferometric Gravitational Laboratory
VIRGO	VIRGO Experiment sponsored by CNRS (France) - INFN (Italy)
TAI	International Atomic Time
UT	Universal Time (GMT + 12^h)
UTC[2]	Universal Coordinated Time

[1] GPS time uses atomic time as its basis and equals TAI within an offset defining the GPS epoch. GPS = TAI + 19.000^s. GPS uses as its origin the standard epoch, 1980 January 6. ^{d}0, Julian date (JD) 2 444 244.5. JD = 0 corresponds to 4713 B.C., January 1.^{d}5.

[2] UTC uses the atomic second as its basis, but to keep UTC close to UT and civil time, integer leap seconds (of either sign) are added to UTC at distinct epochs. GPS and UTC were coincident at the GPS standard epoch, 1980 January 6.^{d}0. The integer number of leap seconds, N_S, between TAI and UTC in the present epoch is defined by the relationship: TAI - UTC = N_S 1^s.000.

Chapter 4

IGWD Frame Structure

This document specifies the frame format version shown in, valid with the release of this document. Subsequent updates to this document will indicate in this paragraph the valid Format Version Number.

Table 3: Frame Format Version Number

Frame Format version number for this specification	9

4.1 Overall

A Frame is a grouping of multiple C structures composed of the following elements:

- Frame header
- Dictionaries permitting reconstruction of the C structures via reading of frame data off media
- Frame history comment
- Detector/instrumental configuration
- Raw fast data
- Serial data
- Event data
- Post-processed/derived data
- Simulated data
- etc.

4.2 Data types

The following C data types are used in frames

Table 4: IGWD frame data types as written to media

Data Class[3]	**Nominal C/C++ Data Type**	**Length**[4]	**Comments**
CHAR	char	1	Character
CHAR_U	unsigned char	1	Unsigned character
INT_2S	signed short	2	Signed integer, Range: $(-2^{15}, 2^{15}-1)$
INT_2U	unsigned short	2	Unsigned integer, Range: $(0, 2^{16}-1)$
INT_4S	int	4	Signed integer, Range: $(-2^{31}, 2^{31}-1)$
INT_4U	unsigned int	4	Unsigned integer, Range:$(0, 2^{32}-1)$
INT_8S	long	8	Signed integer, Range: $(-2^{63}, 2^{63}-1)$
INT_8U	long	8	Unsigned integer, Range: $(0, 2^{64}-1)$
REAL_4	float	4	IEEE-defined single precision floating point number
REAL_8	double	8	IEEE-defined double precision floating point number
Composite Data Types			
STRING	char []	< 65536	Character string; first 2 bytes are interpreted as an INT_2U for length of string, exclusive of these two bytes but inclusive of the '\0' string terminator [5]
PTR_STRUCT	void*	6	Pointer to a structure. This object replaces an actual pointer when the structure is written to media (pointer address would be meaningless). Instead, a pair of int are written, to be interpreted as (data class, data instance) => (INT_2U, INT_4U) NULL == (0, 0) The frame reading software uses these two variable to rebuild a pointer table when the frame is read into memory.
COMPLEX_8	Pair of REAL_4	8	Complex real number, two single precision floats, stored as a pair: (real, imaginary)
COMPLEX_16	Pair of REAL_8	16	Complex real number, two double precision floats, stored as a pair: (real, imaginary)

Byte ordering of all integer and real types is determined by hardware and compiler options. To allow for optimal performance, the actual byte ordering in these frame data types will be free to be either big-endian (most significant byte first) or little-endian (least significant byte first). The actual ordering is encoded in the file header. It is required of the software to transparently determine and allow for translation between these conventions as needed on specific platforms. The ordering applies to individual elements of composite structures, but does NOT apply to ordering of composite elements themselves.

Code which writes and uses frames shall use the capitalized casts to the specified class variable definitions to ease the transportability of the code among platforms and operating systems to the greatest extent possible. I/O methods employed within frame libraries shall be written in a POSIX.1 compliant style. Frame structure assumes a minimum 32-bit computer architecture.

[3] The classes {INT_2S,..., STRING} inclusive, may also be used as lists of such objects. The notation in the specification will be to precede the data class with an asterisk (*): e.g., *INT_2U implies a list or array of INT_2U objects. The information on the length of the list will appear elsewhere within the header of the structure using such objects.

[4] Note: lengths indicated are the minimum lengths for these types; actual lengths must be determined by the encoding of types in the file header. Software assumes only ASCII character set usage.

[5] Note that ALL strings must have a '\0' terminator; even NULL strings. The '\0' character is NOT ALLOWED within a string. However, multiple contiguous '\0' characters are allowed at the end of the string. Nevertheless, program(s) that use and copy such strings are not required to keep these extra '\0' characters.

The structures and all supporting libraries shall conform to recognized standard C/C++ usage. The controlling standard for the C language is ANSI C. The controlling standard for the C++ language is ANSI-ISO C++. Note that all variables are case sensitive.

4.3 Composition of files and frames written to media

A file consists of binary data. Figure 1 shows a schematic representation of a data file as written to media. Figure 2 presents the pointer/structure schema upon which the frame is built. Note that structures stored in RAM have pointer elements associated with them (which are needed for memory allocation and usage in the machine) which are not written as addresses to media, but rather as PTR_STRUCT identifiers.

A file contains a header, frames, and an end of file:

File:{FileHeader, Frame1, Frame2,...,Frame$_m$,... Frame$_N$, EndOfFile}

4.3.1 File Header -- FrHeader

Table 6: Byte-level descriptor for a file header

Byte(s)	Description
0 - 4	ASCII Characters "IGWD" (string terminated with a \0) or other identifier of originator of frame file
5	Data format version for this file (refer to 4)
6	Frame Library minor version number for software used to write this file. The value 255 is reserve to represent unreleased or provisional (Beta) versions of the library.
7	Size of an INT_2 on originating hardware
8	Size of an INT_4 on originating hardware
9	Size of an INT_8 on originating hardware
10	Size of a REAL_4 on originating hardware
11	Size of a REAL_8 on originating hardware
12 - 13	2 bytes containing 0x1234. This is used to determine byte order differences between writing hardware and reading hardware
14 - 17	4 bytes containing 0x12345678. This is used to determine byte order differences between writing hard ware and reading hardware
18 - 25	8 bytes containing 0x123456789abcdef. This is used to determine byte order differences between writing hardware and reading hardware
26 - 29	IEEE single precision floating point representation of $\pi = 3.14159265358979323 84$.......
30 - 37	IEEE double precision floating point representation of $\pi = 3.14159265358979323 84$.......
38	Frame library used: 0=unknown, 1=frameL, 2=frameCPP, > 2 presently unused.
39	File checksum schemes: 0 = none, 1 = CRC, >1 presently unused. The file checksum is recorded in the FrEndOfFile structure.

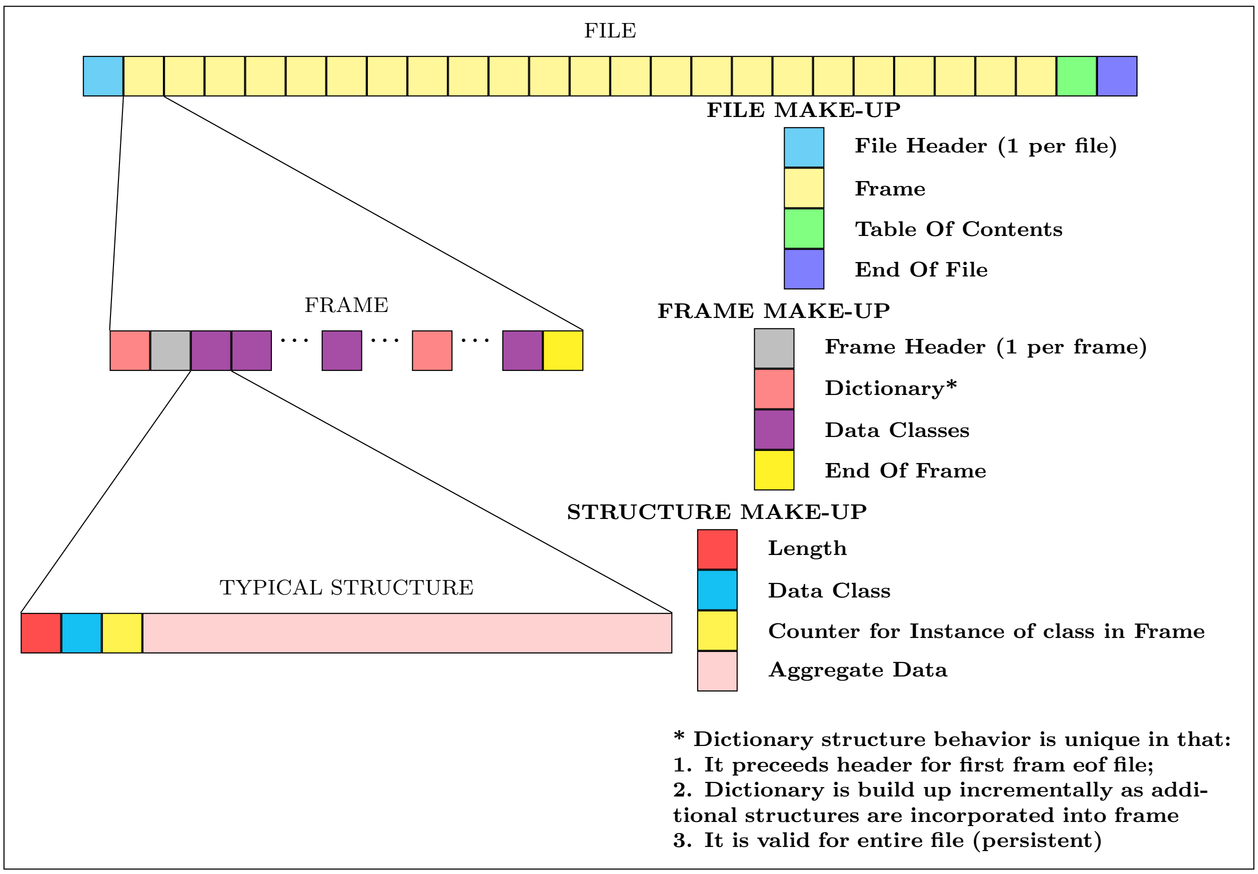

Figure 4.1: Schematic representation of data organization within a file.

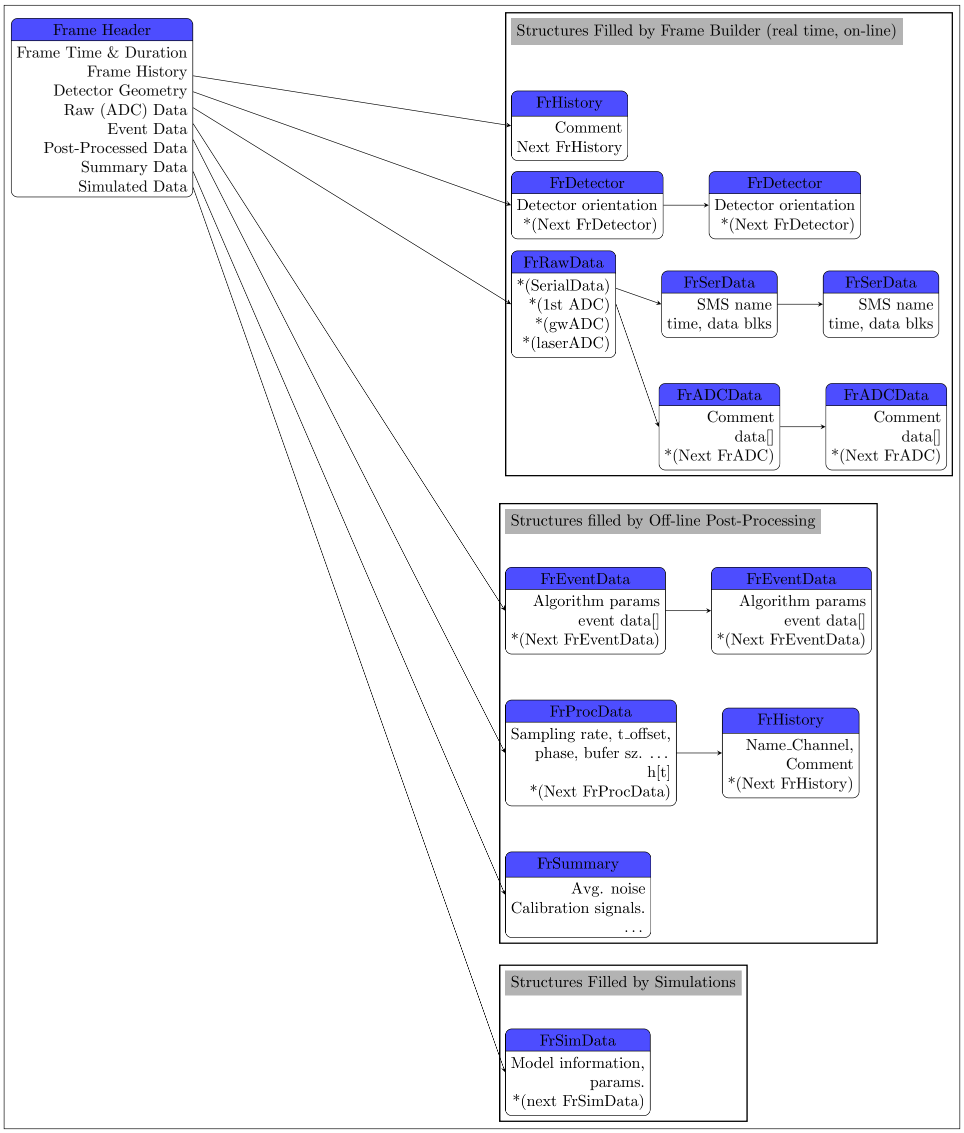

Figure 4.2: Schematic representation of frame structures and relative pointers (not all possible structures are shown).

4.3.2 Frame:{DictionaryStructure, FrameHeader, DictionaryStructure, Structure$_1$, Structure$_2$, ..., DictionaryStructure, ...,Structure$_N$, FrameEnd}

Data are written as frames that are composed of structures. There are a number of unique structures from which a frame may be built; not all possible structures appear in a particular frame.

Any structure that is used for the first time in a file requires that it be preceded by a corresponding dictionary-type structure describing it. Thus, each of the structure types introduced in paragraph 4.3.1 and following below must be described on media by one (and only one) dictionary structure containing the sequence: {FrSH, FrSE,..., FrSE}. There are as many FrSE elements in the sequence as there are elements of a structure in its corresponding table (except the first four rows of each table, which are used as structure headers -- see **Figure 1** and **Tale 6**). These dictionary structures are normally written immediately preceding the first occurrence of the corresponding structure.

Structure class numbers 1 and 2 correspond to FrSH and FrSE. The primitives FrSH and FrSE are themselves not described by dictionary structures on media, and must be known a priori to interpret a file on media. These primitive structures shall be maintained across revisions of frame-writing software libraries to maintain backwards compatibility with data.

With the exception of FrSH, FrSE, FrameH, FrEndOfFile, FrEndOfFrame, and FrTOC, all other structures appearing in a frame must be referenced by other structures.

Except for FrVect and FrTable, all instances of other structures which may appear more than once in a frame must be assigned a unique name variable.

All attributes appear in the order in which they are listed in the following tables. All structure attributes must be written contiguously to media. Ordering of structures is hierarchical so that all lower level structures which are referenced by a given structure must appear contiguously before the next structure appears at the same or higher level of the frame tree. Linked lists of structures must appear in monotonic order through the file.

Table 7: Common Elements of All Frame Structures

cmnelem Structure		
Data class	Variable Name	Descriptor & Comments
INT_8U	length	Byte length of this structure, including byte count of this variable
CHAR_U	chkType	Checksum schemes: 0 = none, 1 = CRC, >1 presently unused
CHAR_U	class	Structure class for this particular structure.
INT_4U	instance	Counter for occurrence of this class of structure within current frame or current file, starting from 0. NOTE: All instance counters are set to 0 after end of frame AND end of file.

4.3.2.1 Frame Structure Header -- FrSH

This is a structure containing the following data:

Table 8: Frame Structure Header Data

FrSH Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7 with class = 1		
STRING	name	Name of structure being described by this dictionary structure
INT_2U	class	Class number of structure being described
STRING	comment	Comment
INT_4U	chkSum	Structure checksum[6]

4.3.2.2 Frame Structure Element -- FrSE

This is a structure containing the following data:

Table 9: Frame Structure Element Data

FrSE Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7 with class = 2		
STRING	name	Name of an element of the structure being described by dictionary. All element names within the structure must be unique. NOTE: The first FrSE begins with row 4 for each of the tables below.
STRING	class	Literally contains "CHAR", "INT_2U",...
STRING	comment	Comment
INT_4U	chkSum	Structure checksum6

The structure types allowed are described below. The list will be revised as the frame design evolves.

4.3.2.3 Frame Header -- FrameH

This is a structure containing the following data:

[6] This checksum is calculated over the content of each structure, starting with the "length" variable and ending just before the chkSum variable. In other words, the chksum variable or any following variable like chkSumFile in FrEndOfFile is excluded from the checksum computation. Whenever a checksum is not calculated, the default value of chkSum is 0

Table 10: Frame Header Structure Definition

FrameH Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Name of project or other experiment description (e.g., GEO; LIGO; VIRGO; TAMA;...)
INT_4S	run	Run number (number $<$ 0 reserved for simulated data); monotonic for experimental runs.
INT_4U	frame	Frame number, monotonically increasing until end of run, re-starting from 0 with each new run.
INT_4U	dataQuality	A logical 32-bit word to denote top level quality of data. Lowest order bits are reserved in pairs for the various GW detectors [7]
INT_4U	GTimeS	Frame start time in GPS Seconds [8]
INT_4U	GTimeN	Frame start time residual, integer nanoseconds.
REAL_8	dt	Frame length in seconds
One or more of the pointers to the structures below may be NULL in any given Frame Header		
(FrVect *) PTR_STRUCT	type	Identifier for array used to store general info like the event type. This is a reserved parameter: the description is presently not specified.
(FrVect *) PTR_STRUCT	user	Identifier for array for user-provided information. Use is generic.
(FrDetector *) PTR_STRUCT	detectSim	Identifier for array storing model or simulation parameter data definition
(FrDetector *) PTR_STRUCT	detectProc	Identifier for detector-derived data. It is used to capture detector information for both raw and post-processed data.
(FrHistory *) PTR_STRUCT	history	Identifier for first history of post-processing with which frame may have been generated.
(FrRawData *) PTR_STRUCT	rawData	Identifier for the raw data structure
(FrProcData *) PTR_STRUCT	procData	Identifier for the first post-processed data
(FrSimData *) PTR_STRUCT	simData	Identifier for the first simulated data buffers
(FrEvent *) PTR_STRUCT	event	Identifier for the first event structure
(FrSimEvent *) PTR_STRUCT	simEvent	Identifier for the first simulated event data structure
(FrSummary *) PTR_STRUCT	summaryData	Identifier for the first statistical summary data
(FrVect *) PTR_STRUCT	auxData	Identifier for the first auxiliary data
(FrTable *) PTR_STRUCT	auxTable	Identifier for the first auxiliary table data
INT_4U	chkSum	Structure checksum 6

[7] See Appendix C A for bit assignments.
[8] GPS time in integer seconds since the GPS standard epoch. This is valid for 143 years.

4.3.2.4 ADC Data -- FrAdcData

This is a structure containing the following data:

Table 11: ADC Data Structure Definition

FrAdcData Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Channel name -- must be unique with the frame
STRING	comment	Comment
INT_4U	channelGroup	Channel grouping number containing ADC[9]
INT_4U	channelNumber	Channel number
1 INT_4U	nBits	Number of bits in A/D output
REAL_4	bias	DC bias on channel (Units @ ADC_counts = 0)
REAL_4	slope	ADC calibration: input units/ct
STRING	units	ADC calibration: input units for slope. If dimensionless, then units == <NONE>, in CAPITALS (without <...>).
REAL_8	sampleRate	Data acquisition rate, samples/s.
REAL_8	timeOffset	Offset of 1st sample relative to the frame start time (seconds) Must be positive and smaller than the frame length[10]
REAL_8	fShift	fShift is the frequency (in Hz) in the original data that corresponds to 0 Hz in the heterodyned series. [11]
REAL_4	phase	Phase (in radian) of heterodyning signal at start of dataset 1
INT_2U	dataValid	Data valid flag: dataValid = 0 -> ADC data valid; dataValid!= 0 -> ADC data suspect/not valid
(FrVect *) PTR_STRUCT	data	Identifier for a single vector of data as sampled from the instrument.
(FrVect *) PTR_STRUCT	aux	Identifier for vector for user-provided information; use is generic. [12]
(FrAdcData *) PTR_STRUCT	next	Identifier for next ADC structure in the linked list.
INT_4U	chkSum	Structure checksum 6

4.3.2.5 Detector Data -- FrDetector

This is a structure containing the following data:

[9] These two variables are determined by site and must be unique over all detectors

[10] Time offsets always added together (to obtain decimal time offset: time_of_1st_sample = (GtimeS + GtimeN/10^9) + timeOffset . Note that all quantities are ALWAYS positive.

[11] In the heterodyning process the real time series is multiplied by cos[2*pi*fShift(t-t_0)+phase] to get the real part and by –sin[2*pi*fShift(t-t_0)+phase] (note the minus sign) to get the imaginary part of the resulting complex time series. The time origin t_0 is the beginning of the frame

[12] As of version 9 of the frame specification, dataValid information is now encoded with the FrVect structure containing the data.

Table 12: Detector Data Structure Definition

FrDetector Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Instrument name as described in Appendix C (e.g., Virgo; GEO_600; TAMA_300; LLO_4k; CIT_40; simulated pseudo data - model version etc.)
CHAR[2]	prefix	Channel prefix for this detector as described in Appendix C.
REAL_8	longitude	Detector vertex longitude, geographical coordinates: radians; Value> 0 => E of Greenwich ($-\pi<$Longitude $<= +\pi$)
REAL_8	latitude	Detector vertex latitude, geographical coordinates: radians; Value > 0 => N of Equator ($-\pi/2 <$ Latitude $<= +\pi/2$)
REAL_4	elevation	Vertex elevation, meters, relative to WGS84 ellipsoid. [13]
REAL_4	armXazimuth	Orientation of X arm, measured in radians East of North (0<= ArmXazimuth $< 2\pi$)
REAL_4	armYazimuth	Orientation of Y arm, measured in radians East of North 0<= ArmYazimuth $< 2\pi$)
REAL_4	armXaltitude	Altitude (pitch) angle of X arm, measured in radians above horizon (local tangent to WGS84 ellipsoid) $-\pi/2 <$ ArmXaltitude $<= +\pi/2$.
REAL_4	armYaltitude	Altitude (pitch) angle of Y arm, measured in radians above horizon (local tangent to WGS84 ellipsoid) $-\pi/2 <$ ArmYaltitude $<= +\pi/2$.
REAL_4	armXmidpoint	Distance between the detector vertex and the middle of the X cavity (meters) (should be zero for bars)
REAL_4	armYmidpoint	Distance between the detector vertex and the middle of the Y cavity (meters) (should be zero for bars)
INT_2U	dataQualityOffset	Bit offset of the low order bit of the data quality mask.
(FrVect *) PTR_STRUCT	aux	Identifier for user-provided (presently undefined) structure for additional detector data..
(FrTable *) PTR_STRUCT	table	Identifier for user-provided (presently undefined) table structure for additional detector data. NOTE: Data quality may be defined using this structure with a 2-column FrTable. Column 1: String containing QA word definitions; Column 2: Value of QA words First entry is "Overall data quality" and has the following FOUR possible values: 0 = "Data QA not evaluated for these data". 1 = "Data should not be used for source searches", i.e. BAD data. 2 = "Data not 'perfect', but suitable for long time stretch searches". 3 = "Data 'perfect' - suitable for burst searches".
(FrDetector *) PTR_STRUCT	next	Identifier for next detector structure in the linked list
INT_4U	chkSum	Structure checksum 6

[13]If elevation is zero, no location information is provided

4.3.2.6 End of File Structure -- FrEndOfFile

This is a structure containing the following data:

Table 13: End of File Data Structure Definition

FrEndOfFile Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
INT_4U	nFrames	Number of frames in this file
INT_8U	nBytes	Total number of bytes in this file (0 if NOT computed)
INT_8U	seekTOC	Bytes to back up from the end of file to the beginning of the table of contents structure. If seekTOC == 0, then there is no TOC for this file.
INT_4U	chkSumTOC	FrTOC uniqueness checksum[14]
INT_4U	chkSumFrHeader	FrHeader checksum[15]
INT_4U	chkSum	Structure checksum Notice that the chkSum AND chkSumFile variables are not included in the computation of chkSum, but chkSumFrHeader is included.
INT_4U	chkSumFile	File checksum.[16]

The CRC should be POSIX.2 checksum as referred to in section 4.9 of P1003.2/D11.2,

(for example: http://ftp.optiva.ee/pub/misc/posix_drafts/p1003.2/d11.2/4.9).

4.3.2.7 End of Frame Data -- FrEndOfFrame

This is a structure containing the following data:

Table 14: End of Frame Data Structure Definition

FrEndOfFrame Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
INT_4S	run	Run number; same as in Frame Header run number datum.
INT_4U	frame	Frame number, monotonically increasing until end of run; same as in Frame Header frame number datum

[14]This checksum is calculated from the following fields of the FrTOC structure: nFrame, dt, nADC, name, nProc, nameProc, nSim, nameSim, nSer, nameSer, nSummary, nameSum, nEventType, nameEvent, nEvent, nTotalEvent, nSimEventType, nameSimEvent, nSimEvent, and nTotalSEvent It is used to determine if a previous table of contents structure can be used to represent information in this file.

[15]This checksum is calculated over the contents of the file header (FrHeader). The checksum type is the same as the file checksum specified in FrHeader. Whenever this checksum is not calculated, the default value is 0.

[16]This checksum is calculated over the contents of each file, starting with the file header (FrHeader) and ending with the FrEndOfFile structure, including the chkSum variable but excluding the chkSumFile variable. The checksum type is specified in FrHeader. Whenever a file checksum is not calculated (see FrHeader), the default value is chkSum =0.

FrEndOfFrame Structure (continued)		
Data class	Variable Name	Descriptor & Comments
INT_4U	GTimeS	Frame start time in GPS Seconds
INT_4U	GTimeN	Frame start time residual, integer nanoseconds.
INT_4U	chkSum	Structure checksum 6

4.3.2.8 Event -- FrEvent

This is a structure containing the following data:

Table 15: Data Structure Definition

FrEvent Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Name of event.
STRING	comment	Descriptor of event.
STRING	inputs	Input channels and filter parameters to event
process. INT_4U	GTimeS	GPS time in seconds corresponding to reference value of event, as defined by the search algorithm.
INT_4U	GTimeN	GPS time in residual nanoseconds relative to GtimeS.
REAL_4	timeBefore	Signal duration before (GTimeS.GTimeN) (seconds)
REAL_4	timeAfter	Signal duration after (GTimeS.GTimeN) (seconds)
INT_4U	eventStatus	Defined by event search algorithm.
REAL_4	amplitude	Continuous output amplitude returned by event
REAL_4	probability	Likelihood estimate of event, if available (probability = -1 if cannot be estimated)
STRING	statistics	Statistical description of event, if relevant or available.
INT_2U	nParam	Number of additional event parameters
REAL_8[nParam]	parameters	Array of additional event parameters (size of nParam).
STRING[nParam]	parameterNames	Array of parameter names (size of nParam).
(FrVecat *) PTR_STRUCT	data	Identifier for vector containing additional event results.
(FrTable *) PTR_STRUCT	table	Identifier for table structure containing additional event information.
(FrEvent *) PTR_STRUCT	next	Identifier for another event.
INT_4U	chkSum	Structure checksum 6

4.3.2.9 History Data -- FrHistory

This is a structure containing the following data:

Table 16: History Structure Definition

FrHistory Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Name of history record. Note: when an FrHistory is linked to an FrProcData, its name variable must be the FrProcData channel name
INT_4U	time	Time of post-processing, GPS time in integer seconds since GPS standard epoch.
STRING	comment	Program name and relevant comments needed to define post-processing.
(FrHistory *) PTR_STRUCT	next	Identifier for next history structure in the linked list.
INT_4U	chkSum	Structure checksum 6

4.3.2.10 Message Log Data -- FrMsg

This is a structure containing the following data:

Table 17: Message Log Data Structure Definition

FrMsg Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	alarm	Name of message, error flag or alarm state.
STRING	message	Message body
INT_4U	severity	Message severity level (To Be Defined); default = 0.
INT_4U	GTimeS	GPS time in seconds corresponding to this FrMsg
INT_4U	GTimeN	GPS time in residual nanoseconds relative to GtimeS
(FrMsg *) PTR_STRUCT	next	Identifier for next message structure in the linked list.
INT_4U	chkSum	Structure checksum 6

4.3.2.11 Post-processed Data -- FrProcData

This is a structure containing the following data:

Table 18: Post-Processed Data Structure Definition

FrProcData Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Data or channel name
STRING	comment	Comment

FrProcData Structure (continued)		
Data class	Variable Name	Descriptor & Comments
INT_2U	type	Type of data object: 0 - Unknown/user defined 1 - Time Series 2 - Frequency series 3 - Other 1D Series data 4 - Time-Frequency 5 - Wavelets 6 - Multi-dimensional 7 - MIME (Multipurpose Internet Mail Extensions)
INT_2U	subType	• Subtype for f-Series 0 - Unknown/user defined 1 - DFT 2 - Amplitude Spectral Density 3 - Power spectral density 4 - Cross spectral density 5 - Coherence 6 - Transfer function • TBD for other types
REAL_8	timeOffset	Offset of 1^{st} sample relative to the frame start time (seconds) Must be positive and smaller than the frame length.[17]
REAL_8	tRange	Duration of sampled data (tStop-tStart)[18]
REAL_8	fShift	fShift is the frequency in the original data that corresponds to 0 Hz in the heterodyned series.1,1,[19] In multidimensional objects this applies to the first frequency dimension
REAL_4	phase	Phase of heterodyning signal at start of dataset (radians, 0 if unknown)1
REAL_8	fRange	Frequency range (=fMax-fMin, 0 if unknown)1
REAL_8	BW	Resolution bandwidth (Sum{w[i]^2}/N where w[i] is the $i^{\{th\}}$ window coefficient (0 if unknown)
INT_2U	nAuxParam	Number of auxiliary parameters
REAL_8[nAuxParam]	auxParam	Array of auxiliary parameters (size of nAuxParam).
STRING[nAuxParam]	auxParamNames	Array of auxiliary parameter names (size of nAuxParam).[20]

[17]Time offsets always added together (with suitable scaling of timeOffsetN) to obtain decimal time offset: time_of_1^{st}_sample = (GtimeS + GtimeN/10^9) + timeOffsetS. If FrVect contains a t-series, then time_of_1^{st}_sample = (GtimeS + GtimeN/10^9) + timeOffsetS + startX[0]. Note that all quantities are ALWAYS positive.

[18]tRange, fShift, fRange are redundant with the axis information in the data Vector in some cases. If a redundancy exists, the data must be identical to that in the FrVect for the earliest time and/or lowest frequency dimension (e.g. for a t-Series tRange = dx[0]*nx[0]). If a discrepancy exists then the FrVect values take precedence

[19]If FrVect contains a f-series then the frequency offsets fShift and startX[0] are added together to obtain the frequency in the original data, original_frequency_of_1st_sample = fShift + startX[0], while startX[0] gives the offset in frequency in the heterodyned data of the 1st sample

[20]auxParamNames may contain only the following ASCII characters: "a-z", "A-Z", "0-9", "_", "-", ":", "#", "$", "@", and must begin with an ALPHA character (a-z, A-Z).

FrProcData Structure (continued)		
Data class	Variable Name	Descriptor & Comments
(FrVect *) PTR_STRUCT	data	Data vector. • The data vector for single dimensional types (t-Series and f-Series) must not be a linked list. • For MIME data, the value of the unitY member will be the MIME type of the data. • The concatination operation for MIME data is defined as extending the link list of data elements with the new elements.
(FrVect *) PTR_STRUCT	aux	Auxiliary data; use is generic.
(FrTable *) PTR_STRUCT	table	Parameter table
(FrHistory *) PTR_STRUCT	history	Channel history.[21]
(FrProcData *) PTR_STRUCT	next	Identifier for next FrProcData structure in the linked list.
INT_4U	chkSum	Structure checksum 6

4.3.2.12 Raw Data -- FrRawData

This is a structure containing the following data:

Table 19: Raw Data Structure Definition

FrRawData Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Name of raw data.
(FrSerData *) PTR_STRUCT	firstSer	Identifier for first serial data structure in the linked list.
(FrAdcData *) PTR_STRUCT	firstAdc	Identifier for first ADC data structure in the linked list.
(FrTable *) PTR_STRUCT	firstTable	Identifier for first table structure in the linked list.
(FrMsg *) PTR_STRUCT	logMsg	Identifier for first error message structure in the linked list.
(FrVect *) PTR_STRUCT	more	Identifier for the additional user-defined data structures.
INT_4U	chkSum	Structure checksum 6

[21] The first FrHistory should describe the processing used to build this FrProcData channel. Its name variable should be the FrProcData name. If the channel(s) used to produce this FrProcData have prior FrHistory structures, then in order to not lose this history, these structures should be copied and added to the FrHistory linked list.

4.3.2.13 Serial Data -- FrSerData

This is a structure containing the following data:

Table 20: Serial Data Structure Definition

FrSerData Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Name of station producing serial data stream.
INT_4U	timeSec	Time of data acquisition, GPS time in integer seconds since GPS standard epoch.1
INT_4U	timeNsec	Frame start time residual, integer nanoseconds.
REAL_8	sampleRate	Sample rate, samples / s.
STRING	data	Pointer to string for ASCII-based data.
(FrVect *) PTR_STRUCT	serial	Identifier for serial data vector.
(FrTable*) PTR_STRUCT	table	Identifier for the user-defined table structure.
(FrSerData *) PTR_STRUCT	next	Identifier for next serial data structure in the linked list.
INT_4U	chkSum	Structure checksum 6

4.3.2.14 Simulated Data -- FrSimData

This is a structure containing the following data:

Table 21: Simulated Data Structure Definition

FrSimData Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Name of simulated data.
STRING	comment	Comment
REAL_8	sampleRate	Simulated data sample rate, samples / s.
REAL_8	timeOffset	Offset of 1st sample relative to the frame start time (seconds). Must be positive and smaller than the frame length.[22]
REAL_8	fShift	fShift is the frequency in the original data that corresponds to 0 Hz in the heterodyned series.1
REAL_4	phase	Phase of heterodyning signal at start of dataset.1
(FrVect *) PTR_STRUCT	data	Identifier for a single vector of simulated data.
(FrVect *) PTR_STRUCT	input	Identifier for input parameters for simulation.
(FrTable *) PTR_STRUCT	table	Identifier for table data structure.

[22] Time offsets always added together (with suitable scaling of timeOffsetN) to obtain decimal time offset: time_of_1st_sample = (GtimeS + GtimeN/10^9) + timeOffset. Note that all quantities are ALWAYS positive

FrSimData Structure (continued)		
Data class	Variable Name	Descriptor & Comments
(FrSimData *) PTR_STRUCT	next	Identifier for next simulated data structure in the linked list.
INT_4U	chkSum	Structure checksum 6

4.3.2.15 Simulated Event Data -- FrSimEvent

This is a structure containing the following data:

Table 22: Simulated Event Data Structure Definition

FrSimEvent Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Name of event.
STRING	comment	Descriptor of event.
STRING	inputs	Input channels and filter parameters to event process.
INT_4U	GTimeS	GPS time in seconds corresponding to maximum of event.
INT_4U	GTimeN	GPS time in residual nanoseconds relative to GTimeS corresponding to maximum of event.
REAL_4	timeBefore	Signal duration before GTimeS
REAL_4	timeAfter	Signal duration after GTimeS
REAL_4	amplitude	Continuous output amplitude returned by event
INT_2U	nParam	Number of additional event parameters.
REAL_8[nParam]	parameters	Array of additional event parameters (size of nParam).
STRING[nParam]	parameterNames	Array of parameter names (size of nParam).
(FrVect *) PTR_STRUCT	data	Identifier for vector containing additional event results.
(FrTable *) PTR_STRUCT	table	Identifier for table structure containing additional event information.
(FrSimEvent *) PTR_STRUCT	next	Identifier for another event.
INT_4U	chkSum	Structure checksum 6

4.3.2.16 Summary Data -- FrSummary

This is a structure containing the following data:

Table 23: Summary Data Structure Definition

FrSummary Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Name of summary statistic.
STRING	comment	Comment.

FrSummary Structure (continued)		
Data class	Variable Name	Descriptor & Comments
STRING	test	Statistical test(s) used on raw data
INT_4U	GTimeS	GPS time in seconds corresponding for this FrSummary
INT_4U	GTimeN	GPS time in residual nanoseconds relative to GtimeS
(FrVect *) PTR_STRUCT	moments	Identifier for vector containing statistical descriptors
(FrTable *) PTR_STRUCT	table	Identifier for table structure containing additional summary information.
(FrSummary *) PTR_STRUCT	next	Identifier for other summary.
INT_4U	chkSum	Structure checksum 6

4.3.2.17 Table Structure -- FrTable

This is a structure containing the following data:

Table 24: Table Data Structure Definition

FrTable Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Name of this table -- not required to be unique within a frame.
STRING	comment	Comment
INT_2U	nColumn	Number of columns in table
INT_4U	nRow	Number of rows in table. The frame API must ensure that all columns have this length.
STRING[nColumn]	columnName	Names of the columns. The frame API must then copy these names into each FrVect structure included in this FrTable
(FrVect *) PTR_STRUCT	column	First column of table (may be names of rows)
(FrTable *) PTR_STRUCT	next	Next table in linked list
INT_4U	chkSum	Structure checksum 6

4.3.2.18 Table of Contents Structure -- FrTOC

This is a structure containing the following data:

Table 25: Table of Contents Data Structure Definition

FrTOC Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		

FrTOC Structure (continued)		
Data class	Variable Name	Descriptor & Comments
STRING	fileBaseName	The name of the file at the time of creation including its extension but excluding any directory components. (ex: /a/b/c.gwf becomes c.gwf) If there is no corresponding file with the stream, the empty string will be stored.
INT_4U	nFrame	Number of frames in this file
INT_4U[nFrame]	dataQuality	Array of integer QA words from each FrameH (size of nFrame)
INT_4U[nFrame]	GTimeS	Array of integer GPS frame times (size of nFrame)
INT_4U[nFrame]	GTimeN	Array of integer GPS residual nanoseconds for the frame (size of nFrame)
REAL_8[nFrame]	dt	Array of frame durations in seconds (size of nFrame)
INT_8U[nFrame]	positionH	Array of FrameH positions, in bytes, from beginning of file (size of nFrame).
For FrSH:		
INT_4U	nSH	Number of FrSH structures in the file.
INT_2U[nSH]	SHid	Array of FrSH IDs (size of nSH).
STRING[nSH]	SHname	Array of FrSH names (size of nSH).
For FrDetector:		
INT_4U	nDetector	Number of distinct types of FrDetector in the file.[23]
STRING[nDetector]	nameDetector	Array of FrDetector names (size of nDetector). They appear alphabetically.
INT_8U[nDetector]	positionDetector	Array of FrDetector positions from the beginning of file (size of nDetector). We capture only the first occurrence for each type of FrDetector.[24]
For FrAdcData:		
INT_4U	nADC	Number of unique FrAdcData names in file. If nADC == 2^{32}-1, then "no data available in FrTOC"
STRING[nADC]	nameAdc	Array of FrAdcData names (size of nADC)
INT_8U[nADC][nFrame]	positionADC	Array of lists of FrAdcData offset positions, in bytes, from beginning of file (size of nADC*nFrame) The ordering of entries:row/column follows the C convention. All positions for one ADC appear sequentially.
For FrProcData:		
INT_4U	nProc	Number of unique FrProcData names in file. If nProc == 2^{32}-1, then "no data available in FrTOC"
STRING[nProc]	nameProc	Array of FrProcData names (size of nProc)

[23] e.g., a file composed of 10 frames, each of which has data from 3 interferometers contains 30 FrDetector structures grouped into 3 types : nDetector=3.

[24] In the large majority of cases, all FrDetector structures corresponding to one detector will be copies of each other and it is sufficient to point to the first one ; this means that files containing multiple frames from the same detector operating in a different mode (e.g., rotations of ALLEGRO) will not have direct access via the FrTOC to all the FrDetector structures.

FrTOC Structure (continued)		
Data class	Variable Name	Descriptor & Comments
INT_8U[nProc][nFrame]	positionProc	Array of FrProcData positions, in bytes, from beginning of file (size of nProc*nFrame). Ordering of entries:row/column: all positions for one FrProcData appear sequentially.
For FrSimData:		
INT_4U	nSim	Number of unique FrSimData names in file. If nSim == 2^{32}-1, then "no data available in FrTOC"
STRING[nSim]	nameSim	Array of FrSimData names (size of nSim)
INT_8U[nSim][nFrame]	positionSim	Array of FrSimData positions, in bytes, from beginning of file (size of nSim*nFrame). Ordering of entries:row/column: all positions for one FrSimData appear sequentially.
For FrSerData:		
INT_4U	nSer	Number of unique FrSerData names in file. If nSer == 2^{32}-1, then "no data available in FrTOC"
STRING[nSer]	nameSer	Array of FrSerData names (size of nSer)
INT_8U[nSer][nFrame]	positionSer	Array of FrSerData positions, in bytes, from beginning of file (size of nSer*nFrame). Ordering of entries:row/column: all positions for one FrSerData appear sequentially.
For FrSummary:		
INT_4U	nSummary	Number of unique FrSummary names in file. If nSummary == 2^{32}-1, then "no data available in FrTOC"
STRING[nSummary]	nameSum	Array of FrSummary names (size of nSummary)
INT_8U[nSummary][nFrame]	positionSum	Array of FrSummary positions, in bytes, from beginning of file (size of nFrame*nSummary). Ordering of entries:row/column: all positions for one FrSummary appear sequentially.
For FrEvent:		
INT_4U	nEventType	Number of type of FrEvent in the file
STRING[nEventType]	nameEvent	Array of FrEvent names (size of nEventType) They appear alphabetically.
INT_4U[nEventType]	nEvent	Number of FrEvent for each type of FrEvent (size of nEventType) If nEvent[i] == 2^{32}-1, then "no data available in FrTOC"
INT_4U	nTotalEvent	Total number of FrEvent: nEvent[0]+nEvent[1]+...+nEvent[nEventType] (excluding the 2^{32}-1 cases)
INT_4U[nTotalEvent]	GTimeSEvent	GPS time in integer seconds (size of nTotalEvent)
INT_4U[nTotalEvent]	GTimeNEvent	Residual GPS time in integer nanoseconds (size of nTotalEvent)
REAL_4[nTotalEvent]	amplitudeEvent	Event amplitude (size of nTotalEvent)
INT_8U[nTotalEvent]	positionEvent	Array of FrEvent positions, in bytes, from beginning of file (size of nTotalEvent)
For FrSimEvent:		
INT_4U	nSimEventType	Number of type of FrSimEvent in the file

FrTOC Structure (continued)		
Data class	Variable Name	Descriptor & Comments
STRING[nSimEventType]	nameSimEvent	Array of FrSimEvent names (size of nSimEventType) They appear alphabetically.
INT_4U[nSimEventType]	nSimEvent	Number of FrSimEvent for each type of FrSimEvent (size of nSimEventType) If nSimEvent[i] == 2^{32}-1, then "no data available in FrTOC"
INT_4U	nTotalSEvent	Total number of FrEvent: nSimEvent[0]+nSimEvent[1]+...+nSimEvent[nSimEventType] (excluding the 2^{32}-1 cases)
INT_4U[nTotalSEvent]	GTimeSSim	GPS time in integer seconds (size of nTotalSEvent)
INT_4U[nTotalSEvent]	GTimeNSim	Residual GPS time in integer nanoseconds (size of nTotalSEvent)
REAL_4[nTotalSEvent]	amplitudeSimEvent	Event amplitude (size of nTotalSEvent)
INT_8U[nTotalSEvent]	positionSimEvent	Array of FrSimEvent positions, in bytes, from beginning of file (size of nTotalSEvent)
INT_4U	chkSum	Structure checksum 6

Index entries of a table of contents are ordered alphabetically by channel names. This ordering is determined by the byte value of the corresponding ASCII character set with 1st byte being the most significant (rules of the strcmp C function). This is a case sensitive order.

Frames, FrEvent and FrSimEvent are ordered according their time in the FrTOC even if they are not ordered in the file.

4.3.2.19 Vector Data -- FrVect

This is a structure containing the following data:

Table 26: Vector Data Structure Definition

FrVect Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Channel name -- not required to be unique
INT_2U	compress	Compression algorithm number[25]
INT_2U	type	Vector class[26]
INT_8U	nData	Number of sample elements in data series
INT_8U	nBytes	Number of bytes in the compressed vector
CHAR[nBytes][27]	data	nData elements of specified class
INT_4U	nDim	Dimensionality of data vector
INT_8U[nDim]	nx	dimension lengths
REAL_8[nDim]	dx	sample spacing along each coordinate;

[25] See Appendix A
[26] See Appendix B
[27] See Appendix A

FrVect Structure (continued)		
Data class	**Variable Name**	**Descriptor & Comments**
REAL_8[nDim]	startX	Origin for each data set
STRING[nDim]	unitX	scale factors in ASCII; “unit per step size along each coordinate”. If dimensionless, then unitX == <NONE>, in CAPITALS (without <...>).
STRING	unitY	String describing how to interpret the value of each element. If dimensionless, then unitY == <NONE>, in CAPITALS (without <...>).
INT_8U	nDataValid	A value of zero indicates no dataValid data is present; a non-zero value indicates dataValid contains data. • nData/nDataValid must be a whole number ≥ 1 • For multidemensional array: – Arrays are stored in row-major order (C array format) – The innermost dimension of the array divided by nData/nDataValid must be a whole number ≥ 1 • Values outside the accepted range will be interpreted as the FrVect being invalid.
INT_2U	dataValidComp-Scheme	Compression scheme
INT_8U	nDataValidComp-Bytes	Number of compressed bytes in dataValid
CHAR[nData-ValidComp-Bytes]	dataValid	• This is an array of dataValid values covering the span of the FrVect.data. A value of 0 in the array signifies the corresponding block of data is valid. A non-zero value in the array signifies data as suspect. • Non-zero values: 1 $\Rightarrow$ invalid 2 $\Rightarrow$ missing samples 3 $\Rightarrow$ out of range 255 $\Rightarrow$ undocumented error
(FrVect *) PTR_STRUCT	next	Identifier for additional data.
INT_4U	chkSum	Structure checksum 6

For multi-dimensional arrays, elements are stored by rows following the C convention.

Chapter 5

Rules for Revision

LIGO and VIRGO will jointly maintain both the definition for the Frame Format and also all associated software libraries needed to write and access the frame structures.

5.1 Frame Formats

The numbering scheme for future revisions of the frame format shall be a single number as indicated in Section 4 above. The Format Version shall be incremented when any data type or data structure described in Section 4 is added, removed, or altered relative to the previous Format Version.

5.2 Frame Library Software

The actual software design of the frame manipulation libraries is the subject of a second document. However, for completeness, the rules for revising this software are indicated here.

There is a corresponding software specification document which provides detailed information in usage and generation of frames. As the frame format evolves, corresponding changes in the software libraries will be made, and a corresponding index of Frame format version and software library version shall be maintained. Version specification for the software libraries shall be in a form A.B.

A = version number. This is incremented whenever the frame format version number is changed. A shall be the same as the frame data format version number. If A is incremented, B is rest to 0.

B = revision number. This is incremented whenever one or more of the following changes are made: (i) software error fixes; (ii) enhancements in existing functionality; (iii) modification or addition of structures not addressed in A above.

5.3 Change Control

Updates will be provided by the following basis.

a. Change requests will be reviewed jointly by VIRGO and LIGO on a regular basis.
b. Those changes which are selected for incorporation shall be assigned for implementation to respective groups.
c. All changes will be validated and verified using a prescribed test procedure.
d. Once available, the new release will be distributed via the LIGO and VIRGO web sites. All affected documentation will be revised to show changes.
e. A history of revisions shall be maintained and made available to users.

Chapter 6

Specification Revision History

After Version D, the specification was transported to a different word processing application. The revision history through the current version is shown in.

Revision	Authority	Pages Affected[28]	Item(s) Affected
A	Initial release	-	-
01	Pending release as Rev. B	14	Table 8, added row 16 (overRange).
		15	Table 9, last row, delete FrDetector* element of structure.
		19	Table 18, row 5, change variable name to type.
		20	Table 18, rows 11, 12, interchange variable names (unitY, unitX).
		20	Table 18, footnote a, change {class#...} to {type#...}. Figure 3, renumber 3rd bytes of various
		22	elements to reflect changes described above.
02	Pending release as Rev. B	5	First version of new format changed to 3
		7	Table 4, edited description of Byte 5 (row 2)
		19	Table 17, delete row 10, FrStatData* element of structure
		19	Table 18, add element class INT_2U, to flag data compression. Add footnote to describe implemented compression algorithms. This becomes footnote a on page 20.
		19	Table 18, change class of variable type to INT_2U
		19	Table 18, introduce new element, class INT_4U, variable name nBytes. Needed to support data compression. Table 18, footnote a becomes footnote b.
		19	Table 18, added footnote a; footnote a becomes
		20	footnote b.
03	Pending release as Rev. B	5	Table 2, Added time standard acronym definitions.

Revision	Authority	Pages Affected	Item(s) Affected
		6	Table 3, introduce a footnote (a) explaining that any of the data classes may be used as a list of such elements. The notation for this shall be *type, just as in C/C++. The number of elements in the list will be a piece of information contained elsewhere within the structure where this list object appears. At present it is exclusively used in the structure FrVect. By doing this, previous footnote a becomes footnote b.
		6	Table 3, correct C/C++ Data Type entry for INT_4U. int_U -> unsigned int.
		20	Table 18, exchange the Descriptor & Comments fields for the entries unitX and unitY (comments are reversed)
		20	Table 18,modify the compression types by editing footnote a.
04	Pending release as Rev. B	6	Table 2, added in footnote b the definition of the leap second.
		13	Table 7, changed variable names UTimeXX -> GTimeXX to reflect change of time basis to GPS (atomic time). Added the new variable ULeapS, giving the time offset between UTC and TAI. Added parenthetical comment to localTime comment field.
		14	Table 8, corrected typographical error: nbits->nBits.
		21	Table 19, changed variable name from UTimeXX->GTimeXX to reflect change of time basis to GPS (atomic) time.
		22	Table 21, corrected typographical errors: nframes->nFrames; nbytes->nBytes.
05	Pending release as Rev. B	5	Section 1.1, Purpose, introduced discussion on primary intent and evolution of frames with time.
		20	Table 17, added an element of class PTR_STRUCT to accommodate use of multiple detector structures within one frame. See also new footnote a to Table 17 and text in section b.13.
		24	Figure 3, corrected various errors in previous versions. Some of these reflect changes in structure definitions which have been introduced to date.
B	DCN E970066-00-E	13	Table 7, parameter dt, frame length, redefined as seconds.
06	Pending release as Rev. C	All	Miscellaneous wording changes, reordering of structure definitions by alphabetical order, etc. Created Table 5 to show the first three elements common to all structures. Added pointers in various structures to newly defined structures (see below).
		7	Section 4, Changed Frame Format Version Number from 3 to 4.

Revision	Authority	Pages Affected	Item(s) Affected
		11	Altered Figure 1 to show Table of Contents structure
		14	Altered structure of FrameH to include dataQuality; changed run number to INT_4S.
		17	Altered FrDetector to remove reference to arm length to accommodate multiple interferometers and bars.
		18	Altered FrEndOfFile to include pointer to the new FrTOC structure.
		21	Introduced FrSimEvent to capture input parameters to a simulated event.
		22	Altered FrStatData to include a data representation variable.
		23	Introduced FrTable structure to accommodate tabular data formats.
		24	Introduced FrTOC structure to capture index (table of contents) into a file of frames.
		26	Altered FrTrigData to include new parameters to capture trigger duration.
		27	Altered FrVect to include offset, "startX", for value of parameter "x".
		31	Added Appendix A to list dataQuality bit values
		32	Added Appendix B to replace two long footnotes to FrVect.
C	DCN E000023-00-E	All	All changes described for Rev. 06 have been incorporated. In addition several other miscellaneous changes and corrections were included.
07	Pending release as Rev. D	10	Section 4.2, called out explicitly that all variables are case sensitive.
		14	Table 5, changed wording in the use of instance counters to be more precise. Sectionn 4.3b, changed "frame" to 'file" in text describing uses of FrSh and FrSE.
		19	Section 4.3b.6, removed uneeded comma in text. Table 12, changed FrEndofFrame run number to INT_4S to match change made in FrHeader
		23	Table 20, changed wording in footnote to be more precise.
		24	Table 22, added pointer to next frTable structure in the definition of frTable.
		33	Table 27, modified table to include big/little endian machines and changed wording in table to be more precise. Also changed the numbering scheme by +1 for compression modes > 256. This was done to be consistent with text in the first paragraph. and to include uncompressed data for both platform types.
08	Pending release as Rev. D	16	Section 4.3, b, added paragraphs at end to discuss how attibutes of structures must appear contiguously and how structures themselves appear hierarchically.

Revision	Authority	Pages Affected	Item(s) Affected
		18	Table 7, called out uniqueness of FrSE element names.
		26	Table 20, called out version numbering convention.
		27	Table 22, called out that name of table structures need not be unique. Section 4.3b.18, specified how and what structures appear in FrTOC.
		28	Table 23, changed the following FrTOC attributes: ULeapS->INT_2U; dt->REAL_8; runs->INT_4S; position->specified in bytes; added nFirstADC, nFirstSer, nFirstTable, nFirstMsg to index these key structures; added channelID, groupID to ADC index; corrected length of FrTrig and FrSimEvt position lists.
		31	Table 25, called out that name of vector structures need not be unique.
09	Pending release as Rev. D	21	Section 4.3b.6, changed wording to reflect that FrTOC comes after all frames in a file.
		27	Table 23, changed UleapS to INT_2S. Changed positionH to refer to FrameH. Changed or added FrStatData elements (nameStat, tStart, tEnd, postionStat). Changed or added FrADCData elements (channelID, groupID, positionADC). Changed or added FrProcData elements (nameProc, positionProc). Changed or added FrSimData elements (nameSim, positionSim). Changed or added FrSerData elements (nameSer, positionSer). Changed or added FrTrgiData elements (nameTrig, position Trig). Changed or added FrSimEvt elements (nameSimEvt, positionSimEvt). Changed or added FrSummary elements (nameSum, position Sum).
10	Pending release as Rev. D	25	Table 20, added footnote noting that the combination of four elements for each FrStatData must be uniquely specified.
		27	Table 23, Made nStat variables unique. Moved location of FrSummary elements within the table.
11	Pending release as Rev. D	18	Table 8, changed variable simEvtData to simEvent
		24	Table 19, replaced word “trigger” by “event” throughout comments.
		26	Table 22, renamed linked list pointer from “table” to “next”.
		29	Table 24, changed timeBefore and timeAfter datatypes to REAL_4.

Revision	Authority	Pages Affected	Item(s) Affected
D	DCN E000550-00-E	All	All changes described for all revisions after Release C have been incorporated. Table 3, added footnote c, regarding STRING terminations. In addition several other miscellaneous changes and corrections were included.
E	DCN E020745-00-E	All	Table 4 , • change instance of PTR_STRUCT from INT_4U to INT_8U • footnote 5: specify '\0' content in string. Table 3, change release version to 5. Figure 2, update to reflect the changes. Rename FrTrigData to FrEvent everywhere it applies. Table 6: • change instance from INT_4U to INT_8U • change length from INT_4U to INT_8U Table 8 (FrameH), remove the elements "strain" and localTime. Table 9 (FrAdcData), • add phase element. • Change timeOffset to decimal. Table 10 (FrDetector): • Add localTime. • Use decimal radians for the detector location (longitude and latitude). • Add arm size (midpoint). • Add arm orientation (altitude). • Add a pointer "next" to support linked list. • Add dataQuality and qaBitList elements. Table 11(FrEndOfFile): • Change chkFlag to chkType. • Change the type of nBytes and seekTOC from INT_4U to INT_8U. Table 12(FrEndOfFrame), add the field checkType and chkSum. Table 14 (FrMsg): • Add GTimeS and GTimeN elements. • Specify the default value for severity. Table 15 (FrProcData), add type, subType, tRange, phase, fRange, BW, auxParam, history elements. Table 18 (FrSimData), add fshift and phase elements. Table 19 (FrSimEvent), add parameters elements. Table 21 (FrSummary), add GTimeS and GTimeN elements.

Revision	Authority	Pages Affected	Item(s) Affected
			Table 22 (FrTable), add the dimension in the array specification. Table 23 (FrTOC): • Allow the TOC to be located at the beginning of the file. • Add dataQuality array. • Modify the FrDetector part to support multiple detectors. • Introduce time and alphabetic order for the TOC content. • reorganize the event information (FrEvent and FrSimEvent). Table 24 (FrEvent), add parameters elements. Table 27 (Data compression): • Add compression scheme 8, 9, 264,265. • Remove unused compression scheme (3, 6, 258, 260). • Clean up the table. Table 28 (FrVect): • Change FR_VECT_C16 to FR_VECT_16C. • Add the dimension in the array specification. • Change the type of nData, nBytes and nx from INT_4U to INT_8U Figure 3 removed Appendix A: • Reserve 2 bits per detector. • Reassign the bit for the prototypes interferometers to the CIT 40m. Add Appendix D about naming convention. And numerous minor changes.
F	DCN: E020807-00-E	6	Table 3 Frame format number changed from 5 to 6 to preclude ambiguities of frame format interpretation. Format 5 has been used extensively to build intermediate prototype frames as this Specification has evolved, and in order to avoid creation of frames with the same Format number but different internal structures.
G		6	Table 3 Frame format number changed from 6 to 7
		7	Table 4, footnote 5: add that multiple contiguous '\0' characters are allowed at the end of the string.
		17, 23	Table 14 and 21 (FrEvent and FrSimEvent): change the type of the "parameters" variable from REAL_4[nParam] to REAL_8[nParam].
		21, 22	Table 19 and 20 (FrSerData and FrSimData): change the type of the "samplerate" variable from REAL_4 to REAL_8.
		39	Table 30: fix the last line to match the unsigned size of the FrVect variable "type".

<table>
<tr><th>Revision</th><th>Authority</th><th>Pages Affected</th><th>Item(s) Affected</th></tr>
<tr><td></td><td></td><td>30</td><td>Drop section 5.3 and part of the section 5.4</td></tr>
<tr><td>H</td><td></td><td></td><td>Change for the checksums:
Add checksum for each structure.
Supress the frame checksum.
Put the file checksum type in the file header.
The checksum variable is no longer included in the checksum computation
FrHeader: store the frame library used.
Rewrite the section 5.1 which describes the update of the frame format version.
Update the Virgo channel prefix in appendix D
FrTOC:
Add the nTotalEvent and nTotalSEvent variables
Update the variable class description especially to capture the multi dimensional array information.
Appendix B on data compression:
Add zero suppress for 8 bytes and for complex numbers
Reserve some ID for the libraries
Update the algorithm description</td></tr>
<tr><td>I</td><td></td><td></td><td>Updated Appendix D with detector information for KAGRA and LIGO India
Modified the assignment of data quality bits and detectors of Table 28 of Appendix A to mirror the more complete information in section 2 of Appendix D
Added Appendices to the Table of Contents
Changed CIT_40 channel prefix to be C1:</td></tr>
</table>

Revision	Authority	Pages Affected	Item(s) Affected
J			**Version 9 Changes** • Removed ULeapS from FrameH structure (Table: 10) [29] • Removed the following instruments as they have been decommissioned.[30] – Allegro – LIGO LHO 2km – Virgo Central Interferometer – Auriga – Explorer – Nautilus – Niobe • Merged Assigment of Data Quallity Bits Appendix with Naming Conventions Appendix • Corrected FrVect type footnote reverence (Table: 26)[31] • Removed the FrStatData Structure (Table: 31) [32] • Added dataValid to FrVect (Table: 26) [33] • Reduced meta data in Table Of Contents (Table: 25) [34] – Renamed *name* to *nameAdc* for consitency – Removed *ULeapS* – Removed *runs* – Removed *frame* – Removed *nFirstADC* – Removed *nFirstSer* – Removed *nFirstTable* – Removed *nFirstMsg* – Removed *channelID* – Removed *groupID* – Removed *nStatType* – Removed *nameStat* – Removed *detector* – Removed *nStatInstance* – Removed *nTotalStat* – Removed *tStart* – Removed *tEnd* – Removed *version* – Removed *positionStat*

[29] https://git.ligo.org/computing/policy/frame-specification/-/issues/2
[30] https://git.ligo.org/computing/policy/frame-specification/-/issues/1
[31] https://git.ligo.org/computing/policy/frame-specification/-/issues/17
[32] https://git.ligo.org/computing/policy/frame-specification/-/issues/16
[33] https://git.ligo.org/computing/policy/frame-specification/-/issues/18
[34] https://git.ligo.org/computing/policy/frame-specification/-/issues/19

Revision	Authority	Pages Affected	Item(s) Affected

J (Continued)			**Version 9 Changes (Continued)**
			• Added chkSumTOC to FrEndOfFile (Table: 13) [35]
			• Added MIME type to FrProcData (Table: 18) [36]
			• Added fileBaseName to FrTOC (Table: 25) [37]
			• Added dataQualityOffset to FrDetector) [38]
			  – Updated FrDetector structure (Table: 12)
			  – Updated detector names appendix to reflect user defined detectors.
			  – Created appendix containing known detector information for static detectors (Table: 30)
			• Added Zstandard to Data Compression Schemes (Table: A) [39]

K			**Version 9.1.0 Changes** • Change data type of nDataValid of FrVect from INT_4U to INT_8U (Table: 26) [40] • Removed Blocking size from the header of dataValid field of the FrVect structure (Table: 26) [41] • Removed localTime from FrDetector (Table: 12) [42]

[35] https://git.ligo.org/computing/policy/frame-specification/-/issues/4
[36] https://git.ligo.org/computing/policy/frame-specification/-/issues/22
[37] https://git.ligo.org/computing/policy/frame-specification/-/issues/23
[38] https://git.ligo.org/computing/policy/frame-specification/-/issues/3
[39] https://git.ligo.org/computing/policy/frame-specification/-/issues/21
[40] https://git.ligo.org/computing/policy/frame-specification/-/issues/27
[41] https://git.ligo.org/computing/policy/frame-specification/-/issues/28
[42] https://git.ligo.org/computing/policy/frame-specification/-/issues/29

Appendix A

Data Compression Schemes

The following are supported compression algorithms. Each compression scheme has two entries to accommodate the possibility that the reading machine and the writing machine have different 'endian-ness'. Big-endian machines typically refer to SUN and Apple(G3/G4) processors and little-endian machines typically refer to Intel, Intel clones and Alpha processors.

Table 28: Compression Schemes Supported

Compression Schemes Supported		
ID on bigendian Writing Platform	**ID on littleendian Writing Platform**	**Compression Description**
0x0000	0x8000	Uncompressed raw values
0x0001	0x8001	Zero suppression algorithm for 2, 4 or 8 bytes words (integer or float and complex floating point numbers)
0x0002	0x8002	Gzip (any data type, including complex)
0x0004	0x8004	Gzip, differential values (for 1, 2 or 4 bytes signed and unsigned integers)
0x0008	0x8008	Zstandard compression (any data type, including complex)
0x0010	0x8010	Zstandard, differential values (for 1, 2 or 4 bytes signed and unsigned integers)

Each compression scheme uses a single bit. This allows the frame libraries API to use bit-masks when writing data to tell which compression schemes should be tried. The compression scheme resulting in the smallest memory footprint will be chosen and only the bit value corrisponding to the compression scheme will be written. Since only the selected compression scheme is recorded (not the bit-mask) on a vector by vector basis, there is no confusion when reading a compressed frame.

For instance:

`0x0003` ask to try the zero suppression and plain gzip compression schemes.
`0x0004` ask to try only the gzip, differential values compression scheme.

`0x0019` ask to try the zero suppression and all Zstandard compression schemes.
`0x7FFF` ask to try all compression schemes.
`0x0000` ask to uncompress the data

Remarks for gzip:

For gzip the compression level is the default level as defined by the implementing library.

Remarks for Zstandard:

For Zstandard the compression level is determined by the implementing library. Additional information regarding the Zstandard algorithm, including implementation libraries for a variety of languages, can be found at https://facebook.github.io/zstd/

Zero suppression algorithm:

The zero suppression algorithm codes the data in the following way:

a. Data are differentiate using the integer subtraction.
b. A block of size (nW) is selected and is written as unsigned short (2 bytes)
c. The input data are split in blocks
d. For each block the minimal number of bits (nB) per word which contain non zero data is determined and (nB-1) is written using 3 bits for one byte words (char), 4 for two bytes words (short), 5 for four bytes words (int or float), 6 for eigth bytes words(long or double)
e. Each word in the bloc is transform to unsigned by adding 2**(nB-1)-1 and the useful nB bits are added to the output buffer.
f. Then the next bloc is processed until completion of the input buffer.

Remarks:

- The zero suppress algorithm for floating point numbers (4 or 8 bytes) treats the data as if they were integers with the same number of bytes.
- Complex vectors are first reorganized in real part only followed by the imaginary part before handling this vector as plain integers like for the regular floating point numbers
- The bit ordering of the "output buffer" is the byte ordering of the machine doing the compression and is recorded by the compression scheme ID.

Example with block size = 3

- Input vector: 82 85 85 81 80 82 84 85 (short)
- Differentiate data: 82 3 0 –4 –1 2 2 1
- Block 1: (83 –3 –4) nBits = 8 → write 8-1=7
  - 82 = 0x52 + 7f = 0xd1
  - 3 = 0x03 + 7f = 0x82
  - 0 = 0x0 + 7f = 0x7f
- Block 2 (-4 –1 2) nBits = 4 → write 4-1 = 3
- Block 3 (2 1 0) nBits = 3 → write 3-1 = 2

- Output is (hex): 0x0003 2D17 37f8 2963 0025

Appendix B

FrVect Data Types

The following valid vector data types are defined: {type#: name: description};

Table 29: FrVect Data Types

ID	Data Type Name	Data Type
0	FR_VECT_C	CHAR
1	FR_VECT_2S	INT_2S
2	FR_VECT_8R	REAL_8
3	FR_VECT__4R	REAL_4
4	FR_VECT_4S	INT_4S
5	FR_VECT_8S	INT_8S
6	FR_VECT_8C	COMPLEX_8
7	FR_VECT_16C	COMPLEX_16
8	FR_VECT_STRING	STRING
9	FR_VECT_2U	INT_2U
10	FR_VECT_4U	INT_4U
11	FR_VECT_8U	INT_8U
12	FR_VECT_1U	CHAR_U
13 - 65535	unimplemented	

Appendix C

Naming Conventions

For Frames intended to be shared in a multi-detector analysis, the following guidelines must be applied:

C.1 Channel names

Channels derived from simulations or instrumental outputs should contain within their names an identification of the instrument with which they are associated. For example, in the case of single channel raw (ADC) data, the scheme that should be used is:

Xn:Name

where

X is a single letter describing the site (H for Hanford, L for Livingston, V for Virgo. . .)

n is the detector number (0 is reserved for environment monitoring)

Name is the channel name which usually contain a location and a signal type.

This naming scheme should be used for:FrAdcData, FrSimData, FrProcData, FrSummary, FrMsg, FrHistory, and FrSerData.

If the channel is the result of a pipeline process or a combination of channels (e.g., correlation spectra stored in FrProcData), then this prefix is not mandatory. However, the string of three characters “Xn:” must still appear uniquely somewhere within channel name in order to be able to trace the channel origin and to parse the channel name to pick out this information.

C.2 Detector names

(FrDetector name and in derived objects like detector response tensors)

A name string should be associated with a unique set of geometrical parameters. This means that detectors (such as ALLEGRO) which can be reoriented should use different variations of their names when storing data corresponding to different orientations, e.g., ”ALLEGRO_45” (azimuth 45 degrees), ”ALLEGRO_63” (azimuth 63 degrees), etc.

The following detector names (FrDetector) have reserved values for all fields of the table. These detectors shoud be considered static. Being static, the FrDetector structure is optional. Implimentation libraries should provide a static representation of the detector’s geometrical data.

Detector	Name in FrDetector	Channel prefix	Bit in data quality word	Bit
TAMA 300	TAMA_300	T1:	0,1	2^0
Virgo 3 km	Virgo	V1:	4,5	2^4
GEO 600	GEO_600	G1:	6,7	2^6
LIGO LHO 4 km	LHO_4k	H1:	10,11	2^{10}
LIGO LLO 4 km	LLO_4k	L1:	12,13	2^{12}
Caltech 40 meters	CIT_40	C1:	14,15	2^{14}
ACIGA	ACIGA	U1:	26,27	2^{26}
KAGRA	KAGRA	K1:	28,29	2^{28}
LIGO India	LIGO_India	A1:	30,31	2^{30}

The following table gives information for for dynamically allocated detectors. The detector name inside of the FrDetector structure along with the channel prefix are determined by the author of the frames. The detector name and channel prefix cannot have a value given in the reserved detector table.

Detector	Bit in data quality word	Bit
Dynamic_01	2,3	2^2
Dynamic_02	8,9	2^8
Dynamic_03	16,17	2^{16}
Dynamic_04	18,19	2^{18}
Dynamic_05	20,21	2^{20}
Dynamic_06	22,23	2^{22}
Dynamic_07	24,25	2^{24}

C.3 Frame File names

The recommended frame file extension is ".gwf". The suggested file name convention is available at https://dcc.ligo.org/DocDB/0026/T010150/000/T010150-00.pdf

Appendix D

FrDetector information for static FrDetector Structures

Name / Prefix	Longitude / Latitude	Elevation	ArmXazimuth / ArmYazimuth	ArmXaltitude / ArmYaltitude	ArmXmidpoint / ArmYmidpoint	LocalTime
ACIGA	0·0	0.0	0·0	0·0	0·0	0
U1:	0·0		0·0	0·0	0·0	
CIT_40	0·0	0.0	0·0	0·0	0·0	0
C1:	0·0		0·0	0·0	0·0	
GEO_600[43]	0·17116780435	114.425	1·19360100484	0·00000000000	300·00000000000	0
G1:	0·91184982752		5·83039279401	0·00000000000	300·00000000000	
KAGRA[44]	2·396441015	414.181	1·054113	0·0031414	1513·2535	0
K1:	0·6355068497		−0·5166798	−0·0036270	1511·611	
LHO_4k[45]	−2·08406	142.554	5·65488	−0·0006195	1997·54	0
H1:	0·810795		4·084808	0·0000125	1997·52	
LIGO_India	0·0	0.0	0·0	0·0	0·0	0
A1:	0·0		0·0	0·0	0·0	
LLO_4k[46]	−1·58431	-6.574	4·40318	−0·0003121	1997·57	0
L1:	0·533423		2·83238	−0·0006107	1997·57	
TAMA_300	2·43536359469	90.0	4·71238898038	0·00000000000	150·00000000000	0
T1:	0·62267336022		3·14159265359	0·00000000000	150·00000000000	
Virgo[47]	−10·5045	51.884	−0·3229	0·0	1500·0	3600
V1:	43·6316		4·3895	0·0	1500·0	

Table 30: Detector description information

[43] http://www.geo600.uni-hannover.de/geo600/project/location.html
[44] http://gwdoc.icrr.u-tokyo.ac.jp/cgi-bin/DocDB/ShowDocument?docid=3824
[45] https://dcc.ligo.org/DocDB/0072/P000006/000/P000006-D.pdf
[46] https://dcc.ligo.org/DocDB/0072/P000006/000/P000006-D.pdf
[47] https://tds.virgo-gw.eu/?call_file=VIR-065A-08.doc

Appendix E

Legacy Data Structures

Data structures described here existed in previous versions of the Frame Specification but are not part of the current Frame Specification. It is optional for frame i/o libraries to implement features to read these data structures when present in older frame files.

E.1 Removed from version 9

The following data structures were removed from version 9 of the Frame Specification.

E.1.0.1 Static Data -- FrStatData

This is a structure containing the following data:

Table 31: Static Data Structure Definition

FrStatData Structure		
Data class	Variable Name	Descriptor & Comments
First elements are as shown in table 7		
STRING	name	Static data name[48]
STRING	comment	Comment
STRING	representation	Type of static data being represented. e.g., calibration, swept sine, pole-zero, FIR or IIR coefficients...
INT_4U	timeStart	Start time of static data validity, GPS time in integer seconds since GPS standard epoch.1

[48]Note that the combined list of FrStatData elements {name,timeStart, timeEnd,version} must be unique. To access FrStatData of a given name for a given epoch, one selects the block(s) with the desired name and with a time range spanning the epoch of interest, one then selects that block with the latest timeStart, and finally one takes the FrStatData with the highest version number.

FrStatData Structure (continued)		
Data class	Variable Name	Descriptor & Comments
INT_4U	timeEnd	End time of static data validity (if unknown, set to 0), GPS time in integer seconds since GPS standard epoch.1
INT_4U	version	Version number for this static structure. i.e, the counter begins at 0 and is incremented by 1 thereafter. Updated statics for the same time window (e.g., modified calibration data) will be identified by unique version numbers.
(FrDetector *) PTR_STRUCT	detector	Identifier for the detector this static data is associated with.
(FrVect *) PTR_STRUCT	data	Identifier for vector of data.
(FrTable *) PTR_STRUCT	table	Identifier for first table structure in the linked list.
INT_4U	chkSum	Structure checksum 6

It is possible for a frame to contain more than one structure requiring different FrStatData structures. For example FrDetector for raw data may require static data associated with instrumental parameters; FrProcData for processed data may require static data associated with the filtering or other analysis parameters; FrSimData for simulated data may require static data to define precisely the input parameters to the simulation. For this reason, there is included in the FrStatData definition a PTR_STRUCT object which provides information on the antecedent detector structure with which any one FrStatData structure is associated. Footnote 1, Table 20, applies to each detector structure separately.