VLBI Data Interchange Format (VDIF) Specification

Release 1.0
6 March 2009
(subject to final approval)

1. Introduction

The VLBI Standard Interface (VSI) specifications, developed in the early 2000's and designated VSI-H and VSI-S[1] and aimed primarily at recording and playback systems, specify standards for a hardware/electrical VLBI data interface and a software control interface, respectively. These VSI specifications intentionally do not address the format of the transported data.

In recent years, a number of new VLBI data-acquisition and capture systems have appeared, along with increasing need to interchange data on a global scale, including real-time and near-real-time transfer via high-speed network, as well as by standard disk-file transfer. These types of data transfers have been increasingly plagued by the lack of an internationally agreed data format, often requiring *ad hoc* format conversions that require both programming effort and computing/storage resources. Recognizing this problem, a so-called VSI-E ('E' for 'e-VLBI') specification, based on standard RTP/RTCP network protocol, was first proposed and implemented in 2003-2004, which specified both data formats and data-transport mechanisms for real-time e-VLBI data transfer. Though VSI-E was comprehensive, it was never formally ratified by the larger VLBI community. Its adoption was further hampered by its complexity, and it has been largely abandoned.

The VLBI Data Interface Specification (VDIF) has a somewhat different goal from VSI-E, specifying only a standardized transport-independent VLBI data-interchange format that is suitable for all types of VLBI data transfer, including real-time and near-real-time e-VLBI, as well as disk-file storage. The VDIF specification, unlike VSI-E, explicitly makes no attempt to define an on-the-wire data-transport protocol, which is expected to be the subject of a subsequent specification document. The combination of VDIF, along with this follow-on data-transport-protocol specification will, when completed, essentially constitute a replacement for VSI-E. And though the VDIF specification makes no mention of data-transport protocol, it has been developed with an awareness of expected methods of data transport, including network transport using various standard protocols, as well as physical or electronic transport of standard disk files.

2. VDIF Task Force

The 2008 International e-VLBI Workshop, held 14-17 June 2008 in Shanghai, China, included panel and group discussions specifically targeting the subject of creation an international data-format standard. Those discussions led to the creation of a small, broadly-based international task force (subsequently known as the VDIF Task Force) to study the problem and make recommendations to the larger VLBI community. This document is the result of the extensive deliberations and discussions of the VDIF Task Force, mostly via e-mail, as well as solicitation of comments and suggestions from key members of the broader VLBI community, and represents our best effort to answer the challenge presented to us.

[1] VSI-H and VSI-S specifications are available at http://www.haystack.mit.edu/tech/vlbi/vsi/index.html

3. Basic VDIF structure

The discussions at the Shanghai meeting supported the concept of a 'framed' data-stream format consisting of stream of "Data Frames", each containing a short self-identifying Data Frame Header, followed by a Data Array (containing the actual samples), as shown in Figure 1. A similar format is already used by several current and proposed disk-based recording systems.

Accordingly, the VDIF specification is based upon a basic self-identifying Data Frame, which carries a time segment of time-sampled data from one or more frequency sub-bands. The length of a Data Frame may be chosen by the user to best match the chosen transport protocol; for example, in the case of real-time network transfer, a VDIF Data Frame length would normally be chosen so that exactly one Data Frame is carried by each on-the-wire packet. It is important to emphasize that the VDIF Data Frame is fundamentally transport-protocol independent, so that exactly the same set of Data Frames can represent VLBI data through a network transfer or be stored to a physical disk file.[2]

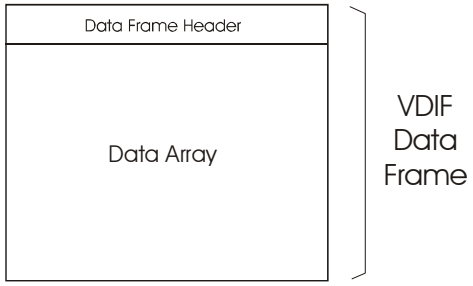

Figure 1: VDIF Data Frame structure showing Data Frame Header and Data Array

In some cases, an entire set of sampled frequency sub-bands (or 'channels') may be carried in each Data Frame. In other cases, a single Data Frame may carry data from only a single data sub-bands (channel) from among a set of many, in which case a logically parallel set of Data Frames is needed to represent the entire data set. In the VDIF concept, each time-series of Data Frames from the same set of sub-bands(s) is known as a 'Data Thread', where each of the Data Frames within the Data Thread is identified by a 'Thread ID' embedded in the Data Frame Header. For actual transmission over a serial-data network, or for storage on a disk file, the set of Data Threads that comprise the data set are merged into a single serial 'Data Stream'. Figure 2 show a schematic example of a Data Stream comprised of three Data Threads. The collection of Data Threads from the beginning to end of a particular observation, typically lasting seconds to minutes, is known as a Data Segment.

In normal usage, it is expected that two types of Data Streams will predominate: 1) a Data Stream consisting of a single Data Thread carrying multi-channel Data Frames or 2) a Data Stream consisting of multiple single-channel Data Threads, though mixing of single-channel and multi-channel Data Threads is not prohibited.

[2] A VDIF-compliant disk file consists simply of a serial stream of VDIF Data Frames. A real-time VDIF data transfer, on the other hand, normally consists of a serial stream of network data packets, each containing a single VDIF Data Frame surrounded by various layers of transport protocol (TCP, UDP, IP, etc) information. A stream of such network-transported VDIF Data Frames may be recorded directly to disk to create a valid VDIF-compliant data file. However, due to network packet-length restrictions, the reverse is not always true (i.e. a VDIF disk file could, for example, have valid Data Frame lengths much longer than can be supported in a single network packet), and the disk data would need to be "re-framed" to a different Data Frame length before network transmission using one packet per VDIF Data Frame. Normally, however, network transfer of a VDIF disk file would be done using ftp or similar file-transfer protocol that is independent of the VDIF specification.

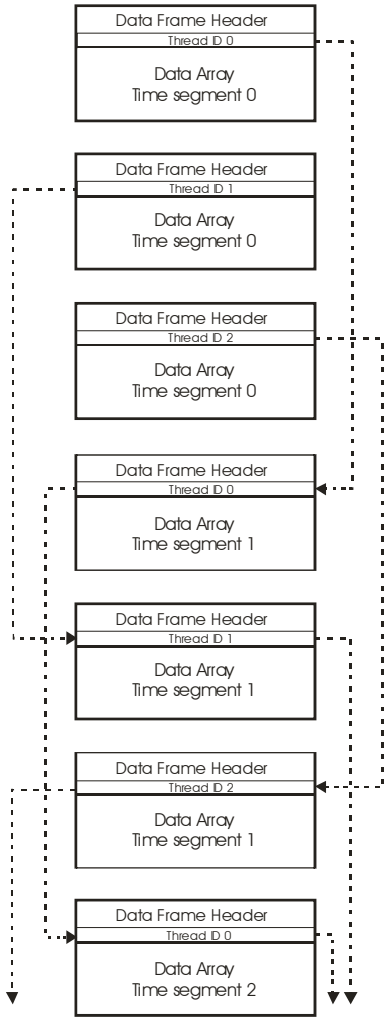

Figure 2: Illustration of Data Threads within a Data Stream

4. VDIF attributes

The following considerations guided the creation of the VDIF specification:

1. The data in each Data Stream must be decodable using only information embedded within its constituent Data Frames, including a Stream ID, a time tag for the Data Array, the number of sampled channels, and the bits/sample. (This information should be sufficient to permit computation of spectra and state-statistics information without reference to any external information.)
2. A Data Thread may be discontinuous in time at the resolution of a Data Frame (e.g. transmit/capture Data Frames only during active part of a pulsar period)
3. Each Data Frame may carry single-bit or multiple-bit samples up to 32 bits/sample.
4. Up to maximum of 1024 Data Threads, each with a unique Thread ID, may be included in a single Data Stream.

5. A minimum of data manipulation should be necessary to move data between various data-transmission techniques (e.g. disk file or real-time transfer).
6. Data rates up to at least ~100 Gbps should be supported.
7. The data overhead (e.g. embedded auxiliary information required to meet the VDIF requirements) must be as low as practical.
8. Observations over leap seconds and year boundaries must be transparently supported.
9. The VDIF data format must be compatible, in as natural way as possible, with all expected data-transport methods (e.g. network transfer, file transfer, etc).
10. Some limited amount of auxiliary user-defined data should be allowed in the Data Stream.
11. Within certain defined limits, out-of-time-order data within a Data Thread should be accommodated.

5. Data Frame Rules

The following rules govern each VDIF Data Frame within a given Data Thread:

1. Each Data Frame contains a Data Frame Header followed by a Data Array.
2. All Data Frames must have the same Data Frame Header length, Data Array length, #channels, #bit/sample and Station ID.
3. If a Data Frame contains data from multiple channels, the same time-tag must apply across channels.
4. If a Data Frame contains multiple channels, all channels must be sampled with the same number of bits/sample.
5. Each Data Array contains sample data from one or more channels with format (Section 9) and encoding (Section 10) specified by VDIF.
6. The Data Frame length (including Data Frame header) for each Data Thread must meet the following criteria:
   a. Must be a multiple of 8 bytes (for maximum compatibility with various computer-memory-address schemes and disk-addressing algorithms).
   b. Must be chosen so that an integer number of complete Data Frames are created in a continuous data flow of exactly one-second duration.
7. Data Frame #0 of each one-second period must contain, as its first sample, the data taken on a second tick of UTC; note that, in the case of time-discontinuous data, Data Frame #0 may not always be present.

Notes

These rules are intended to cover both 'on-the-wire' e-VLBI data formats as well as disk-file formats. For 'on-the-wire' real-time e-VLBI, it is expected that each transmitted non-fragmented packet will contain a single VDIF Data Frame as its data payload, in which case the Data Frame length is normally restricted to the range ~64-9000 bytes[3]. These restrictions do not apply to disk-file data format, for which the Data Frame length is limited (by the number of bits available to specify the Data Frame length) to 2^{27} bytes (~134 MBytes).

[3] In some cases, longer logical packets may be transmitted (UDP, for example, supports packet lengths up to 65527 bytes), but these packets are fragmented at the Ethernet layer and may not be suitable for some types of VLBI usage.

6. Data Frame Header

The standard 32-byte VDIF Data Frame Header is shown in Figure 3.

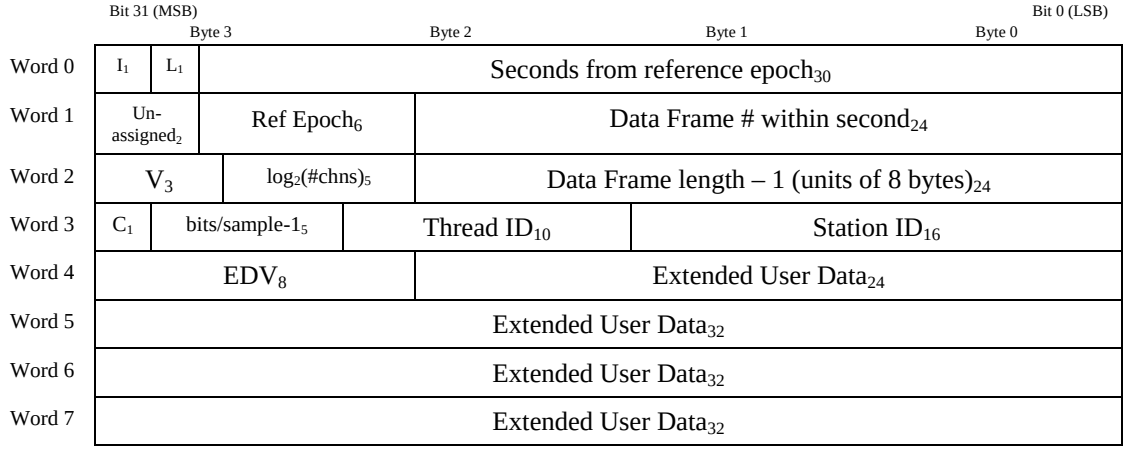

Figure 3: VDIF Data Frame Header format; subscripts are field lengths in bits; byte #s indicate relative byte address within 32-bit word (little endian format)

The words within the Data Frame Header are assigned as follows:

Word 0

Bit 31: Invalid data (i.e. data in this Data Frame has been tagged Invalid by the data source)

Bit 30: Legacy mode; see Note 1

'0' - standard 32-byte VDIF Data Frame header
'1' – legacy header-length' mode; Words 4-7 omitted from header

Bits 29-0: Seconds from reference epoch; see Note 2

Word 1

Bits 31-30: Unassigned (should be set to all '0's)

Bits 29-24: Reference Epoch for second count; see Note 2

Bits 23-0: Data Frame # within second, starting at zero; must be integral number of Data Frames per second

Word 2

Bits 31-29: VDIF version number; see Note 3

Bits 28-24: $\log_2$(#channels in Data Array); #chans must be power of 2; see Note 4

Bits 23-0: Data Frame length-1 (including header) in units of 8 bytes; see Note 5

Word 3

Bit 31: Data type; see Note 6

'0' – Real data
'1' – Complex data

Bits 30-26: #bits/sample-1 (32 bits/sample max); see Note 7

Bits 25-16: Thread ID (0 to 1023)

Bits 15-0: Station ID; see Note 8

Words 4-7

Extended User Data: Format and interpretation of extended user data is indicated by the value of Extended Data Version (EDV) in Word 4 Bits 31-24; see Note 9

Notes

1. For purposes of easing the transition from legacy VLBI disk-based data system to the VDIF standard, a 'legacy header-length mode' is supported, as specified in Word 3 bit 31. When this mode is active, Words 4-7 are omitted so that header length is reduced to 16 bytes, which is compatible with existing Mark 5B/K5/LBADR disk-based data systems.
2. The VDIF time code in Word 0 is divided into two fields:
   a. Reference epoch: A 6-bit field in Word 1 that contains the 6-month period in which the VDIF clock was *set*, with an origin of at 00UTC 1 Jan 2000, such that '0' corresponds to the first 6 months of 2000, '1' corresponds to the 6 months starting 00H 1 July 2000, etc. After setting, this 6-bit field is static and is *not incremented*. This field rolls over to 0 again when the reference epoch corresponds to the 6-month period starting 00H 1 Jan 2032.
   b. Seconds from reference epoch: A 30-bit field in Word 0 that is initially set to second count within 6-month period [i.e. (day number within a 6-month period, starting at zero)*86400+(UTC second number within day)], and thereafter counts all seconds (including any leap seconds). This field counts seconds unambiguously for 34 years, then rolls-over back to 0 after reaching a count of 2^{30}-1.

   Note that the VDIF time-code format naturally supports observations through leap seconds and over year boundaries. However, the correlator must be aware of leap seconds which occur following the Reference Epoch. No knowledge of future leap seconds is required by the VDIF clock.

   Some users may prefer to *always* fix the Reference Epoch to some particular value (such as 00UTC 1 Jan 2000). This is acceptable provided that all leap seconds are accounted for between the chosen Reference Epoch and the initial setting of the 'seconds from reference epoch' value.
3. The VDIF version number in Word 3 supports up to seven future VDIF frame-header formats to be defined, allowing decoding software to automatically determine and appropriately parse the corresponding Data Frame Header.
4. Theoretical maximum number of channels is 2^{31} = 2,147,483,648, but in practice is probably much smaller (perhaps 2^{16} = 65536). For e-VLBI, where normally one full Data Frame is transmitted in each network data packet, the number of channels may be further limited by the maximum supported length of a network data packet.
5. The Data Frame length includes the Data Frame Header and must be a multiple of 8 bytes, with a maximum length of 2^{27} bytes.
6. Each complex sample consists of two sample components, designated 'I' (In-phase) and 'Q' (Quadrature), each containing the same number of bits.
7. If the data type is 'complex', this parameter is set according to the #bits in each complex-sample component (i.e. half the total #bits per complex sample).
8. The 16-bit 'Station ID' field will accommodate the standard globally assigned 2-character ASCII ID. Or, if the number of stations is very large (SKA, for example), the Station ID may be numeric; in this case, the 16-bit field will be interpreted as an unsigned 16-bit integer. The two cases can be distinguished by the value of the first 8-bit character of the field; if this character has ASCII value $<48_{10}$ (representing the zero character '0'), a numeric ID is assumed.
9. Extended Data Words 4-7 are available for user-generated data. Each different usage of the extended words should be assigned a different Extended Data Version (EDV) number in Word 4 Bits 31-26 so that decoding software can automatically apply the proper decoding

algorithm. EDV version numbers will be coordinated through the VDIF Task Force. Up to 255 different such versions can be accommodated; if Words 4-7 are unused, the value of EDV is set to '0' and the Extended User Data fields should be set to all '0's. Each VDIF version may have an independent set of EDV numbers.

Perhaps it is worth noting that, within the first 16-bytes of the header (Word 0 through Word 3), only Words 0 and 1 contain data that change with time within a given Data Thread; data in Words 2 and 3 are static within the Data Thread.

7. Byte ordering

Byte ordering is little-endian (Intel x86 order), which is consistent with most existing disk-based systems and software-based correlators.

8. Data Frame ordering

Data coming from a single data source (e.g. a single dBBC or DBE) will normally be transmitted in strict time order. If directly connected to a local recording device, the recorded data will almost certainly be recorded in exactly the same order. However, Data Frames transmitted through a switch or over a network are not guaranteed to arrive in order.

The VDIF specification does not mandate strict Data Frame ordering within a Data Thread, but a best effort should be made to so. Some correlation equipment, particularly older types, may be sensitive to Data Frame order, in which case the requirements of Data Frame ordering will be dictated by the correlation equipment. Modern software correlators are generally rather tolerant of minor Data Frame re-ordering of the type that might occur.

9. Data Array formats

VDIF specifies the format of a Data Array based solely on the #channels and #bits/sample specified in the corresponding Data Frame Header. Adherence to these specified formats is necessary to ensure that the data are properly interpreted.

9.1 Single-channel real-data Data Array Format

The following rules apply to the composition of a Data Array carrying a single channel real data:

1. Each Data Array is composed of an even number of 32-bit words.
2. Each time sample is a single value of 1 to 32 bits.
3. The oldest data sample in each word occupies the field including Bit 0, with newer samples filling in adjacent higher-order bits until no more space is available for a full sample.
4. For multi-bit samples, the LSB of each sample occupies the most-LSB bit of the sample field.
5. A data sample may not cross a word boundary, but instead must be placed in next word; this may result in one or more unused high-order bits.
6. The first sample in a Data Array corresponds precisely to the time indicated in the Data Frame Header.

For Data Arrays carrying single-channel data, the Data Array formats for 1, 2, 3, 4, 20 and 32 bits/sample are shown in Figures 4 through 9; other values of bits/sample are formatted similarly, but are not illustrated here.

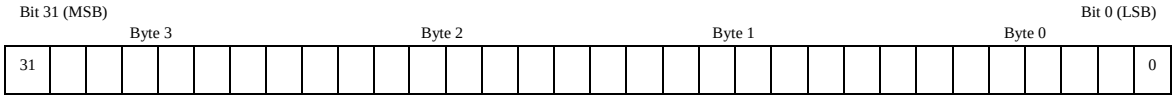

Figure 4: Real 1-bit/sample data-word format (numbers in boxes indicate relative sample #'s in time order); byte #s indicate relative byte address within 32-bit word (little endian format).

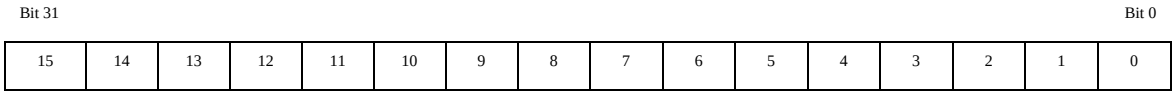

Figure 5: Real 2 bits/sample data-word format

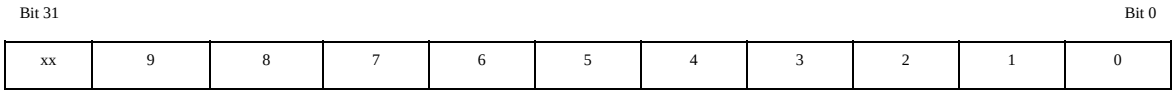

Figure 6: Real 3 bits/sample data-word format. 'xx' indicates unused bits (set to 0)

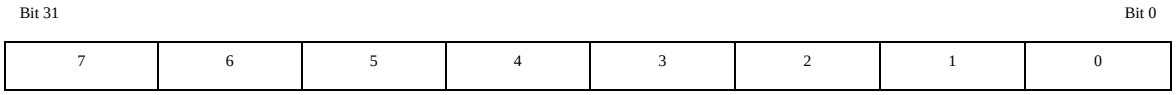

Figure 7: Real 4 bits/sample data-word format

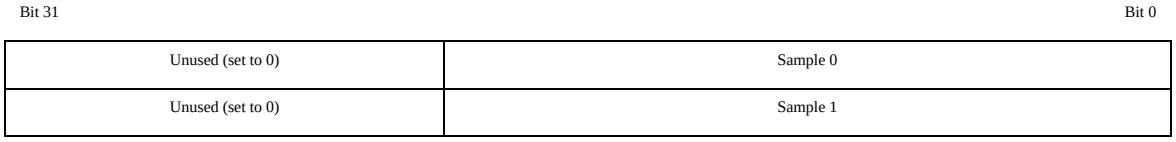

Figure 8: Real 20 bits/sample data-word format

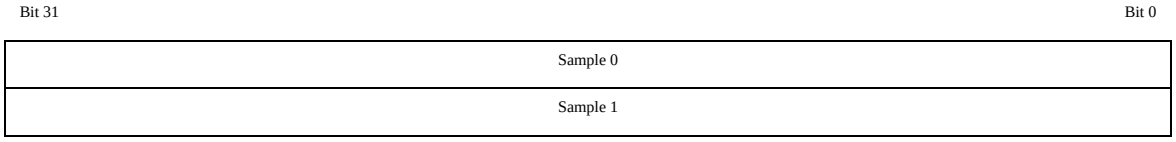

Figure 9: Real 32 bits/sample data-word format

This general scheme can be extended to an arbitrary number of bits/sample (up to 32), but a subtlety arises which must be recognized: For standard VLBI use, the number of samples/word must divide evenly into 2^n*10^6 samples/sec, which can never be satisfied for 5, 9 or 10 bits/sample. So the only legal bits/sample values in this regard are 1-4, 6-8 and 11-32 (the VDIF specification supports a maximum of 32 bits/sample). Interestingly, of all of these possibilities, only 3-bit and 6-bit samples have any storage-efficiency improvements over padding the bits/sample to the next power of 2.

9.2 Single-channel complex-data Data Array Format

The Data Array format for complex data is similar to real data, except that each complex sample consists of two scalar components, usually denoted as 'I' (for In-Phase) and 'Q' (for Quadrature), and each with the same number of bits. These two complex components are always treated as a pair and placed adjacent to each other as shown in Figure 10 through Figure 15, with the 'I' component always occupying the lower-order bits (or the first of two words when each sample component contains more than 16 bits). Note that the formatting is exactly the same as real data except that the two complex-component values occur in pairs.

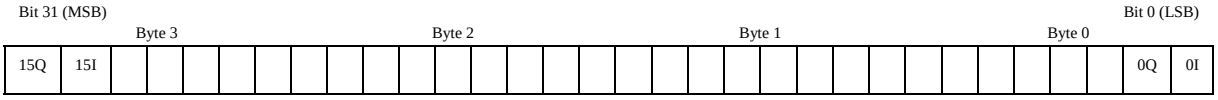

Figure 10: Complex 1-bit/sample-component data-word format; 'I' and 'Q' represent 'In-phase' and 'Quadrature' components; numbers in boxes indicate relative sample #'s in time order; byte #s indicate relative byte address within 32-bit word (little endian format)

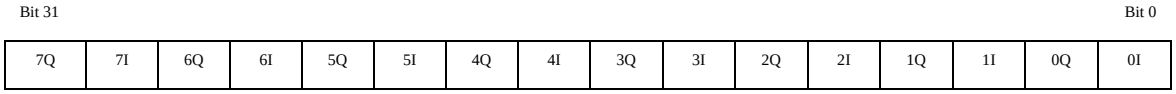

Figure 11: Complex 2 bits/sample-component data-word format

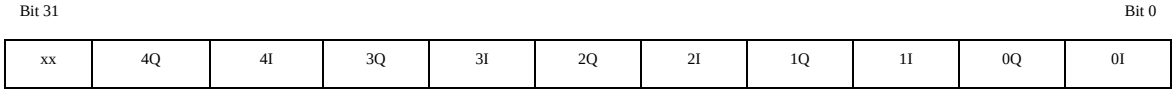

Figure 12: Complex 3 bits/sample-component data-word format. 'xx' indicates unused bits

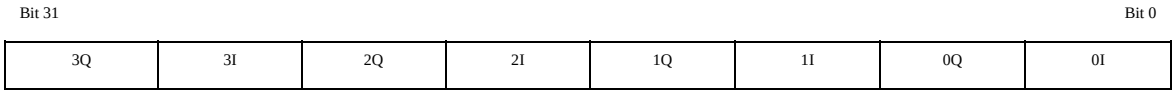

Figure 13: Complex 4 bits/sample-component data-word format

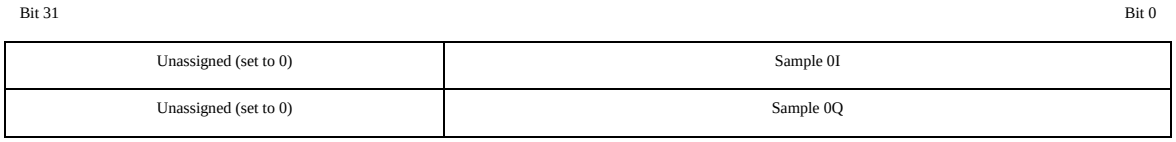

Figure 14: Complex 20 bits/sample-component data-word format

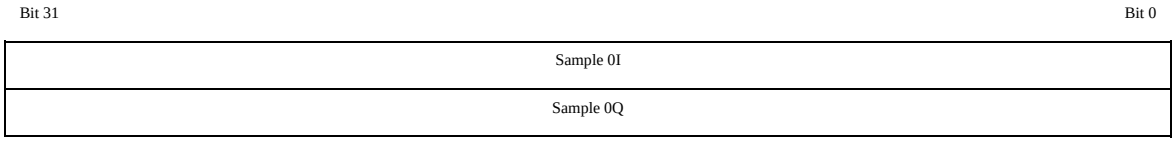

Figure 15: Complex 32 bits/sample-component data-word format

Note that, for complex data, the bits/sample parameter in the VDIF header refers to the number of bits in each complex *sample component*, not the total number of bits in the complex sample; this is required to support the case where each complex-sample component contains more than 16 bits since the Data Frame Header only supports a 5-bit binary field for specifying the sample length. Except for the case of >16 bits/sample-component, each 32-bit Data Array word always contains an integral number of complex samples regardless of the number of bits per sample. For the case of >=16 bits/sample-component, each complex sample occupies two adjacent words (see Figure 14 and 15).

9.3 Multi-channel real-data Data Array format

For simplicity, and in accordance with historical VLBI practice, the VDIF specification for multi-channel Data Arrays supports only 2^n channels with 2^k bits/sample; maximum #channels is 2^{31}=2,147,483,648 and maximum bits/sample is 2^5=32. Extension of these formats to include an arbitrary number of bits/sample is not contemplated. In such cases, users are strongly encouraged to use single-channel Data Threads, which do not impose this constraint.

For purposes of defining the multi-channel Data Array format, we define the term "complete sample". A complete sample is the sample data from all channels for a single sample time, consisting of 2^n*2^k bits.

Three rules govern the filling of 32-bit data words into a multi-channel Data Array:

1. Each individual-channel sample within a complete sample is represented by an adjacent set of 2^k bits, with the 2^0 sample bit occupying the most LSB bit.
2. The resulting individual-channel samples within a single complete sample are packed into 2^n adjacent clusters of 2^k bits each, creating a formatted field of 2^n*2^k bits. Figure 16 through Figure 19 show 32-bit Data Array word usage for several example complete-sample lengths; other cases may be inferred.
3. Each Data Array must contain an integer number of complete samples (i.e. a complete sample may not span a Data Array boundary).

Legacy disk-based systems such as the Mark 5B, K5 and LBADR systems conform to these rules, but are limited to cases where $2^n<=32$ (i.e. <=32 channels), $2^k<=2$ (i.e. 1 or 2 bits/sample), and $2^n*2^k<=32$ (i.e. complete-sample length is limited to 32 bits).

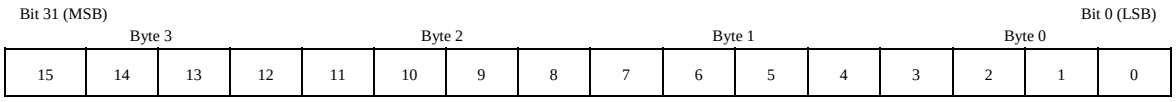

Figure 16: Data Array word for complete-sample length of 2 bits (i.e. $2^n*2^k=2$);
relative sample times are indicated in each 2-bit complete sample;
byte #s indicate relative byte address within 32-bit word (little endian format)

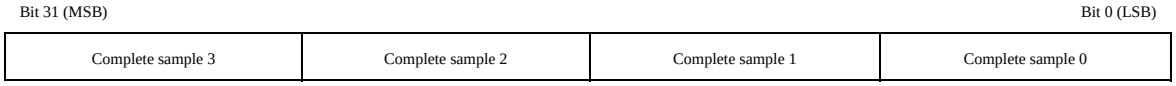

Figure 17: Data Array word for complete-sample length of 8 bits (i.e. $2^n*2^k=8$);
relative sample times are indicated in each 8-bit complete sample

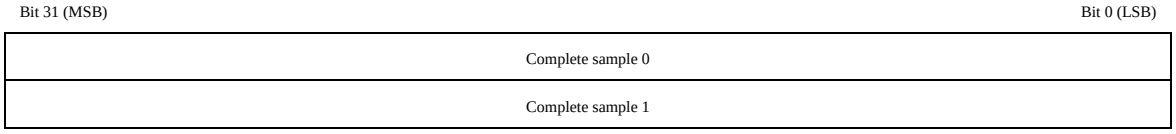

Figure 18: Data Array word for complete-sample length of 32 bits (i.e. $2^n*2^k=32$);
relative sample times are indicated in each 32-bit complete sample

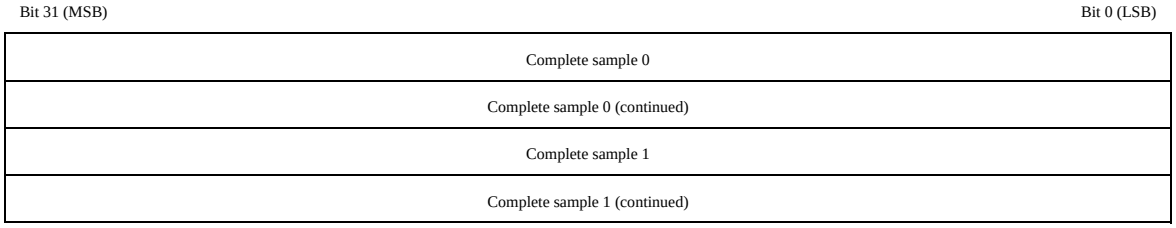

Figure 19: Data Array word for complete-sample length of 64 bits (i.e. $2^n*2^k=64$);
relative sample times are indicated in each 64-bit complete sample

Caution: Note that the length of multi-channel Data Frames can be much more constrained than single-channel Data Frames due to the requirement that each Data Frame contains an integral number of complete samples. This is particularly problematic when the number of channels is

large. Thus, multi-channel Data Frames may be significantly less suitable for e-VLBI transfer when it is desired to encapsulate one Data Frame per network packet. No such constraint normally exists for VDIF Data Frames stored in disk files or transferred via ftp.

9.4 Multi-channel complex-data Data Array format

For multi-channel complex data, the number of channels is also limited to 2^n, with each complex-sample component containing 2^k bits, so that the length of a complete sample is $2*2^n*2^k$ bits. Within each complex sample, the 'I' and 'Q' (In-Phase and Quadrature, respectively) components are adjacent with the 'I' occupying the lesser-significant position; all samples with the same time tag are then packed adjacently into a complete sample of length $2*2^n*2^k$ bits. Samples cell are populated into the Data Array according to the three rules specified in Section 9.3.

10. Sample representation

VDIF-encoded data samples are represented by the desired number of bits in a fixed-point 'offset binary sequence', beginning with all 0's for the most-negative sampled value to all 1's for the most-positive sampled value. For example, 2-bit/sample coding is (in order from most negative to most positive) 00, 01, 10, 11. This coding is compatible with existing Mark 5B, K5 and LBADR disk-based VLBI data systems.

11. Non-continuous data

Note that the VDIF specification allows Data Frames that are discontinuous in time. For example, Data Frames may be generated or transmitted only over the active part of a pulsar pulse. This is a perfectly legitimate use of the VDIF. Each Data Frame stands on its own so that there is no ambiguity or confusion.

12. Multiple Data Threads

Up to 1024 Data Threads may be mixed into a single Data Stream. Common usages would be 1) one Data Thread per channel in a multi-channel system, or 2) one or more multi-channel Data Threads, each with the same number of channels, sample rate and bits/sample. Arbitrary mixing of Data Threads is allowed within a single Data Stream, even with different sample rates, bits/sample, etc. In an effort to categorize types of multi-Thread Data Streams, we define below 'simple' and 'complex' Data Streams. Regardless of the type of Data Stream, the user is cautioned to verify that the intended receiving/processing target system can properly accept such a data flow

12.1 'Simple' VDIF Data Stream

Though the VDIF specification is quite flexible, even allowing a Data Stream to contain multiple Data Threads with different numbers of channels and even different sample rates and bits/sample, it is expected that most usage will be of two types: 1) one or more single-channel Data Threads, each with the same sample rate and bits/sample, and 2) one or more multi-channel Data Threads, each with the same number of channels, sample rate and bits/sample. We define both of these usage types as 'simple' Data Streams.

The rule for creation of a 'simple' Data Stream is as follows:

> Each Data Thread within a 'simple' VDIF Data Stream must have the same #channels, #bits/sample, data type ('real' or 'complex'), #Data Frames/sec, Data Frame Header Length and Data Array length.

Note that Data Threads with different Station IDs are allowed within a 'simple' Data Stream.

It is useful to create a Format Designator to specify the characteristics of a 'simple' VDIF Data Stream to aid in logging and processing the data. The VDIF Format Designator is an ASCII character sequence defined as:

<total date rate[4] (Mbps)> - <total # chans> - <bits/sample> [- <#threads>]

where #threads is assumed to be "1" if the <#threads> parameter is not specified. For example, the Format Designator

1024-16-2-1

specifies a total data rate of 1024 Mbps divided into 16 channels of 2 bits/sample and formatted into a single (16-channel) Data Threads; since this is a single thread, the Format Designator '1024-16-2' is equivalent (note this designator follows the widely used legacy VLBA recording mode designator). Another example – the Format Designator

1024-16-2-16

also specifies a total data rate of 1024 Mbps divided into 16 channels of 2 bits/sample, but formatted into 16 single-channel threads. One more example:

1024-16-2-4

again specifies a total data rate of 1024 Mbps divided into 16 channels of 2 bits/sample, but formatted into four 4-channel threads.

When using the VDIF Format Designator in any context where it might be confused with a mode or format designator of another type (for example, the legacy VLBA mode designator), it is suggested that the prefix 'VDIF-' be added – for example:

VDIF-1024-16-2-4

It is important to recognize that the VDIF Format Designator is a convenience only and is not required to decode the actual Data Stream. In fact, the VIDF Format Designator can always be re-constructed by reading a small section of the Data Stream.

12.2 'Compound' VDIF Data Stream

A 'compound' Data Stream contains multiple intermixed 'simple' Data Streams, each of which independently adheres to the rule stated in Section 12.1.

The rules for creation of a 'compound' VDIF Data stream are as follows:

1. A 'compound' VDIF Data Stream is a merging of two or more 'simple' Data Streams.
2. The set of Thread ID numbers within each constituent 'simple' Data Stream must occupy an exclusive, non-overlapping numerical range.

The lowest numbered Thread ID in each such 'simple' Data Stream' is defined as the Base Thread ID. Knowledge of the Base Thread ID for each constituent 'simple' Data Stream allows easy identification of all Data Threads within each constituent 'simple' Data Stream simply by examining the Thread ID in each Data Frame.

Usage of 'compound' Data Streams is discouraged unless the intended processing target is explicitly able to accept such a data format.

Within the context of a 'compound' Data Stream, each constituent 'simple' Data Stream is known as a Data Group, each of which has a corresponding 'simple' Data Thread Format Designator. The corresponding 'compound' Data Stream Format Designator is of the form

<DataGroup1 Designator> + <DataGroup2 Designator> + …..

For example, '1024-16-2-16+256-8-2' specifies two 'simple' Data Streams within the 'compound' Data Stream, the first with a data rate of 1024 Mbps, 16 channels, 2 bits/channel in 16 threads, and the second with a data rate of 256 Mbps, 8 channels, 2 bits/sample in 1 thread.

[4] Actual total sample data rate, not including Frame Headers or pad bits in Data Arrays.

13. Channel-numbering convention

Unique Channel identification with a Data Stream is often necessary for the proper specification of data handling or processing. For single-channel Data Threads, the most straightforward identifier is simply the numerical Thread ID. For channels within multi-channel Data Threads, a specification of the form '<ThreadID>-<chan# within thread>' (e.g. '5-2') seems most natural, where channel numbering begins at zero for the channel in the most LSB position of a sample cell and increments by one for each neighboring channel within the sample cell.

14. File-naming conventions

Disk files composed of Data Frames in VDIF format should be named with the suffix "vdf". Otherwise, file-naming should adhere to the internationally-agreed file-naming conventions specified in

http://www.haystack.mit.edu/tech/vlbi/evlbi/evlbi_memos/filenaming_conventions.pdf.

In some cases it may be useful to include the VDIF Format Designator as part of the filename. For this purpose, it is suggested that the VDIF Format Designator be included in the file name following the scan name, as in this example:

'gre53_ef_scan035_fd=1024-16-2.vdif'

which specifies data from experiment 'gre53', station 'ef', scan name 'scan035' with data in VDIF format '1024-16-2'.

VDIF Task Force:
Mark Kettenis, JIVE
Chris Phillips, CSIRO/ATNF
Mamoru Sekido, NICT
Alan Whitney, MIT (chair)

Addendum 1: Glossary

Base Tread ID (see Data Stream)

Channel (Section 3): In the context of the VDIF specification, a "channel" is normally the time-sampled data from a single frequency sub-band.

Complete Sample (Section 9.3): Within a Data Array, a *complete sample* is the sample data from all data channels for a single sample time (relevant for multi-channel data only).

Data Array (Section 3): The part of a Data Frame that contains the sampled data corresponding to the time-tag in the Data Frame Header.

Data Frame (Section 3): The basic VDIF data structure, containing a Data Frame Header with time-tag and data-identification information, followed by a Data Array containing the actual corresponding time-sampled data from one or more data channels, as shown in Figure 1.

Data Group (see Data Stream)

Data Frame Header (Section 3): The part of a Data Frame with time-tag and data-identification information and, optionally, with Extended User Data.

Data Segment (Section 3): For purposes of VDIF, a "Data Segment" is defined as a Data Stream with a well-defined beginning and end. For real-time network data transfer, a Data Segment normally starts at the first received packet of a VDIF Data Stream and ends at the termination of that Data Stream. For data captured onto a VDIF-format disk file, a single disk file constitutes a Data Segment. A Data Segment may contain any number of physical observations (often called "scans").

Data Stream (Section 3): An intermixed data flow of one or more Data Threads, where each Data Thread consists contains a unique Thread ID

- 'Simple' VDIF Data Stream (Section 12.1): A Data Stream limited to 1) one or more single-channel Data Threads, each with the same sample rate, #bits/sample, and data type ('real' or 'complex'), or 2) one or more multi-channel Data Threads, each with the same number of channels, sample rate, #bits/sample, and data type ('real' or complex').
- 'Compound' VDIF Data Stream (Section 12.2): A Data Stream containing two or more 'simple' VDIF Data Streams. Each constituent 'simple' Data Stream is identified by a Base Thread ID: the number corresponding to the lowest-numbered Thread ID of a constituent 'simple' Data Stream. Knowledge of the set of Base Thread IDs for a 'compound' Data Stream allows easy association of any particular Data Thread with its corresponding constituent 'simple' Data Stream (known as Data Group).

Data Thread (Section 3): A Data Thread consists of a time sequence of Data Frames with the same numerical Thread ID (embedded in the Data Frame Header).

Extended User Data (Section 6): Part of the Data Frame Header with unspecified format which may be optionally used for user-generated data.

Reference Epoch (Section 6): The epoch for which the second counter in the Data Frame Header is identically zero.

VDIF (Section 1): Acronym for VLBI Data Interchange Format

VDIF Format Designator (Sections 12.1 and 12.2): A short-hand ASCII descriptor for the characteristics of a particular VDIF Data Stream

Addendum 2: VDIF Hardware/Firmware Design Considerations

Although not part of the formal VDIF specification, the following notes are offered as suggestions to be kept in mind when designing VDIF-compatible systems:

1. The VDIF second tick is normally set by a *one-time* synchronization with a high-quality station tick (typically from H-maser or GPS), and thereafter is independently generated by reference only to the station frequency standard (or a signal that is phase-locked to the station frequency standard, such as the sampler clock). It is suggested that synchronization of the VDIF second tick be done prior to, and independently of, the VDIF clock setting, so that the VDIF clock setting can be adjusted, if necessary, without a risk of changing the epoch of the VDIF second tick.
2. The VDIF clock setting is typically done by arming the VDIF clock to be set to a user-specified value (UTC at next second tick) at the occurrence of the 'next VDIF second tick'. Following its initial setting, the VDIF clock increments the VDIF second count (carried in Word 0 of each Data Frame Header) on every subsequent second tick. The VDIF second counter should rollover back to zero after reaching a value of 2^{28}-1.
3. As a double-check on the correct setting of the VDIF clock, it is suggested that each station periodically record (in the experiment log file) the correspondence between station UTC time and the corresponding VDIF time code (read from the hardware/firmware VDIF clock). This will allow correlator personnel to double-check the VDIF clock setting if necessary. When performing this check, care must be taken to ensure that the both recorded times refer to the same second. One simple procedure is for the VDIF hardware/firmware to delay response to a VDIF clock-reading request until just after the next-occurring VDIF second tick (responding with the just-updated VDIF time code), after which the corresponding station time is immediately read.
4. The user must be able to infer the Data Array format from the specification of the #channels and #bits/sample in the Data Frame Header. This will work only if the Data Array is strictly formatted according to the Data Array format specifications in this document.
5. At a minimum, the user must be able to specify the Data Frame length and Station ID. Some systems may require additional flexibility in allowing the user to specify additional parameters.
6. Users are strongly encouraged to adopt an operating mode of one channel per Data Thread, which the hardware/firmware should support. This mode of operation is most compatible with the emerging generation of software-based correlators, and will result in the most efficient operation when used in association with such correlator systems.