VLBI Data Interchange Format (VDIF) Specification

23 September 2008
VDIF Task Force
Version Draft A.2

1. Introduction

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 resources.

The goal of the VLBI Data Interchange Format (VDIF) specification is to define 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, including disk-file format.

The VDIF specification explicitly makes no attempt to define a data-transport protocol, which is expected to be the subject of a subsequent specification document. However, the VDIF specification 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 result of the deliberations and discussions of the VDIF Task Force, mostly via e-mail, and represents our best effort to answer the challenge presented to us.

3. Basic VDIF structure

The discussions at the Shanghai meeting affirmed the concept of a ‘framed’ data-stream format consisting of stream of fixed-length “Data Frames”, each carrying a short identifying Data Frame Header, followed by a Data Array (containing the actual samples), as the preferred method of addressing the VDIF specification challenge.

Accordingly, the VDIF specification is based upon the creation of a ‘Data Stream’ which consists of a serial set of ‘Data Frames’, each of which carries a time segment of time-sampled data from one or more data channels. The length of a Data Frame will normally 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 Data Stream of VDIF Data Stream may represent VLBI data through a network transfer or on a physical disk file.[1]

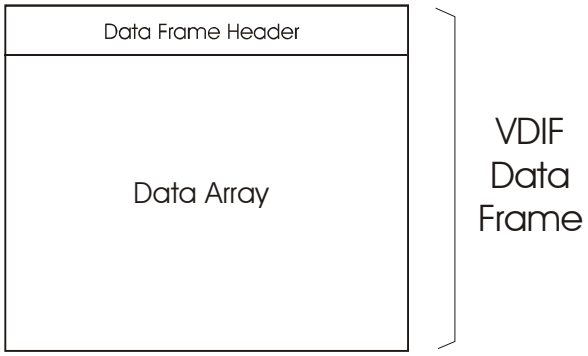

Figure 1: VDIF Data Frame structure showing Data Frame Header and Data Array

In some cases, the entire set of sampled channels can be carried in each Data Frame, so that a serial set of Data Frame carries the entire data set. In other cases, a single Data Frame may carry data from only a single (or perhaps a few) data channel(s) from among a set of many channels, in which case a logically parallel set of Data Frames are needed to represent the entire data set. In the VDIF concept, each time-series of Data Frames from the same set of channel(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 entire data set are generally organized so that all Data Frames with the same time-segment of data are adjacent, as shown schematically in Figure 2.

4. VDIF Definitions

For purposes of VDIF, we define the following terms:

Data Array: A sequence of contiguous uniformly time-sampled values from one or more data channels (usually frequency channels) packaged into a single array. All channels in the Data Array must be sampled at the same rate and with the same number of bits/sample.

Data Frame: Same as a Data Array, but with an added informational Data Frame Header containing critical bookkeeping information, such as time-tag and an ID (a number). The same ID is assigned to the time sequence of Data Frames containing data from the same set of channels.

Data Thread: A Data Thread consists of a time sequence of Data Frames with the same ID ('Thread ID').

Data Stream: An intermixed data flow of one or more Data Threads, each Data Thread with a unique ID.

---

[1] A VDIF-compliant disk file consists of simply a serial stream of fixed-length VDIF Data Frames. A real-time VDIF data transfer, on the other hand, would normally consist of a single VDIF Data Frame surrounded by various layers of transport protocol (TCP, UDP, IP, etc). A serial stream of such network-transported VDIF Data Frames can always 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 VDIF Data Frame length before network transmission requiring 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 oblivious to the internal VDIF Data Frame length.

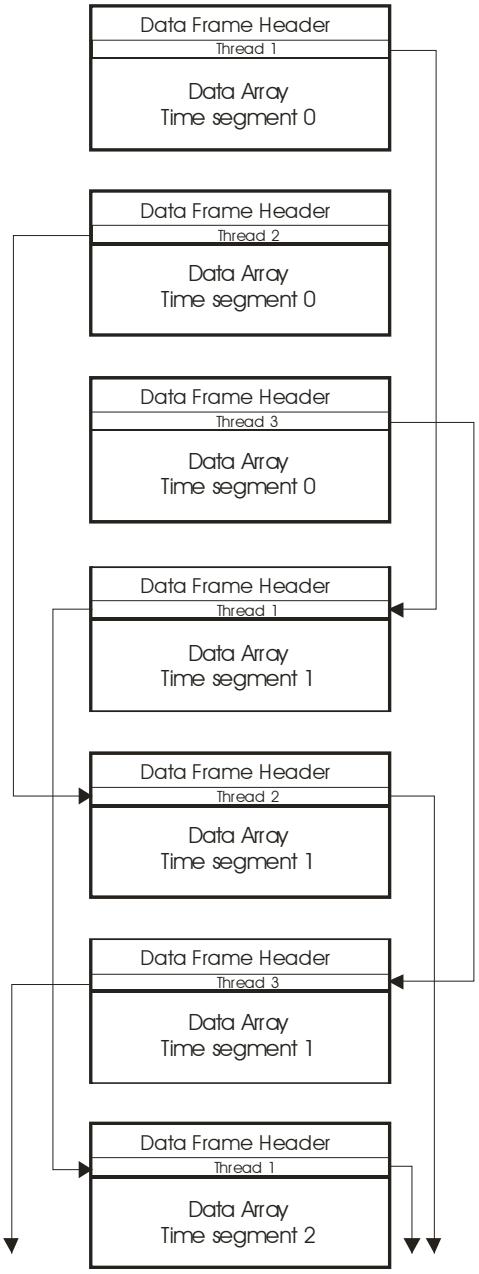

Figure 2: Illustration of Data Threads within a Data Stream

VDIF data file: A Data Stream captured to a disk file.

VDIF Scan: For purposes of VDIF, a "scan" is defined as a Data Stream with a well-defined beginning and end. For real-time network data transfer, a scan would normally start at the first received packet of a VDIF Data Stream and end at the termination of that Data Stream. For data captured into a VDIF-format disk file, the disk file typically constitutes a "scan". A "scan" may contain multiple physical observations.

In normal usage, it is expected that two types of Data Stream 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.

5. 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 data ID, the time tag of each sample, the number of sampled channels, and the bits/sample. (This allows 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 256 Data Threads 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.

6. VDIF Data Frame Rules

The following rules govern the VDIF Data Frame:

1. Each Data Frame contains a Data Frame Header followed by a Data Array.
2. All Data Frames within a given Thread ID must be of fixed length during a scan.
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 same number of bits/sample.
5. Each Data Array contains sample data from one or more channels in a format specified by VDIF (see Section 10).
6. The Data Frame length (including Data Frame header) 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. The first Data Frame of each one-second period must contain, as its first sample, the data taken on a UT second tick.

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 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. 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.

7. VDIF Data Frame Header

The standard 32-byte VDIF Data Frame Header is shown in Table 1.

	Bit 31					Bit 0
Word 0	Ref Epoch$_4$	Seconds from reference epoch$_{28}$				
Word 1	I_1	Data Frame # within second$_{32}$				
Word 2	Thread ID$_8$	Data Frame length – 1 (units of 8 bytes)$_{24}$				
Word 3	V_3	C_2	X_2	$\log_2$(#chns)$_4$	bits/sample$_5$	Station ID (2-char ASCII)$_{16}$
Word 4	EDV$_8$	Extended User Data$_{24}$				
Word 5	Extended User Data$_{32}$					
Word 6	Extended User Data$_{32}$					
Word 7	Extended User Data$_{32}$					

Table 1: Standard VDIF Data Frame Header format; subscripts are field lengths in bits

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

Word 0

Bit 31-28: Reference epoch for second count; see Note 1a

Bits 27-0: Seconds from reference epoch; see Note 1b

Word 1

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

Bits 30-0: Data Frame # within second; must be integral number of Data Frames per second

Word 2

Bits 31-24: Thread ID

Bits 23-0: Data Frame length-1 (including header) in units of 8 bytes; see Note 2

Word 3

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

Bit 28-27: Backwards compatibility modes; see Note 4

'00' - standard 32-byte VDIF Data Frame header and Data Array format
'01' - undefined
'10' – 'Header Compatibility' mode; Words 4-7 omitted from header
'11' – 'Header and Data-format Compatibility' mode

Bits 26-25: Unassigned (X)

Bits 24-21: $\log_2$(#channels in Data Array); #chans must be power of 2

Bits 20-16: #bits/sample-1 (32 bits/sample max)

Bits 15-0: 2-character ASCII station ID; see Note 5

Words 4-7

Extended User Data: Format and interpretation of extended user data is indicated by the value of User Data Version (UDV) in Word 4 Bits 31-24; see Note 6

Notes

1. The VDIF time code in Word 0 is divided into two fields:
   a. Reference epoch: A 4-bit field that contains the 6-month period in which the VDIF clock was *set*, with an origin of at 00H 1 Jan 2008, such that '0' corresponds to the first 6 months of 2008, '1' corresponds to the 6 months starting 00H 1 Jun 2008, etc. After setting, this 4-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 2016.
   b. Seconds from reference epoch: A 28-bit field that is initially set to second count within 6-month period [i.e. (day number within 6-month period, starting at zero)*86400+(second number within day)], and thereafter counts all seconds (including any leap seconds). This field counts seconds unambiguously for somewhat more than 8 years, then rolls-over back to 0 after reaching a count of 2^{28}-1.

   The VDIF time rolls over only if the VDIF clock continues running uninterrupted and un-reset for more than 8 years!

   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.
2. 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.
3. The VDIF version number in Word 0 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. The Data Array is unaffected.
4. For purposes of easing the transition from legacy VLBI disk-based data system to the VDIF standard, two backwards-compatibility modes are supported, as specified in Word 3 bits 28-27. These modes will be abandoned when the transition to VDIF is complete.
   a. 'Header Compatibility' mode: Words 4-7 are omitted, so that header length is reduced to 16 bytes
   b. 'Header and Data-Format Compatibility' mode: Words 4-7 are omitted, so that header length is reduced to 16 bytes. In addition, sample representation (i.e. coding) reverts to that used by the legacy Mark 5B/K5/LBADR disk-based data systems.
5. The 2-character ASCII station ID is intended to utilize the international assignment of station IDs coordinated through NRAO.
6. Extended User Data Words 4-7 are available for user-generated data. Each different usage of the extended words should be assigned a different User Data Version number (UDV) in Word 4 Bits 31-26 so that decoding software can automatically apply the proper decoding algorithm. UDV 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 UDV is set to '0'. Each VDIF version may have an independent set of UDV numbers.

8. Multiple Data Threads

Data Threads (up to max of 256) are mixed into a single Data Stream. The following rules apply:

1. Each Data Thread may have its own #channels, #bits/sample, sample rate and Data Frame length.
2. All Data Frame lengths within a single Data Threads must be identical.

Note: Use care in using mixed Data Frame lengths and/or sample rates within a single Data Stream; some data-processing systems may not accept such a data stream.

9. 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.

10. VDIF Data Array formats

VDIF specifies the format of the Data Array for a number of common combinations of #channels and #bits/sample. Adherence to these specified formats is necessary to ensure that the data are properly interpreted. The current definition of Data Array formats is not exhaustive, and additional formats will be defined as required.

10.1 Single-Channel Data Array Format

For purposes of our discussion, each Data Array is composed of an even number of 32-bit words. For Data Arrays carrying single-channel data, the suggested Data Array formats for 1, 2, 3, and 4 bits/sample are shown in Tables 2 through 5; larger number of bits/sample can be allocated similarly, but are not considered here. The first sample in the Data Array corresponds precisely to the time indicated in the Data Frame Header.

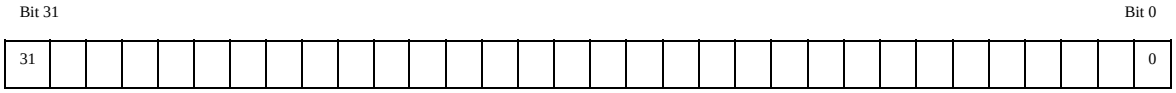

Table 2: 1-bit/sample data word format (numbers in boxes indicate relative sample #'s in time order)

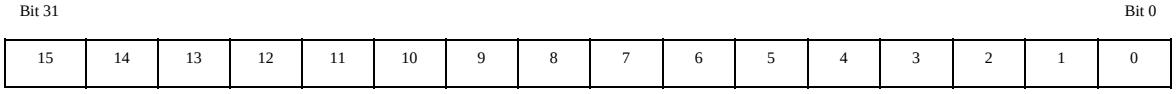

Table 3: 2 bits/sample data word format

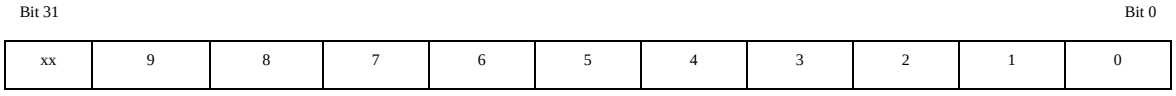

Table 4: 3 bits/sample data word format. 'xx' indicates unused bits

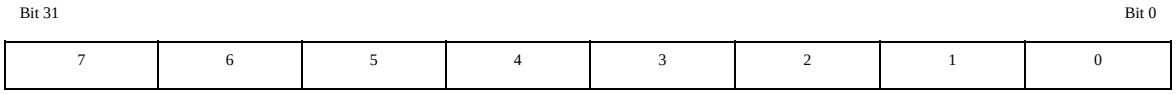

Table 5: 4 bits/sample data word format

For multi-bit samples, the LSB of each sample occupies the rightmost bit of the sample field.

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 in this regard are 1-4, 6, 7 and 11-32 (though the VDIF specification supports a maximum of 16 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.

10.2 Multi-channel 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^{15}=32768, maximum bits/sample is 2^5=32. Extension of these formats to include an arbitrary number of bits/sample is not contemplated. In all 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 must 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. The data from

Three rules govern the filling of 32-bit data words in the Data Array:

1. Each individual-channel sample within a complete sample is represented by an adjacent set of 2^k bits, with the LSB occupying the lesser-significant position (i.e. rightmost 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 sample cell of length 2^n*2^k bits. The sample cell is Tables 7 through 12 show 32-bit data-word usage for several example sample-cell lengths; other cases can be inferred.
3. Each Data Array must contain an integer number of complete samples (i.e. a complete sample may not span more than one Data Array).

Legacy disk-based systems such as the Mark 5B, K5 and LWADR 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. sample-cell length limited to 32 bits).

Bit 31 (MSB) Bit 0 (LSB)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

Bit 31 (MSB) Bit 0 (LSB)

Sample 3	Sample 2	Sample 1	Sample 0

Table 10: Data Array word for sample-cell length of 8 bits (relative sample times indicated in each sample cell); i.e. $2^n*2^k=8$

Bit 31 (MSB) Bit 0 (LSB)

Sample 0
Sample 1

Table 12: Data Array word for sample-cell length of 32 bits (relative sample times indicated in each sample cell); i.e. $2^n*2^k=32$

Bit 31 (MSB)	Bit 0 (LSB)
Sample 0	
Sample 0 (continued)	
Sample 1	
Sample 1 (continued)	

Table 12: Data Array word for sample-cell length of 64 bits (relative sample times indicated in each sample cell); i.e. $2^n*2^k=64$

11. Sample representation

Samples delivered by a modern DDS have already passed through substantial digital processing, and potentially can be carried with numerous bits of precision in cases where bandwidth limitations are not a consideration. Thus, it makes little sense to define specialized coding schemes based on the input sample thresholds. Further, disk-based recordings are not subject to paired-bit errors that caused legacy tape formats to adopt a sign-symmetric coding scheme. Accordingly, VDIF-encoded data samples are represented by the desired number of bits of a fixed-point two's-complement numeric value, beginning with the most significant bit. For example, 2-bit/sample coding is (in order from most negative to most positive) 10, 11, 00, 01.

For 'backward compatibility' mode 11 (Data Frame header Word 3 bit 28-27 set to '11'), the 2 bits/sample coding, in order from most negative to most positive, is 00, 01, 10, 11.

12. 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 still stands on its own as far as time-tag, so there is no ambiguity or confusion.

13. Byte ordering

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

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.

15. VDIF Hardware/Firmware Design Considerations

The following should be kept in mind when implementing the VDIF specification:

1. The VDIF clock that generates the VDIF time code must be settable by the user. Typically this is done by arming the VDIF clock to be set to a user-specified value at the 'next station tick'. This allows the VDIF clock to be set synchronously with the station second tick. Following its initial setting, the VDIF clock increments the VDIF second count (in Word 0 of each Data Frame Header) on every subsequent second tick. VDIF second counter should rollover back to zero after reaching a value of 2^{28}-1.
2. 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 UT 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.

3. 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.
4. 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.
5. 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.

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