SKA SOUTH AFRICA
SQUARE KILOMETRE ARRAY

SPEAD: Streaming Protocol for Exchanging Astronomical Data

Document number:	SPEAD
Revision:	K
Classification:	Open Source, GPL
Author:	Jason Manley, Marc Welz, Aaron Parsons
Date:	2009/11/09

Document History

Revision	Date of Issue	ECN Number	Comments
A	2009/04/17	N/A	Misc changes made after meeting between DBE, CSS and DSG teams. Packet header field numbering altered, inclusion of higher resolution fields, etc
B	2009/04/30	N/A	Added Timestamp scale factore field (47), updated explanation and example meta data packets accordingly
C	2009/07/01	N/A	Conversion from memo to interface control document
D	2009/07/10	N/A	Renamed to NRF-FF-ICD-W-402
E	2009/07/13	N/A	Renamed to NRF-FF-ICD-F-402
F	2009/07/20	N/A	Made changes requested by Alan, retrospectively changed the version numbers to letters
G	2009/07/27	N/A	Renamed to K8000-0027V1-002 ICD
H	2009/08/05	N/A	Changes from internal review
I	2009/09/21	N/A	Started incorporating changes from OAR of Thomas
J	2009/10/27	N/A	Renamed to SPADE for public release. Incremented to version 3 of the protocol. Added data descriptors.
K	2009/11/09	N/A	Renamed to SPEAD. Added unified option and payload descriptor.

Document Software

	Package	Version	Filename
Stylesheet	casperdoc	1.1.1	casperdoc.sty
Word processor	LaTeX	3.1415926-1.40.9 (Web2C 7.5.7)	spead.tex
Diagrams	epstopdf	2.9.5gw	images/ska_logo.pdf

Contents

List of Figures

List of Tables

List of Abbreviations

KAT	Karoo Array Telescope
KATCP	KAT Communication Protocol
ICD	Interface Control Document
IP	Internet Protocol
FF	Fringe Finder

1 Applicable and Reference Documents

1.1 Applicable Documents

The following documents are applicable to the extent stated herein. In the event of conflict between the contents of the applicable documents and this document, the applicable documents shall take precedence.

No applicable documents

1.2 Related Documents

The following documents are referenced in this document. In the event of conflict between the contents of the referenced documents and this document, this document shall take precedence.

2 Scope

This document describes a datastream format suitable for use by radio astronomy instruments. The data stream is distinct from the *KATCP* control protocol which follows a format described in [**?**].

3 Protocol Layers

The data is transmitted via a network interface. The network interface consists of the following layers, from top to bottom:

Application and Presentation : The data format as described in later sections of this document.

Session and Transport : UDP, noting that the receiving party can not request retransmission of lost, duplicated or corrupt packets.

Network : IP (Internet protocol)

Link : 10 Gbps, 1 Gbps or 100 Mbps Ethernet

Physical : SFP+, CX4 or other 10 Gigabit ethernet cable, Cat6, Cat5e etc.

4 Overview

Many instruments output data over ethernet. These streams are often high speed, unsuitable for TCP/IP links. UDP is often prefered in the cases where partial data-loss can be accomodated. This document aims to standardise the output format of such UDP packets.

An essential feature of the datastream is that it is self describing. This self description extends to both the content and interpretation of the option fields in the packet header, and of the data payload itself.

The same receive and transmit code may be used for multiple instruments as the layout of the packets are well defined. It is possible to record the raw datastream directly to disk for later parsing. It is flexible to allow metadata and descriptors to be injected into the datastream at runtime and recorded or processed along with the primary datastream.

The format of the packetised stream is described in this document.

Note that lower speed and ad-hoc data products may be transferred using the *KATCP* control interface [**?**].

5 Packet Data Representation

Each user datagram contains an application header, zero more optional fields (*options*) and a data payload. The header is 8 octets (64 bits) long, as is each option.

Data is sent Most Significant Byte first. Most significant bit is first on the wire (network byte order).

5.1 Header

All data stream packets have a header of the following form:

Number of bits	*Description*
(MSB) 16	Magic Number (constant 0x4b52, in ascii *KR*)
16	Version identifier)
16	Reserved
(LSB) 16	Number of options

Magic number : This is a constant value. All packets will start with the following 16 bit pattern: *(msb) 0100 1011 0101 0010 (lsb).*

Version identifier : This field identifies the version of the packet format. Designed in case future packet structures differ. The version described here is 0003.

Reserved : This field should be ignored. The space is reserved for future allocation.

Number of options : The value stored in this field indicates the number of optional 64-bit fields following the header. Any attached data follows directly after this number of 64-bit fields.

5.2 *Options*: Optional Fields

Each packet can contain zero or more 64-bit data fields, called *options*. Each option consists of 16 bits of identifier (MSb) and 48 bits of data (LSb). The number of options in a packet is defined by the *Number of options* field in the header.

If a packet containing an unknown option is received, it might be recorded, ignored or otherwise the receiver might request a metadata reissue to explain the content (see field 0x30). In any case, the reception of unknown options should not be considered illegal and should be handled cleanly by a receiver.

Generally the options will be self defining and thus do not need strict identification in this document. However, it is useful to define a number of standard options that are likely to be used across a wide range of deployments.

The following table defines these standardised *options* identifier codes and their meaning.

Dec	*Hex*	*Description*
0	0000	NULL - Ignore
1	0001	Instrument type
4	0004	Data payload length
5	0005	Data payload offset
13	000D	Stream control
14	000E	Metadata packet counter
48	0030	Option descriptor
49	0031	Payload descriptor

In addition to these standardised *options*, a number of reserved options are defined. These are options that are currently in use in systems, including those using previous versions of this protocol. Overloading of these should be avoided where practical.

User defined options are allowed, but these should have the MSb of the option ID set (i.e. option ID > 32767)

Dec	*Hex*	*Description*
2	0002	Instance ID and sub-instance source (24 bit instance + 24 bit engine num)
3	0003	Timestamp
6	0006	Previous packet data error counter
7	0007	ADC Sample rate (Hz)
8	0008	ADC Sample rate fractional (femto-Hz)
9	0009	Number of frequency channels
10	000A	Number of antennas
11	000B	Number of baselines
12	000C	FPGA clock cycles per sync pulse (F engine)
15	000F	Last DBE system sync time (Unix time since epoch, seconds)
16	0010	Last DBE system sync time (fractional seconds, femto-seconds)
17	0011	Center frequency (Hz)
18	0012	Center frequency fractional (femto-Hz)
19	0013	Bandwidth (Hz)
20	0014	Bandwidth fractional (femto-Hz)
21	0015	Number of accumulations per integration
22	0016	Quantisation scalar X-polarisation, real component
23	0017	Quantisation scalar X-polarisation, imaginary component
24	0018	Quantisation scalar Y-polarisation, real component
25	0019	Quantisation scalar Y-polarisation, imaginary component
26	001A	Phase
27	001B	Phase timestamp (ADC sample number)
28	001C	Delay
29	001D	Delay timestamp (ADC sample number)
30	001E	FFT shift pattern
31	001F	X engine accumulation length
32	0020	F engine quantised bits per sample
33	0021	F engine packet payload length
34	0022	10GbE destination port (X engine)
35	0023	10GbE destination port (F engine)
36	0024	10GbE destination IP address (X engine)
37	0025	10GbE destination IP address (F engine)
38	0026	Processing engine clock rate (Hz)
39	0027	ADC overflow
40	0028	ADC overflow Timestamp (ADC sample number)
41	0029	Quantisation overflow
42	002A	Quantisation overflow Timestamp (ADC sample number)
43	002B	FFT overflow
44	002C	FFT overflow Timestamp (ADC sample number)
45	002D	Accumulator overflow
46	002E	Accumulator overflow Timestamp (ADC sample number)
47	002F	Timestamp scale factor

5.3 Standard Options

5.3.1 Instrument Type

The instrument type field defines the source of the data. It also includes the payload id identifying the decoding to be used to interpret the data.

The 48-bit field is subdivided into 16 bits representing instrument type and 32 bits representing the payload id (currently only the 16lsb are used). The instrument type is used to provide scope to particular option and payload id's. Generally the instrument type will be fairly broad (e.g. KAT-7 which includes streamed data from all KAT-7 facility instruments).

The following are defined existing instrument types (16 MSbs):

Hex identifier	*Instrument*
0000	NULL instrument
0001	Pocket correlator (KAT Fringe Finder)
0002	Single-dish spectrometer

Note that header and options are specified to be transmitted MSb first, regardless of instrument type.

5.3.2 Data Payload Length

This is the size in bytes of the packet payload length (excluding the header and optional fields).

5.3.3 Data Payload Offset

The position, relative to the integration, at which the payload of the current packet starts.

This is used for re-assembling data which was split across multiple packets. It is reset back to zero for the following integration (use in conjunction with the timestamp to determine if data loss occurred).

5.3.4 Stream control and metadata packet counter

Indicates the status of the stream. This control packet is contained in a stand-alone packet not containing any actual data products. If present, the data field of the packet (ie after all the 64bit options) might contain text descriptors related to option are payload ids' in the stream. See fields 48 (0x30) and 49 (0x31) for further details.

The *most significant bit* of the stream control field indicates that this is the last metadata packet. Used in conjunction with the metadata packet counter, this allows you to check if all the metadata was successfully received.

The lower 47 bits are a number representing:

0 : Stream start

1 : Metadata reissue (response to command requesting metadata reissue)

2 : Stream stop

3 : Metadata update (some values have changed, probably in response to a command)

4 : Metadata description packet (see *Stream descriptors*)

>=5 : Reserved for future use. Ignore.

If only one packet is sent, the Metadata packet counter will be zero and the MSb in Stream control will be set.

5.4 Stream descriptors

Stream descriptors are used to provide receiving clients with the metadata required to decode and interpret options and data sent as part of a particular SPEAD stream. These descriptors are sent in a special metadata-only packet (*stream_control*==4), and must never appear in a packet where ordinary instrument data is present, lest the description strings be confused with actual data. In general these packets will be sent at the start of a session before data and general metadata packets.

Provision is made for both option (describe the meaning and interpretation of header options), and payload (the structure of the streamed data) descriptors. Although they share a similar syntax, it is important to understand the distinction between the two types.

5.4.1 Payload descriptor

This option describes the starting position of a string in the data field where the description for a given payload id may be found.

Number of bits	*Representation*
(MSB) 16	Payload ID that we are describing.
16	Index in bytes of string start position in data field. Indexed from zero.
(LSB) 16	Length of string in bytes including a null-terminator.

The data field of the descriptor packet then contains strings of the form:

unpack_string>[LF]count[LF]Payload_name[LF]Description

The unpack string tells the parser how to unpack the payload, and follows a C style printf convention, with the addition of referenced data types. The unpack string can contain one or more 24-bit unpack directives, with each directive referring to a consecutive data type in the payload. Each 24-bit directive has the following form: *unpack_type[bit_length | payload_id]*

Unpack character	*Representation*
0	interpret remaining 16 bits as a Payload ID
i	signed integer
u	unsigned integer
f	IEEE float
c	ASCII character
b	boolean

This allows the unpack string to reference another payload type, thus allowing hierarchical constructs. For non payload ID types, the remaining 16 bits are the bit length of a single field of the specified type.

For example an unpack string could contain: *u[32]u[32]f[64]0[002]*. This would translate into payload segments that contain two 32-bit unsigned integers, followed by a double, followed by a structure that is defined in Payload ID 002.

The count string allows multiple data constructs of the type specified in the unpack string to be placed contiguously in the data stream. It has the following form (56 bits): *count_source[count | option_id]*

Unpack character	*Representation*
0	the remaining 48 bits provide the count directly. (cast to an unsigned integer)
1	the count is read as the 48-bit value associated with the specified Option ID.

Payload names are intended to be parsed and presented in the receiver namespace and thus should not include whitespace. They may not contain [LF] either, as this is used for argument demarkation when parsing.

As an example, consider a descriptor packet containing the 16bit field id 0x31 with 48 bit content 0x0005 0000 0035.

Unpack size	*Unpacked value*
16 msb	048 decimal.
16b	005 decimal.
16b	000.
16b lsb	53 decimal.

This would describe a Payload ID of 0005 starting at offset 0 and having a length of 53 bytes. And the start of the packet's datafield could contain the string (with additional payload id's included that have not been shown in the above header.)

0[2][LF]1[9][LF]frequency[LF]A single frequency bin.[NULL]0[3][LF]1[11][LF]baseline[LF]A single baseline. Baseline ordering starts with the cross correlation between A0 and A1. Thereafter all cross correlations in natural order. Auto correlations follow.[NULL]0[4][LF]0[4][LF]polarisation[LF]The polarisation products. Order is XX, YY, XY, YX[NULL]u[32][LF]0[2][LF]complex_data[LF]The complex data point. Two uint32, real followed by imaginary[NULL].

This above would describe the current v0002 correlator output format. Essentially it shows a hierarchy the begins with two uint32's the form a single complex sample. Each 4 complex samples describe a set of polarisation products. The polarisation products are then grouped into a number of baselines (that is specified in option 0x0011. The baselines are finally grouped into frequency channels, the number of which is specified in option 0x0009. This fully describes the data payload for a single correlator dump.

5.4.2 Option descriptor

In exactly the same fashion as the payload descriptor, the option descriptor describes the starting position of a string in the packet data where the description for a given option field may be found. The syntax is the same, but only a subset of the descriptive features are used since the format of the option field is much more tightly defined.

Number of bits	*Representation*
(MSB) 16	Option ID that we are describing.
16	Index in bytes of string start position in data field. Indexed from zero.
(LSB) 16	Length of string in bytes including a null-terminator.

The data field of the metadata packet then contains strings of the form:

unpack_string>[LF]0[1][LF]Option_name[LF]Description

The unpack string is interpreted as before, with the exclusion of the indirect referencing of other descriptors. Further, the option field has a fixed 48-bit length. This implies, for instance, that a float can only be a maximum of 32-bits long. The count string is not used and is set to *0[1]* to indicate a single instance of the unpack string is to be used. An unpack string might look like: *u[4]b[1]f[32]c[8]b[1]b[1]b[1]*.

As an example, consider a metadata packet containing the 16bit field id 0x30 with 48 bit content of 12884902140, this would decode as:

Unpack size	*Unpacked value*
16 msb	048 decimal.
16b	003 decimal.
16b	000.
16b lsb	252 decimal.

And the start of the packet's datafield could contain the string *u[48][LF]0[1][LF]timestamp[LF]This field timestamps all data for any DBE instrument. Time zero corresponds to the time at which the system was last sync'd to a 1PPS signal. Each increment of this field corresponds to a change of timestamp_scale_factor x n_ADC_samples.[NULL]*

In the same packet, we might have a second metadata field with ID 48 and 48bit content of 128865534254. This would decode as:

Unpack size	*Unpacked value*
16 msb	048 decimal.
16b	030 decimal.
16b	252 decimal.
16b	302 decimal.

And at offset 252 in the datafield, we might find: *u[16]b[1]u[31][LF]0[1][LF]FFT Shift pattern[LF]Indicates if a divide-by-2 was done between butterfly stages in the FFT. LSB is last FFT stage, with every higher bit representing the previous FFT stage. Only the <FFT Length> LSbs are valid. Struct is input number (16b msb), polarisation (1bit), FFT shift pattern (31 bits). [NULL]*

A single metadata description should never straddle a packet boundary. Each metadata packet should be self-contained. This effectively limits the length of a single description string. Jumbo frames typically allow for over 8KiB of data, so these strings can still be reasonably long.

5.5 Reserved Options

5.5.1 Instance ID

In order to support sub-arrays, we need a means to identify to which instance of an instrument this packet belongs. Further, since larger instruments consist of multiple processing engines, this field provides a means of labelling the source engine (required for reassembling packetised correlator output in the correct order, for example).

It is subdivided into two 24-bit fields: the MSB 24-bits represent the instrument instance and the second 24-bits represent the engine instance ID of instrument subgroup. The significance of this field will change depending on the instrument.

For example, this entire option will always be zero for a system consisting of a single-board, single-antenna spectrometer. For more complex instruments, this option will be used to indicate that the stream belongs to a particular beam or correlator subgroup. For example, a beamformer might consist of ten FPGAs, each one outputting a subset of frequencies and capable of forming two beams. The first 24-bit field could then range [0 1] and the second field [0 1 2 3 4 5 6 7 8 9].

5.5.2 Timestamp and Timestamp Scale Factor

This field timestamps all data for the correlator, beamformer, spectrometer and other KAT DBE instruments. This field was designed to be easily generated on FPGA instruments. The product of the timestamp multiplied by the timestamp scale factor represents the number of ADC samples acquired.

$$Samples\_acquired = Timestamp * Timestamp\_scale\_factor$$

A zero acquired sample count corresponds to the time at which the system was last synchronised to a 1PPS signal. The last synchronisation time is reported in the metadata fields IDs 15 and 16 in real-time seconds since the Unix epoch.

$$Time_{current} = Time_{sync} * \frac{Samples\_acquired}{Clock_{adc}}$$

If data is integrated (eg correlator or spectrometer), then this value will represent the time that the accumulation began (the time of the ADC sample of the first channel in the first spectrum of the accumulation)[1].

5.5.3 Previous Packet Data Error Counter

If non-zero, indicates the number of accumulations that were not performed due to a failure. For example, if the user requested 1000 integrations, but only 891 were performed, this field's value would be $1000 - 891 = 109$. This applies to the PREVIOUS packet's data from this engine (ie you need to match the instance ID field). Omitting this option is equivalent to a zero error count.

5.5.4 Quantisation Scalar (X and Y Polarisation, Real and Imaginary)

Most instruments perform quantisation before accumulation (eg, packetised correlator and beamerformer: 4 bits). Digital gain control is implemented before quantisation to ensure correct bit selection. The value in this field indicates the level of gain applied. Gains are not necessarily linear. Values are unit-less and are specified per-frequency channel, per input. The gains are complex, having a real and imaginary component. The system's amplitude response should be recalibrated if this value changes.

The 48-bit field is subdivided as follows:

Number of bits	*Representation*
(MSB) 16	Input number
16	Frequency channel
(LSB) 16	Digital gain

5.5.5 Phase slope and Phase offset

LSB 32-bit values indicating current phase slope and offset for a given polarisation. Timestamps will accompany these updates.

5.5.6 FFT Shift pattern

Indicates if a divide-by-2 was done between butterfly stages in the FFT. LSB is last FFT stage, with every higher bit representing the previous FFT stage. Only the *FFT Length* LSBs are valid. This is an internal DBE parameter and should be constant.

Number of bits	*Representation*
(MSB) 16	Input number
1	Polarisation (X=0, 1=Y)
(LSB) 31	FFT shift pattern

[1]This places an upper bound on the scale factor of the number of samples processed per accumulation, while the lower bound is 1.

5.5.7 X Engine Accumulation Length

Represents the accumulation which takes place within the X engine (packetised correlator specific). This places a lower limit on the integration time as well as sets the granularity of the accumulation period.

5.5.8 F Engine Packet Payload Length

Should be equal to X engine accumulation length (packetised correlator specific).

5.5.9 Processing engine clock rate

The clock rate of the X engines and B engines. This value is an estimate. These run asynchronously to the F engines and should be clocked at a higher rate to ensure buffers don't overflow. For POCO, this should be $sample\_clk/4$.

Number of bits	*Representation*
(MSB) 16	Board number
(LSB) 32	Clock speed (Hz)

5.5.10 ADC overflow status

This field indicates that an overflow occurred on said ADC (lsb set) at time indicated by field ID 42. Assume that overflow continued until another update is received for this ADC (lsb cleared).

Number of bits	*Representation*
(MSB) 16	ADC Input number
1	Polarisation (X=0, 1=Y)
31	Reserved. Ignore.
(LSB) 1	ADC overflow occurred on this input.

5.5.11 Center frequencies, bandwidth, ADC sample rates

These frequencies are given in integer Hz, with an optional fractional component in femto-Hz. If no optional fractional component is given, assume it to be zero.

A Example Packet Exchange

This section provides an example packet exchange that could occur between a simple correlator and a SPEAD receiver instance.

A.1 Packet Sequence

At the start of the data transmission, initial metadata packets are sent. These should setup any stream descriptors required to describe options and payloads. They will also include non-payload specific option fields that do not require transmission synchronously with the data packets.

During transmission of the data stream, packets should contain only a limited number of optional fields, in order to maximise the utilisation of the link. Certain options are sent if parameters change or errors are encountered.

Full metadata packets can be regenerated upon request, allowing a receiver to process a stream which had been started earlier.

When a data stream terminates, a futher metadata packet is sent, this may not contain a full description of the data stream, but does include stream control options to indicate that the transmission has completed.

The metadata packets may not arrive from the same source as the data packets. They may not even arrive from the same subnet. The receiver should not care about the source of the packet, as the contents of a stream are sufficient to fully identify the sender (barring any rogue packets on the network).

It is recommended that unknown options be recorded, in order to facilitate debugging and forward compatibility, even if these options are not understood.

A.2 Descriptor packets

Revision 3 of the protocol (as described here) allows for initial descriptor packets to describe the nature of the variables in the stream and also detail how to unpack the data for the receiving parser. Revisions 1 and 2 did not include this facility, though the protocol was designed to be forwards and backwards compatible. Earlier receivers might have these decode values hard-coded.

A transmitter conforming to Revision 3 could send a descriptor packet similar to the following:

Packet contents	*Representation*	*Significant meaning*
`0x4B52 0003 0000 000B`	Header	11 options will follow.
`0x000D 0000 0000 0004`	Stream control	Metadata descr, not the last.
`0x000E 0000 0000 0000`	Metadata pkt cnt	First metadata packet in this stream.
`0x0001 0001 C020 000C`	Instrument type	PoCo, 12 complex 32b unsgnd ints, msb first per data item.
`0x0003 0000 0000 0000`	Instance ID	Stream 0 from engine 0.
`0x0030 001E 00FC 012E`	Option descr	Option 0x1E descr. at payload offset 252 is 302 bytes long.
`0x0030 0003 0000 00FC`	Option descr	Option 0x03 descr. at payload start is 252 bytes long.
...	...	...
`0x0031 0001 0352 011D`	Payload descr	Paylod 0x01 descr. at payload offset 850 is 285 bytes long.

Directly following those options, we'd have the following ASCII string, where values in brackets correspond to raw binary representations:

u[48][LF]0[1][LF]Timestamp[LF]This field timestamps all data for any DBE instrument. Time zero corresponds to the time at which the system was last sync'd to a 1PPS signal. Each increment of this field corresponds to a change of timestamp_scale_factor x n_ADC_samples.[NULL]u[16]b[1]u[31][LF]0[1][LF]FFT Shift pattern[LF]Indicates if a divide-by-2 was done between butterfly stages in the FFT. LSB is last FFT stage, with every higher bit representing the previous FFT stage. Only the <FFT Length> LSbs are valid. Struct is input number (16b msb), polarisation (1bit), FFT shift pattern (31 bits). [NULL]....0[2][LF]1[9][LF]frequency[LF]A single frequency bin.[NULL]

Thus the receiver would know how to interpret option ID 0x1E and 0x03 as well as payload ID 0x01.

These descriptions are intended to supplement this document. Although every effort is made to describe all required fields, it is understood that future systems may need to supplement these fields. These descriptors provide a mechanism for achieving this. It will still be necessary for a receiver to understand certain standard fields a-priori (eg Stream control, Metadata packet counter, Instrument type, Instance ID, Metadata descriptor etc), in order to decode adescriptor packet.

It is recommended that all option fields used in a given stream be described this way and that a full list of descriptors be pre-pended to the datastream.

A.3 Metadata packets

Once the descriptors have been transmitted the receiver will be in a position to decode additional option fields. These are then sent in a number of metadata packets before the start of payload transmission. For example,the *poco* and KAT-7 *corr* instruments (and any other instrument implementing per-channel gain control) will need to issue many metadata packets describing the digital gain setting on each frequency channel for each ADC.

Number of bytes	*Option ID*	*Representation*
8	Header	
8	0x000D	Stream control
8	0x000E	Metadata packet counter
8	0x0001	Instrument type
8	0x0003	Instance ID
8	0x000F	Start time, whole seconds
8	0x0011	Center frequency (Hz)
8	0x0013	Bandwidth (Hz)
8	0x0007	ADC Sample Rate (Hz)
8	0x002F	Timestamp Scale Factor
8	0x0009	Number of channels
8	0x000A	Number of antennas
8	0x000B	Number of baselines
8	0x0015	Number of accumulations per integration
8	0x0016	Quantisation scalar X-pol, real
8	0x0016	Quantisation scalar X-pol, real
8	0x0016	Quantisation scalar X-pol, real
...	...	...
8	0x0016	Quantisation scalar X-pol, real

Many such metadata packets will be sent to describe all the quantisation scalars for all ADCs, all polarisations, all channels. Each subsequent packet will increment the metadata packet counter field's value by one, which allows the receiver to check that no metadata packets were missed. The length of each packet can be calculated based on the number of options in the preamble. The last packet will have the MSb in the Stream Control field set, indicating the end of the metadata bundle.

If only one packet is sent, the metadata packet counter will be zero and the MSb in Stream Control will be set.

During the normal course of operation, additional metadata packets may be received with updated values. In this case, the Stream Control field will have a value of 3 and the updated values should be recorded.

A.4 Data packets

Data packets will generally have only a small number of option fields that are required to unambiguously describe the payload of the particular packet in question.

Both the *poco* mode and packetised *corr* (correlator) instruments will emit packets with a data payload typically containing the following options, noting that the ordering of options may vary.

Number of bytes	*Option ID*	*Representation*
(MSB) 8	Header	
8	0x0001	Instrument type
8	0x0002	Instance ID
8	0x0003	Timestamp
8	0x0004	Data payload length
8	0x0000	NULL
8	0x0005	Data payload offset (in bytes)
(LSB) 4096	Data	

The receiver will use the payload ID that forms part of the instrument type to correctly decode the data payload. Note that a payload descriptor refers to a complete, unfragmented data block. Blocks of data that are too large for a single packet will get fragmented. The receiver can re-assemble these fragments using the payload length and payload offset information. Once unfragmented that block can be parsed using the supplied descriptor.

The length of the payload may vary, but is likely to be 2kiB, 4kiB or 8kiB.

The last packet of a data block may be shorter if there is not enough data to fill the packet. ie.

$$(n\_bls * n\_chans * n\_stokes * 2/n_{xeng})\%(pkt\_len)! = 0$$

The fifth field is kept in reserve for future use.

Once data transmission is complete a Stream Stop packet is issued.