Raw OLAP data formats

OLAP produces several data formats, which are intended to be replaced by their final format, such as HDF5. The formats below are not officially supported and subject to change without notice.

Beamformed Data

Beamformed data can be recorded as either complex voltages (yielding X and Y polarisations) or one or more stokes. In either case, a sequence of blocks will be stored, each of which consists of a header and data. The header is defined as:

```
struct header {
  uint32 sequence_number; /* big endian */
  char padding[508];
};
```

in which sequence_number starts at 0, and is increased by 1 for every block. Missing sequence numbers implies missing data. The padding can have any value and is to be ignored.

Complex Voltages

Each (pencil) beam produces two files: one containing the X polarisation, and one containing the Y polarisation. The names of these files adhere to the following scheme:

Lxxxxx_Byyy_S0-bf.raw	X polarisations of beam yyy of observation xxxxx
Lxxxxx_Byyy_S1-bf.raw	Y polarisations of beam yyy of observation xxxxx

Each file is a sequence of blocks of the following structure:

```
struct block {
  struct header header;

  /* big endian */
  fcomplex voltages[SAMPLES|2][SUBBANDS][CHANNELS];
}
```

Stokes

Each (pencil) beam produces one or four files: one containing the Stokes I (power) values, and optionally three files for Stokes Q, U, and V, respectively. The names of these files adhere to the following scheme:

Lxxxxx_Byyy_S0-bf.raw	Stokes I of beam yyy of observation xxxxx
Lxxxxx_Byyy_S1-bf.raw	Stokes Q of beam yyy of observation xxxxx
Lxxxxx_Byyy_S2-bf.raw	Stokes U of beam yyy of observation xxxxx
Lxxxxx_Byyy_S3-bf.raw	Stokes V of beam yyy of observation xxxxx

Currently (release 2010-09-20), the Stokes U and V are multiplied by a factor of 1/2, but that will be changed in a subsequent release.

Each file is a sequence of blocks of the following structure:

```
struct block {
  struct header header;

  /* big endian */
  float stokes[SAMPLES|2][SUBBANDS][CHANNELS];
}
```

Types and constants

Types

A 'float' is a 32-bit IEEE floating point number. An 'fcomplex' is a complex number defined as

```
struct fcomplex {
  float real;
  float imag;
};
```

Constants

Constants can be computed using the parset file. Below is a translation between the C constants used above and their respective parset keys:

SAMPLES	The number of time samples in a block	OLAP.CNProc.integrationSteps
SUBBANDS	The number of subbands (beamlets) specified	len(Oberservation.subbandList)
CHANNELS	The number of channels per subband	Observation.channelsPerSubband

Useful routines

The following routines might be useful when reading raw OLAP data.

Byte swapping

```
uint32 swap_uint32( uint32 x )
{
  union {
    char c[4];
    uint32 i;
  } src,dst;
```

```
  src.i = x;
  dst.c[0] = src.c[3];
  dst.c[1] = src.c[2];
  dst.c[2] = src.c[1];
  dst.c[3] = src.c[0];

  return dst.i;
}

float swap_float( char *x )
{
  union {
    char c[4];
    float f;
  } dst;

  dst.c[0] = x[3];
  dst.c[1] = x[2];
  dst.c[2] = x[1];
  dst.c[3] = x[0];

  return dst.f;
}
```

Note that swap_float does not take a float as an argument. If you need to convert endianness, you cannot read the wrong endian float and correct for it later, as the compiler/platform is allowed to change the bytes in the float before you can do the conversion (for instance, normalisation).