The Data Access Library (DAL)

Lars Bähren

Radboud Universiteit Nijmegen
Afdeling Sterrenkunde

LOFAR/HDF5 Meeting
09 Sep 2010
1 Motivation & Goals

2 Architecture & Development

3 Standard LOFAR data products

4 HDF5

5 Representation of World Coordinates
Motivations for creating the DAL

- Abstraction of details specific to a set of data formats
  - FITS
  - MeasurementSet
Motivations for creating the DAL

- Abstraction of details specific to a set of data formats
  - FITS
  - MeasurementSet
- Common interface to access conceptually similar data structures
  - Group
  - Attribute / Keyword
  - Table
  - Array of datapoints
Motivations for creating the DAL

- Abstraction of details specific to a set of data formats
  - FITS
  - MeasurementSet
- Common interface to access conceptually similar data structures
  - Group
  - Attribute /Keyword
  - Table
  - Array of datapoints
- Reference implementation for creation of and access to LOFAR standard data products
  - new types of data products
  - support for complex data model matching experiment characteristics
Requirements

- C++ library with a small amount of external package dependencies
  - DAL embedded in LUS, to handle external dependencies
  - object-oriented design
Requirements

- C++ library with a small amount of external package dependencies
  - DAL embedded in LUS, to handle external dependencies
  - object-oriented design
- Python bindings wrapping the functionality of the C++ library
Requirements

- C++ library with a small amount of external package dependencies
  - DAL embedded in LUS, to handle external dependencies
  - object-oriented design
- Python bindings wrapping the functionality of the C++ library
- portable across various platforms
  - build from source
  - cross-platform build system $\rightarrow$ CMake Cross-Platform Makefile Generator

Lars Bähren
RU Nijmegen
The Data Access Library (DAL)
**Motivation Architecture LOFAR Data HDF5 Coordinates**

## DAL library components

- **core**
  - common methods
  - definition of types
  - data format abstraction
**DAL library components**

- **core**
  - common methods
  - definition of types
  - data format abstraction

- **coordinates**
  - representation of world coordinates
**DAL library components**

- **core**
  - common methods
  - definition of types
  - data format abstraction

- **coordinates**
  - representation of world coordinates

- **data_common**
  - definition of interfaces for building blocks from which to construct LOFAR standard data products
DAL **library components**

- **core**
  - common methods
  - definition of types
  - data format abstraction
- **coordinates**
  - representation of world coordinates
- **data_common**
  - definition of interfaces for building blocks from which to construct LOFAR standard data products
- **data.hl**
  - high-level interfaces to LOFAR standard data products
DAL **library components**

- **core**
  - common methods
  - definition of types
  - data format abstraction

- **coordinates**
  - representation of world coordinates

- **data_common**
  - definition of interfaces for building blocks from which to construct LOFAR standard data products

- **data_nl**
  - high-level interfaces to LOFAR standard data products

- **bindings**
  - Bindings to the Python scripting language
## DAL external dependencies

- Data format libraries
  - HDF5 library
  - CFITSIO
  - CASACORE
Motivation Architecture LOFAR Data HDF5 Coordinates

DAL external dependencies

- Data format libraries
  - HDF5 library
  - CFITSIO
  - CASACORE

- Python bindings (optional)
  - Python
  - Boost
  - numutil
**DAL external dependencies**

- **Data format libraries**
  - HDF5 library
  - CFITSIO
  - CASACORE

- **Python bindings** (optional)
  - Python
  - Boost
  - num\_util

- **Misc. 3rd party**
  - WCSLIB
  - MPI
  - MySQL
DAL **external dependencies**

- **Data format libraries**
  - HDF5 library
  - CFITSIO
  - CASACORE

- **Python bindings** (optional)
  - Python
  - Boost
  - num\_util

- **Misc. 3rd party**
  - WCSLIB
  - MPI
  - MySQL

- **Configuration & build**
  - CMake
DAL *distribution & installation*

- Part of LOFAR User Software (LUS)
**DAL distribution & installation**

- Part of **LOFAR User Software (LUS)**
- Source available through SVN repository (public readable)

`svn co http://usg.lofar.org/svn/code/trunk lofarsoft`
DAL distribution & installation

- Part of LOFAR User Software (LUS)
- Source available through SVN repository (public readable)
  
  `svn co http://usg.lofar.org/svn/code/trunk lofarsoft`

- External package dependencies handled through build system

  cd build
  ./bootstrap
  make dal
DAL distribution & installation

- Part of LOFAR User Software (LUS)
- Source available through SVN repository (public readable)
  
  `svn co http://usg.lofar.org/svn/code/trunk lofarsoft`

- External package dependencies handled through build system
  
  `cd build`
  
  `./bootstrap`
  
  `make dal`

- Installed components
  
  - `libdal` – C++ library
  - `pydal` – Python module
  - Header files
  - Executables (application & test programs)
### DAL test builds

No file changed as of Wednesday, September 08 2010 07:00:00 CEST

<table>
<thead>
<tr>
<th>Site</th>
<th>Build Name</th>
<th>Update</th>
<th>Configure</th>
<th>Build</th>
<th>Test</th>
<th>Build Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Files</td>
<td>Min</td>
<td>Error</td>
<td>Warn</td>
<td>Min</td>
</tr>
<tr>
<td>ife001</td>
<td>Linux-c++</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

No Continuous Builds

### Experimental

<table>
<thead>
<tr>
<th>Site</th>
<th>Build Name</th>
<th>Update</th>
<th>Configure</th>
<th>Build</th>
<th>Test</th>
<th>Build Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ice063</td>
<td>Linux-c++</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ice063</td>
<td>Linux-c++</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ice063</td>
<td>Linux-c++</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>dop143</td>
<td>Linux-c++</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>36</td>
<td>1.6</td>
</tr>
</tbody>
</table>
Standard data products defined through ICDs:

- TBB Time-Series Data
  - raw signal: digitized voltages from individual dipole antennas
  - match hierarchical structure of physical elements
Standard data products defined through ICDs:

- **TBB Time-Series Data**
  - raw signal: digitized voltages from individual dipole antennas
  - match hierarchical structure of physical elements

- **Beam-Formed Data**
  - joint (correlated) signals from multiple dipoles / stations
Standard data products defined through ICDs:

- **TBB Time-Series Data**
  - raw signal: digitized voltages from individual dipole antennas
  - match hierarchical structure of physical elements

- **Beam-Formed Data**
  - joint (correlated) signals from multiple dipoles / stations

- **Radio Sky Image Cubes**
Standard data products defined through ICDs:

- **TBB Time-Series Data**
  - raw signal: digitized voltages from individual dipole antennas
  - match hierarchical structure of physical elements
- **Beam-Formed Data**
  - joint (correlated) signals from multiple dipoles / stations
- **Radio Sky Image Cubes**
- **Dynamic Spectrum Data**
Standard data products defined through ICDs:

- TBB Time-Series Data
  - raw signal: digitized voltages from individual dipole antennas
  - match hierarchical structure of physical elements
- Beam-Formed Data
  - joint (correlated) signals from multiple dipoles / stations
- Radio Sky Image Cubes
- Dynamic Spectrum Data
- Visibility Data
Standard data products defined through ICDs:

- **TBB Time-Series Data**
  - raw signal: digitized voltages from individual dipole antennas
  - match hierarchical structure of physical elements
- **Beam-Formed Data**
  - joint (correlated) signals from multiple dipoles / stations
- **Radio Sky Image Cubes**
- **Dynamic Spectrum Data**
- **Visibility Data**
- **Rotation Measure Synthesis Cubes**
TBB time-series: top-level view
Motivation
Architecture
LOFAR Data
HDF5
Coordinates

Beam-Formed H5
(single pathway shown to data)

LOFAR beam-formed
Root group
HEADER

SYSTEM-WIDE
LOSS
GROUP

SUB-ARRAY
POINTING 0
GROUP

SUB-ARRAY
POINTING 1
GROUP

MORE
SUB-ARRAY
POINTING
GROUPS...

MORE BEAM
GROUPS...

BEAM 0
GROUP

BEAM 1
GROUP

PROCESSING
HISTORY
GROUP

PROCESSING
HISTORY
GROUP

BEAM 0
GROUP

BEAM 1
GROUP

MORE BEAM
GROUPS...

COORDINATES
GROUP

PROCESSING
HISTORY
GROUP

Stokes_0
GROUP

Stokes_1
GROUP

Stokes_2
GROUP

Stokes_3
GROUP

LINEAR-COORD
GROUP

TABULAR-COORD
GROUP

Subband 0

Subband 1

Subband 2

Subband N

CH0 CH1 CH2 CH4

Time

Data Packaging Options:
1) One Table per Subband (1d) [Shown here]
2) One Table for all Subbands (Nd)
3) 1 Array per Subband (1d)
4) 1 Array for all Subbands (Nd)

Lars Bähren
RU Nijmegen

The Data Access Library (DAL)
Design approach

- modular, object-oriented implementation
**Design approach**

- modular, object-oriented implementation
  - one C++ class per type of group / dataset
Design approach

- modular, object-oriented implementation
  - one C++ class per type of group / dataset
  - common basic interface for all object classes
Design approach

- modular, object-oriented implementation
  - one C++ class per type of group / dataset
  - common basic interface for all object classes
  - hide low-level library call behind reasonably simple API
Design approach

- modular, object-oriented implementation
  - one C++ class per type of group / dataset
  - common basic interface for all object classes
  - hide low-level library call behind reasonably simple API
- recursive creation and transversal of hierarchical structures
Design approach

- modular, object-oriented implementation
  - one C++ class per type of group / dataset
  - common basic interface for all object classes
  - hide low-level library call behind reasonably simple API
- recursive creation and transversal of hierarchical structures
- allow working on sub-trees of hierarchical structure
**DAL dataset support – high-level interfaces**

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>More...</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DAL::BeamFormed</code></td>
<td>High-level interface between beam-formed data and the DAL.</td>
<td></td>
</tr>
<tr>
<td><code>DAL::BeamGroup</code></td>
<td>High-level interface between beam-formed data and the DAL.</td>
<td></td>
</tr>
<tr>
<td><code>DAL::BeamSubband</code></td>
<td>High-level interface between beam-formed data and the DAL.</td>
<td></td>
</tr>
<tr>
<td><code>DAL::BF_Beam</code></td>
<td>High-level interface to the station beam of a BF dataset.</td>
<td></td>
</tr>
<tr>
<td><code>DAL::BF_Dataset</code></td>
<td>High-level interface to the root-group of a beamformed dataset.</td>
<td></td>
</tr>
<tr>
<td><code>DAL::BF_PrimaryPointing</code></td>
<td>High-level interface to the Primary Pointing Direction of a BF dataset.</td>
<td></td>
</tr>
<tr>
<td><code>DAL::BF_ProcessingHistory</code></td>
<td>High-level interface to the processing history attached to a BF dataset.</td>
<td></td>
</tr>
<tr>
<td><code>DAL::BFRaw</code></td>
<td>High-level interface between raw beam-formed data and the DAL.</td>
<td></td>
</tr>
<tr>
<td><code>DAL::ITS_ExperimentMeta</code></td>
<td>Storage of meta information from an LOFAR ITS experiment.</td>
<td></td>
</tr>
<tr>
<td><code>DAL::LOPES_EventFile</code></td>
<td>Read in LOPES event files.</td>
<td></td>
</tr>
<tr>
<td><code>DAL::SysLog</code></td>
<td>High-level interface to the system logs attached to a beamformed dataset.</td>
<td></td>
</tr>
</tbody>
</table>
```cpp
class BF_Dataset : public CommonInterface {

    //! Name of the data file
    std::string filename_p;
    //! LOFAR common attributes attached to the root group of the dataset
    CommonAttributes commonAttributes_p;
    //! Sub-Array Pointings
    std::map<std::string,BF_SubArrayPointing> subArrayPointings_p;
    //! Container for system-wide logs
    std::map<std::string,SysLog> sysLog_p;

public:

    //! Default constructor
    BF_Dataset (std::string const &filename);

    //! Argumented constructor
    BF_Dataset (DAL::Filename &infile,
                bool const &create=true);

... 

    //! Open a beam group
    bool openBeam (unsigned int const &pointingID,
                   unsigned int const &beamID,
                   bool const &create=true);

```
**data_common example (2)**

Filename file ("123456789", "test", Filename::bf, Filename::h5);

BF_Dataset dataset (file);

/* open sub-array pointing groups, create if they do not exist yet */
dataset.openSubArrayPointing(0,true);
dataset.openSubArrayPointing(1,true);
dataset.openSubArrayPointing(2,true);

/* open beams residing in an existing group */
dataset.openBeam(0,0,true);
dataset.openBeam(0,1,true);

/* open beams residing in a not yet existing group */
dataset.openBeam(10,0,true);
dataset.openBeam(10,1,true);

/* Extract sub-group */
DAL::BF_SubArrayPointing pointing = dataset.subArrayPointing (0);
**DAL** dataset support – common functionality

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DAL::CommonAttributes</strong></td>
<td>Collection of attributes common to all LOFAR datasets. More...</td>
</tr>
<tr>
<td><strong>DAL::CommonInterface</strong></td>
<td>Common functionality for the high-level interfaces to the datasets. More...</td>
</tr>
<tr>
<td><strong>DAL::Filename</strong></td>
<td>Class to generate filenames matching the LOFAR convention. More...</td>
</tr>
<tr>
<td><strong>DAL::HDF5Dataset</strong></td>
<td>A class to encapsulate the operations required to work with a HDF5 dataset. More...</td>
</tr>
<tr>
<td><strong>DAL::HDF5Hyperslab</strong></td>
<td>A hyperslab region for selective access to a dataspace. More...</td>
</tr>
<tr>
<td><strong>DAL::HDF5Property</strong></td>
<td>Brief description for class HDF5Property. More...</td>
</tr>
<tr>
<td><strong>DAL::HDF5Table</strong></td>
<td>Brief description for class HDF5Table. More...</td>
</tr>
<tr>
<td><strong>DAL::SAS_Settings</strong></td>
<td>Brief description for class SAS_Settings. More...</td>
</tr>
<tr>
<td><strong>DAL::Timestamp</strong></td>
<td>Wrapper for the time information in its various formats. More...</td>
</tr>
<tr>
<td><strong>DAL::HDF5Attribute</strong></td>
<td>Collection of methods to deal with HDF5 attributes. More...</td>
</tr>
</tbody>
</table>
**data** **hl** example  

DAL::CommonInterface

- common functionality for the high-level interfaces to the datasets
data HL example

DAL::CommonInterface

- common functionality for the high-level interfaces to the datasets
- infrastructure for placing/opening a group/dataset within a file

```cpp
bool open (hid_t const &location,
           std::string const &name,
           bool const &create)
```
**data.hl example**  DAL::CommonInterface

- common functionality for the high-level interfaces to the datasets
- infrastructure for placing/opening a group/dataset within a file

```cpp
bool open (hid_t const &location,
           std::string const &name,
           bool const &create)
```

- common method to get/set attributes

```cpp
template <class T> bool getAttribute (std::string const &name,
                                      T &val)
```

```cpp
template <class T> bool setAttribute (std::string const &name,
                                      T const &val)
```
Data HL example DAL::HDF5Dataset

- encapsulate the operations required to work with a HDF5 dataset

```cpp
HDF5Dataset (hid_t const &location,
             std::string const &name,
             std::vector<hsize_t> const &shape,
             hid_t const &datatype)
```
**data_hl example**  DAL::HDF5Dataset

- encapsulate the operations required to work with a HDF5 dataset

```cpp
HDF5Dataset (hid_t const &location,
             std::string const &name,
             std::vector<hsize_t> const &shape,
             hid_t const &datatype)
```

- (partial) read of the data

```cpp
template <class T> bool readData (T data[],
                                  HDF5Hyperslab &slab)
```

```cpp
template <class T> bool writeData (T const data[],
                                   std::vector<int> const &start,
                                   std::vector<int> const &count,
                                   std::vector<int> const &block)
```

- automatically create hyperslab
- dynamically grow data at write if required
**data_hl example**   DAL::HDF5Hyperslab

- hyperslab region for selective access to a dataspace
data_hl example  DAL::HDF5Hyperslab

- hyperslab region for selective access to a dataspace
- book-keeping of hyperslab parameters

```cpp
HDF5Hyperslab (std::vector<int> const &start,
               std::vector<int> const &stride,
               std::vector<int> const &count,
               std::vector<int> const &block,
               H5S_seloper_t const &selection)
```
**data_hl example**  DAL::HDF5Hyperslab

- hyperslab region for selective access to a dataspace
- book-keeping of hyperslab parameters

```cpp
HDF5Hyperslab (std::vector<int> const &start,  
std::vector<int> const &stride,  
std::vector<int> const &count,  
std::vector<int> const &block,  
H5S_seloper_t const &selection)
```

- service functions

  ```cpp
  bool setGap (std::vector<int> const &gap)
  unsigned int nofDatapoints ()
  std::vector<hsize_t> end ()
  bool setHyperslab (hid_t &datasetID,  
                     hid_t &dataspaceID,  
                     bool const &resizeDataset)
  ```
World Coordinates

- coordinates that serve to locate a measurement in some multidimensional parameter space
World Coordinates

- coordinates that serve to locate a measurement in some multidimensional parameter space
- measurable quantity such as
  - time
  - length / distance
  - frequency or wavelength associated with a point in a spectrum
  - longitude and latitude in a conventional spherical coordinate system which define a direction in space
  - projection of spherical coordinates onto 2-dim space
World Coordinates

- coordinates that serve to locate a measurement in some multidimensional parameter space
- measurable quantity such as
  - time
  - length / distance
  - frequency or wavelength associated with a point in a spectrum
  - longitude and latitude in a conventional spherical coordinate system which define a direction in space
  - projection of spherical coordinates onto 2-dim space
- enumerations
  - “Stokes parameters” to describe the polarization properties of an electromagnetic wave
  - ID of a channel / dipole / ...
Series of seminal papers which are part of the FITS standard
Representations in FITS

■ Series of seminal papers which are part of the FITS standard
■ conversion of pixel coordinates to world coordinates is regarded as a multi-step process
Representations in FITS

Series of seminal papers which are part of the FITS standard
conversion of pixel coordinates to world coordinates is regarded as a multi-step process
standardized set up keywords to describe coordinate (frames)
Coordinates group

--- COORDINATES_GROUP
  |-- GROUPTYPE
  |--- REF_LOCATION_VALUE
  |--- REF_LOCATION_UNIT
  |--- REF_LOCATION_FRAME
  |--- REF_TIME_VALUE
  |--- REF_TIME_UNIT
  |--- REF_TIME_FRAME
  |--- NOF_COORDINATES
  |--- NOF_AXES
  |--- COORDINATE_TYPES
  |--- COORDINATE0
  | ... 
  `-- COORDINATE{N}

- container to collect set of coordinates

- Direction
- Linear
- Tabular
- Stokes
- Frequency
Coordinates group

/> COORDINATES_GROUP Group

|-- GROUPTYPE Attr. string
|-- REF_LOCATION_VALUE Attr. array<double,1>
|-- REF_LOCATION_UNIT Attr. array<string,1>
|-- REF_LOCATION_FRAME Attr. array<string,1>
|-- REF_TIME_VALUE Attr. double
|-- REF_TIME_UNIT Attr. string
|-- REF_TIME_FRAME Attr. string
|-- NOF_COORDINATES Attr. int
|-- NOF_AXES Attr. int
|-- COORDINATE_TYPES Attr. array<string,1>
|-- COORDINATE0 Group
|  ...
|-- COORDINATE{N} Group

- container to collect set of coordinates
  - Direction
  - Linear
  - Tabular
  - Stokes
  - Frequency

- hold basic data defining common reference frame
## Direction Coordinate

<table>
<thead>
<tr>
<th>Field/Keyword</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUPTYPE</td>
<td>string</td>
<td>Group type descriptor, DirectionCoord</td>
</tr>
<tr>
<td>COORDINATE_TYPE</td>
<td>string</td>
<td>Coordinate Type descriptor, Direction</td>
</tr>
<tr>
<td>NOF_AXES</td>
<td>int</td>
<td>N of coordinate axes</td>
</tr>
<tr>
<td>AXIS_NAMES</td>
<td>array&lt;string,1&gt;</td>
<td>World axis names</td>
</tr>
<tr>
<td>AXIS_UNITS</td>
<td>array&lt;string,1&gt;</td>
<td>Physical units along each coordinate axis.</td>
</tr>
<tr>
<td>REFERENCE_VALUE</td>
<td>array&lt;double,1&gt;</td>
<td>Coordinate value at the reference point</td>
</tr>
<tr>
<td>REFERENCE_PIXEL</td>
<td>array&lt;double,1&gt;</td>
<td>Array location of the reference point in pixels.</td>
</tr>
<tr>
<td>INCREMENT</td>
<td>array&lt;double,1&gt;</td>
<td>Coordinate increment at reference point.</td>
</tr>
<tr>
<td>PC</td>
<td>array&lt;double,1&gt;</td>
<td>Linear transformation matrix</td>
</tr>
<tr>
<td>EQUINOX</td>
<td>string</td>
<td>Equinox of the observation</td>
</tr>
<tr>
<td>RADEC_SYS</td>
<td>string</td>
<td>System of equatorial coordinates</td>
</tr>
<tr>
<td>PROJECTION</td>
<td>string</td>
<td>Spherical map projection</td>
</tr>
<tr>
<td>PROJECTION_PARAM</td>
<td>array&lt;double,1&gt;</td>
<td>Spherical projection parameters</td>
</tr>
<tr>
<td>LONPOLE</td>
<td>double</td>
<td>Native longitude of the celestial pole, $\phi_p$</td>
</tr>
<tr>
<td>LATPOLE</td>
<td>double</td>
<td>Native latitude of the celestial pole, $\theta_p$.</td>
</tr>
<tr>
<td>CONVERSION_SYSTEM</td>
<td>string</td>
<td>Coordinate conversion reference system</td>
</tr>
</tbody>
</table>

- projection of spherical coordinates onto 2-dim space
- specification of algorithm
Spherical map projections

- TAN (Gnomonic), STG (Stereographic), ZEA (Zenithal equal-area)
- CAR (Plate carrée), AIT (Hammer-Aitoff)
. Group ---
|-- GROUPTYPE Attr. string 'Coordinates'
|-- REF_LOCATION_VALUE Attr. array<double,1>
|-- REF_LOCATION_UNIT Attr. array<string,1>
|-- REF_LOCATION_FRAME Attr. string
|-- NOF_COORDINATES Attr. int 2
|-- NOF_AXES Attr. int 2
|-- COORDINATE_TYPES Attr. array<string,1> ['Linear','Linear']
|-- COORDINATE0 Group ---
| | -- GROUPTYPE Attr. string 'LinearCoord'
| | -- COORDINATE_TYPE Attr. string 'Linear'
| | -- NOF_AXES Attr. int 1
| | -- AXIS_NAMES Attr. array<string,1> ['Time']
| | -- AXIS_UNITS Attr. array<string,1> ['s']
| | -- REFERENCE_VALUE Attr. array<double,1> [1.0]
| | -- REFERENCE_PIXEL Attr. array<double,1> [0.0]
| | -- INCREMENT Attr. array<double,1> [0.5]
| | -- PC Attr. array<double,1> [1.0]
|-- COORDINATE1 Group ---
| | -- GROUPTYPE Attr. string 'LinearCoord'
| | -- COORDINATE_TYPE Attr. string 'Linear'
| | -- NOF_AXES Attr. int 1
| | -- AXIS_NAMES Attr. array<string,1> ['Frequency']
| | -- AXIS_UNITS Attr. array<string,1> ['Hz']
| | -- REFERENCE_VALUE Attr. array<double,1> [200.0]
| | -- REFERENCE_PIXEL Attr. array<double,1> [0.0]
| | -- INCREMENT Attr. array<double,1> [10.0]
| | -- PC Attr. array<double,1> [1.0]
Motivation Architecture LOFAR Data HDF5 Coordinates

Group ---
|-- GROUPTYPE Attr. string 'Coordinates'
|-- REF_LOCATION_VALUE Attr. array<double[1],1>
|-- REF_LOCATION_UNIT Attr. array<string[1],1>
|-- REF_LOCATION_FRAME Attr. string
|-- NOF_COORDINATES Attr. int 2
|-- NOF_AXES Attr. int 1
|-- COORDINATE_TYPES Attr. array<string[1],1> ['Linear', 'Tabular']
|-- COORDINATE0 Group ---
| |-- GROUPTYPE Attr. string 'LinearCoord'
| |-- COORDINATE_TYPE Attr. string 'Linear'
| |-- NOF_AXES Attr. int 2
| |-- AXIS_NAMES Attr. array<string[1],1> ['x', 'y']
| |-- AXIS_UNITS Attr. array<string[1],1> ['m', 'm']
| |-- REFERENCE_VALUE Attr. array<double[1],1> [0.0, 0.0]
| |-- REFERENCE_PIXEL Attr. array<double[1],1> [0.0, 0.0]
| |-- INCREMENT Attr. array<double[1],1> [1.0, 1.0]
| `-- PC Attr. array<double[1],1> [1.0, 0.0, 0.0, 1.0]
|-- COORDINATE1 Group ---
| |-- GROUPTYPE Attr. string 'TabularCoord'
| |-- COORDINATE_TYPE Attr. string 'Tabular'
| |-- NOF_AXES Attr. int 1
| |-- AXIS_NAMES Attr. array<string[1],1> ['z']
| |-- AXIS_UNITS Attr. array<string[1],1> ['m']
| |-- PIXEL_VALUES Attr. array<double> [0, 1, 2, 3, ... ]
| `-- WORLD_VALUES Attr. array<double> [0, 1, 4, 9, ... ]

The Data Access Library (DAL)

Lars Bähren
RU Nijmegen