Basics of Offline Analysis

- ROOT as scripting environment and interactive shell (customized version with ISDC support S/W included).
- Several levels of access for user:
  1. Direct access to executables from ROOT (see next vg).
  2. Analysis Scripts/Objects, wrapping several individual tools, otherwise similar to executables. Also with parameters.
  3. Analysis Pipelines (similar to ISDC production pipelines).
  4. Graphical User Interface (GUI) based on ROOT.
- Interfaces to generic analysis tools (XSPEC, SAOtng, . . .) will be included but be limited, e.g., to launching with the correct fileset.
- In addition to ISSW a suite of helper tools (group manipulation, export to standard formats, . . .) will be supplied.

Accessing Executables from ROOT

Dynamic interface, based on .par file for executable access from within ROOT:

```
// This creates a ROOT object named j_bin_evts_spectra
// allowing to run the executable j_bin_evts_spectra
ISDCtask j_bin_evts_spectra("j_bin_evts_spectra");

// set the parameters

j_bin_evts_spectra["inSWG"] = "swg1.fits[1]";
j_bin_evts_spectra["jemxNum"] = "1";   // Detector 1
j_bin_evts_spectra["evtType"] = "0";   // FULL events
j_bin_evts_spectra["binTime"] = "0.0"; // Bin all data
j_bin_evts_spectra["chatty"]  = "100"; // Quite chatty

// actually run the executable with these parameters
j_bin_evts_spectra.Run();
// log messages will be displayed as the AE runs

// display the output again
cout << j_bin_evts_spectra.Output();
```

User interface still under discussion and development.

Possible improvement: no quotes around numerical parameter values.

Analysis Scripts

Interface foreseen to be (practically) identical to that for executables.

May consist of several executables or of single executable plus ISDC "wrapper".

Example could be `j_correction`.

Analysis Pipelines

Built in hierarchical fashion from executables and scripts. Will probably again have parameters.

Supposed to be well commented and editable *at user's risk*.

GUI

Only very general ideas at the moment.

Built on top of Analysis Pipelines or Scripts.

Probably optimized for ease of use vs. control, but possibly individually configurable.

ISDC Schedule for ISSW V1

Extracts from the updated schedule:

`13 Mar 2000`	Generation of test Science Windows
`20 Mar 2000`	Resume integration of ScW pipeline
`31 May 2000`	Completion of ScW pipeline integration and installation on Operations Network
`05 Jun 2000`	Start of subsystem tests on ScW pipeline
`24 Jun 2000`	Start of system tests for ISDC V1

ISDC Schedule for ISSW V2

`01 Jun 2000`	Begin integration of V2 pipelines

All other dates to be discussed.

Analysis Levels (1)

- ISDC has defined *named levels* for analysis pipelines.
- Different instruments do not need to have all levels and can have a different sequence. But where possible, same naming scheme should be used.
- These levels correspond to specific instances of the OG or SWG. Marked with `ISDCLEVL` keyword.
- A user may repeat the analysis from any given level without necessarily repeating previous steps.
- Levels currently defined in SPI ICD:

`ISDCLEVL`	Analysis step
`RAW`	PreProcessing
`PRP`	Data Preparation
`COR`	Data Correction
`POIN`	Virtual pointing definition
`GTI`	Good-Time Interval definition
`DEAD`	Deadtime calculation
`IBIN`	Event binning for Imaging
`IBKG`	Background generation for Imaging
`ISRC`	Source Selection for Imaging
`IRSP`	(Image) Response derivation for Imaging
`IMAG`	Image deconvolution

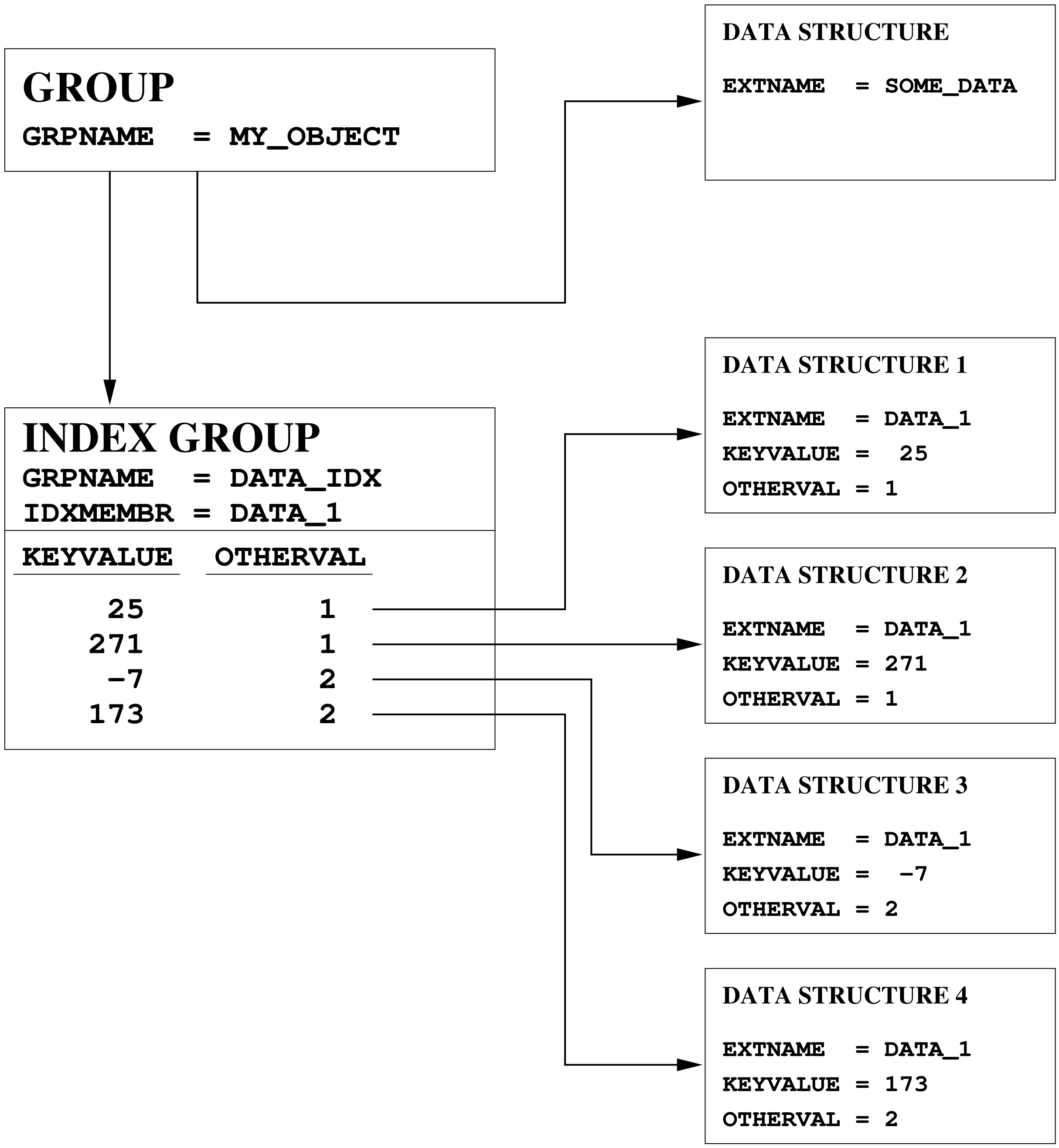
Analysis Levels (2)

- SDAST needs to define JEM-X analysis levels (⇒ analysis groups), matching names defined for SPI where possible.
- Also needs to define how a user would be able to repeat a certain step in Offline Analysis.
- All this should finally be described in the ADD.

Other Groups

- OG and SWG will only hold science, housekeeping and auxiliary data.
- Other data (calibration, instrument model, . . .) should be grouped “logically” or left with individual DOLs.
- Need to firm up discussion on these issues.

The idea behind Index Groups



Index functions in DAL3GEN (1)

```
int DAL3GENindexOpenElement(const char   *IndexDOL,
                            const char   *Selection,
                            int          *NumFound,
                            dal_element **OutputDS,
                            int           status)
```

Searches in an Index Group *given by its DOL* the data structures that match the given CFITSIO selection string.

```
int DAL3GENindexFindElement(dal_element  *InputDS,
                            const char   *NameDS,
                            const char   *Selection,
                            int          *NumFound,
                            dal_element **OutputDS,
                            int           status);
```

Searches named Index Group within an encompassing group and selects within the Index those data structures that match the given CFITSIO selection string.

```
int DAL3GENindexGetMember(dal_element  *InputDS,
                          const char   *NameDS,
                          int           NumMember,
                          dal_element **OutputDS,
                          int           status);
```

Returns the nth Member in an Index

Index functions in DAL3GEN (2)

```
int DAL3GENindexAttach(dal_element *Element,
                       dal_element *Group,
                       int          status)
```

Adds a structure to an INDEX.

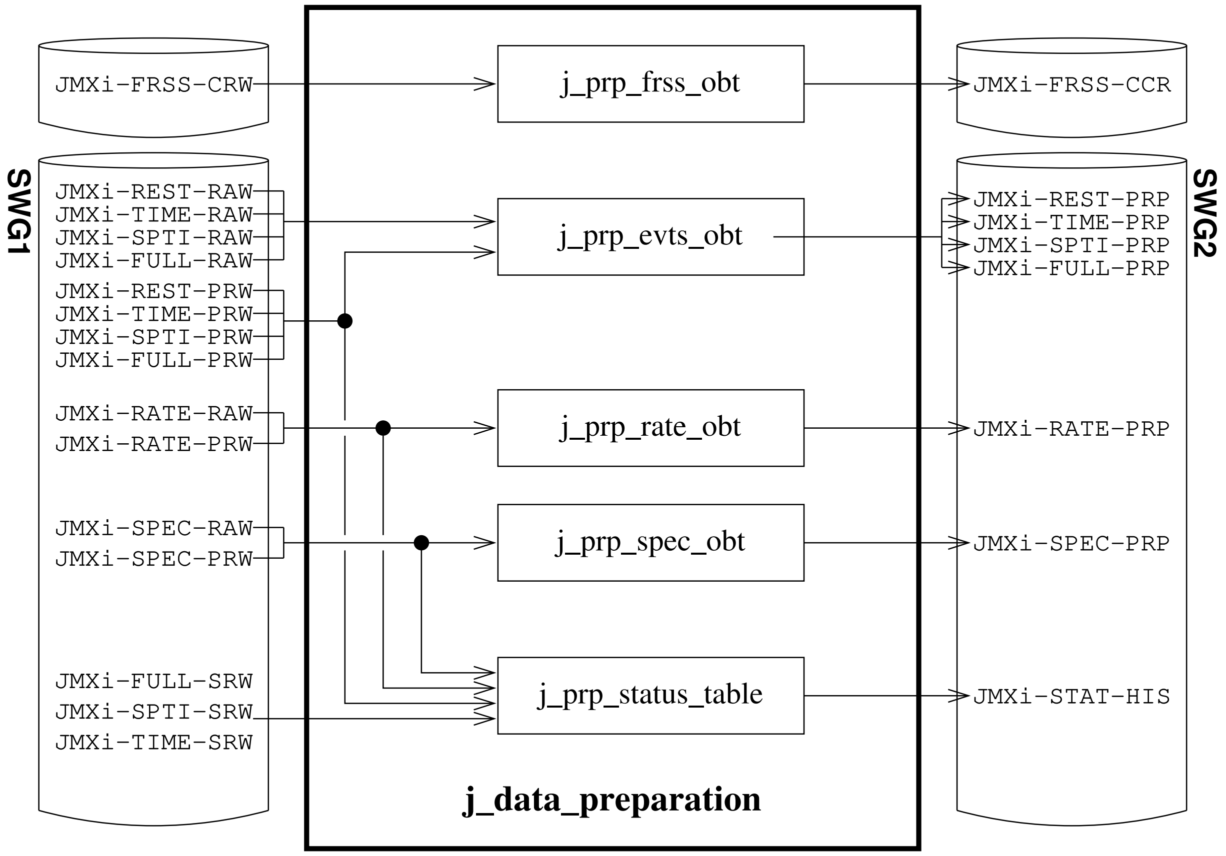
```
int DAL3GENindexUpdate(dal_element *Element,
                       dal_element *Group,
                       int          status);
```

Updates the keyword content of an Index for a particular Child.

```
int DAL3GENindexCreateMember(
        dal_element  *InputDS,
        const char   *NameDS,
        const char   *FileName,
        dal_element **NewMember,
        int           status);
```

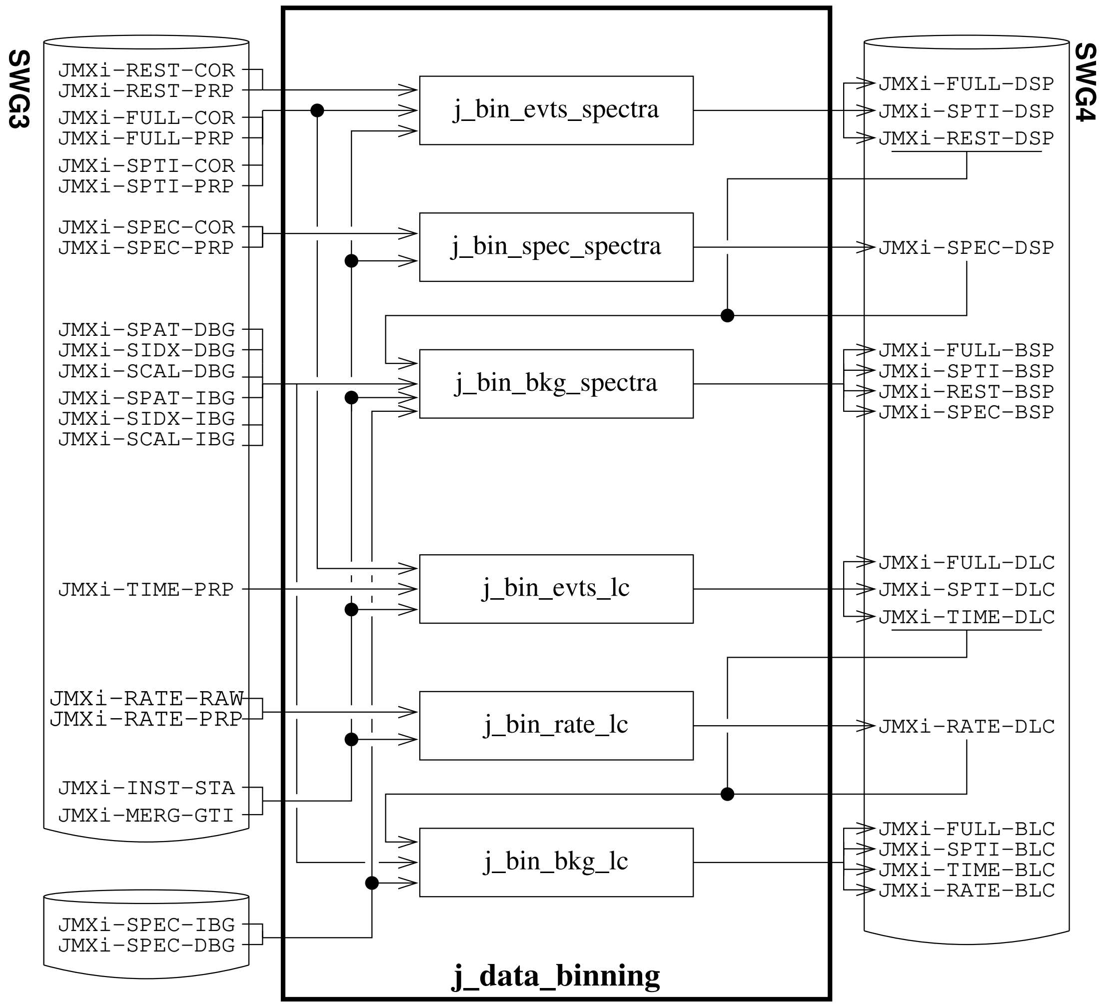
Creates a new instance of a Member inside a Group, and returns the `dal_element` pointer of the Member.

j_data_preparation



Component	DD	Code	Test	Doc
j_prp_frss_obt	100%	90%	0%	?%
j_prp_evts_obt	100%	100%	100%	100%
j_prp_rate_obt	100%	100%	100%	100%
j_prp_spec_obt	100%	100%	100%	100%
j_prp_status_table	$\longrightarrow$ CAO			

j_data_binning



Component	DD	Code	Test	Doc
j_bin_evts_spectra	90%	90%	0%	30%
j_bin_spec_spectra	50%	0%	0%	0%
j_bin_bkg_spectra	30%	0%	0%	0%
j_bin_evts_lc	90%	60%	0%	30%
j_bin_rate_lc	50%	0%	0%	0%
j_bin_bkg_lc	30%	0%	0%	0%