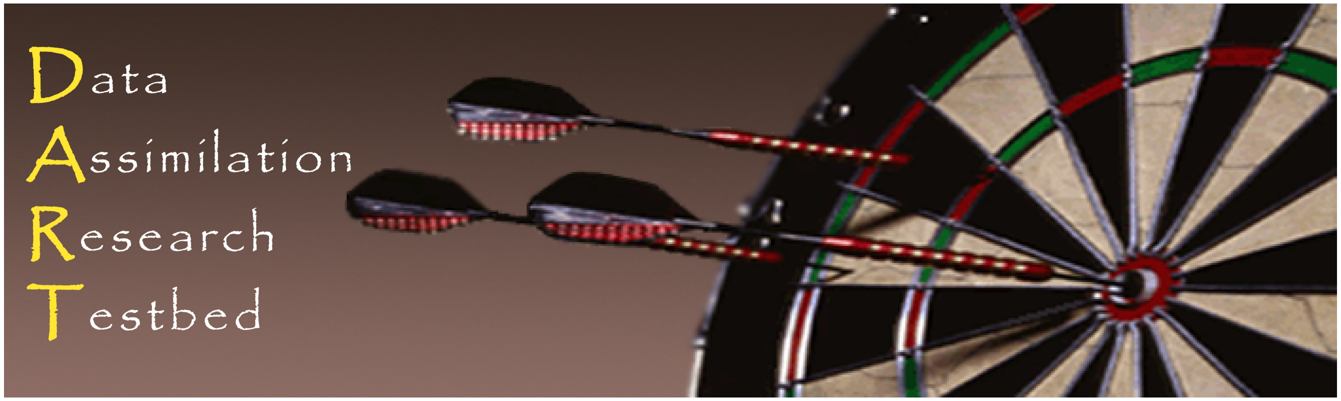
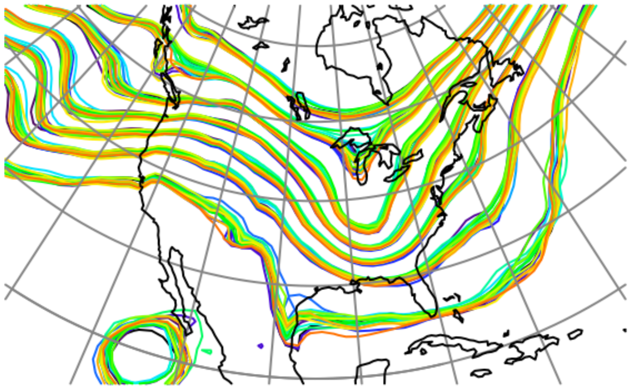


DART Tutorial Section 21: Observation Types and Observing System Design

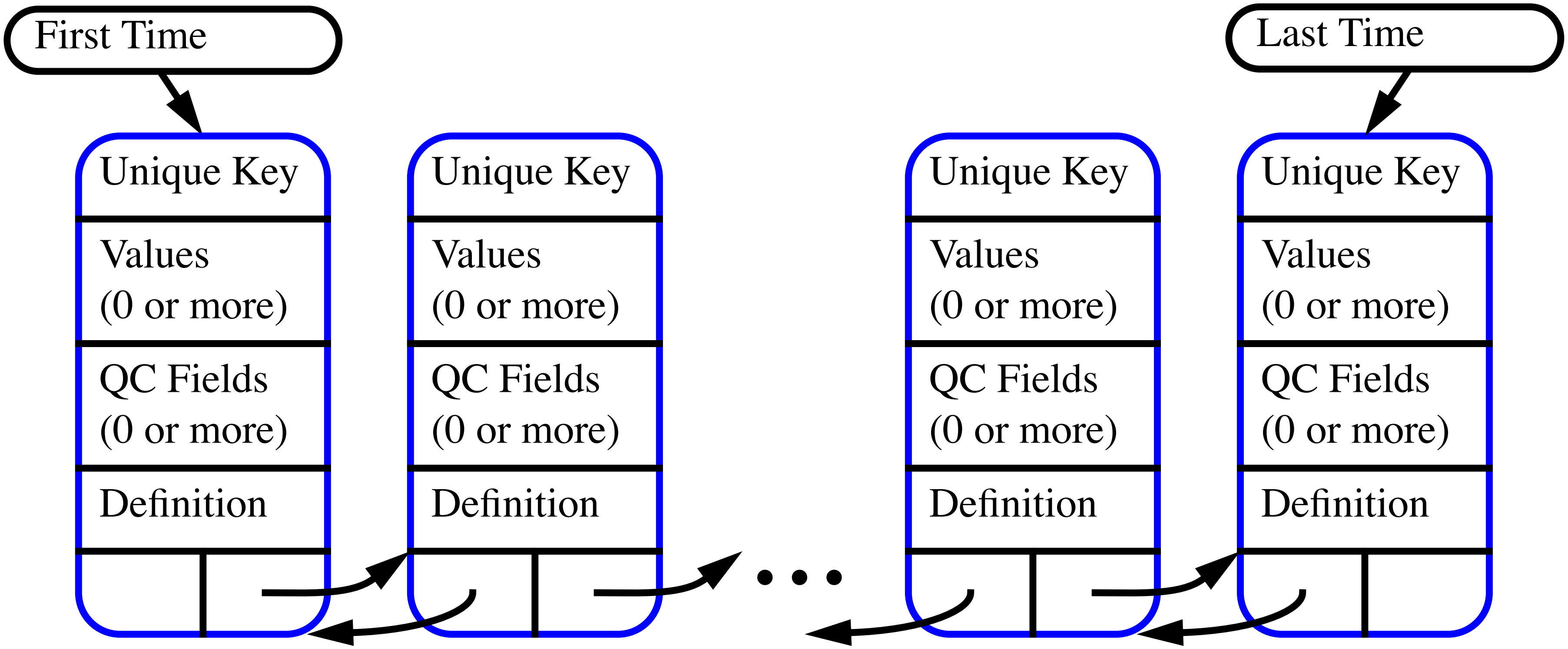


The National Center for Atmospheric Research is sponsored by the National Science Foundation. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

NCAR UCAR | National Center for Atmospheric Research

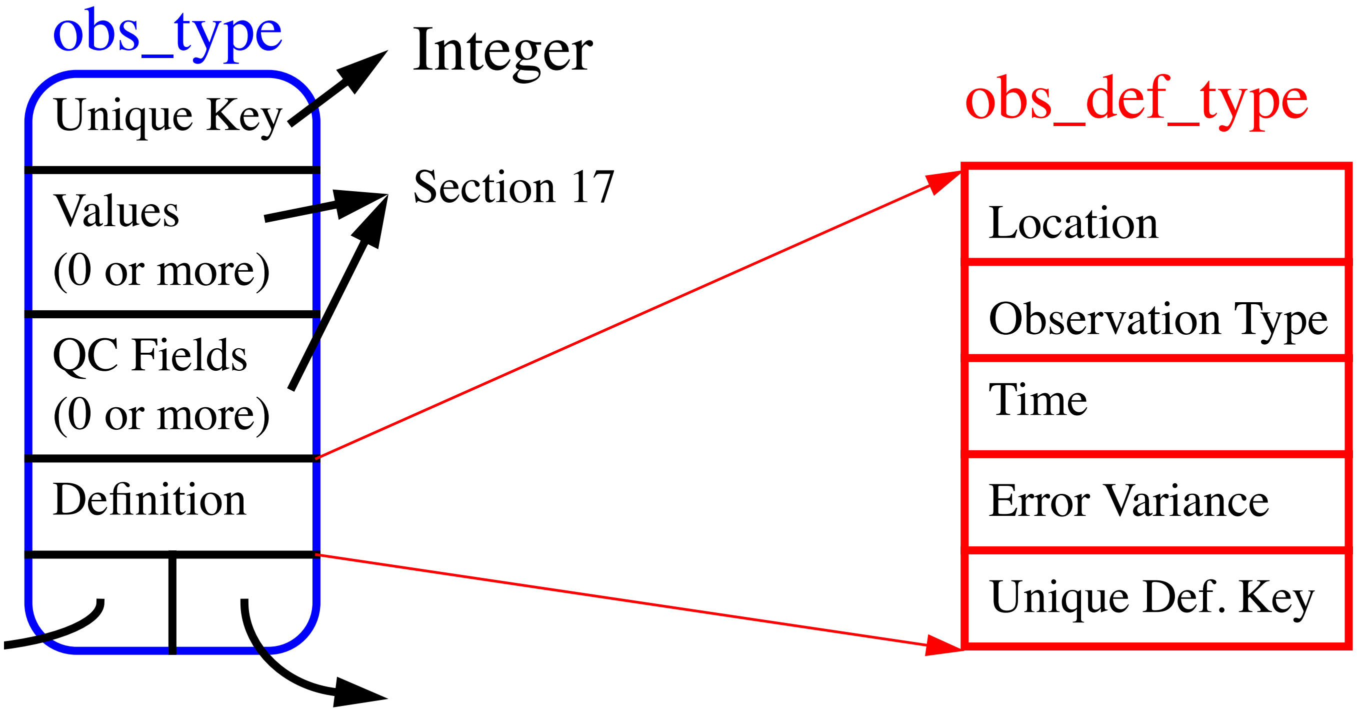
DART Assimilations controlled by Observation Sequence Files

Observation sequence files contain a time-ordered list of observations. (Stored with a 'linked list' of increasing times; obs do not have to be physically in time order in the file.)

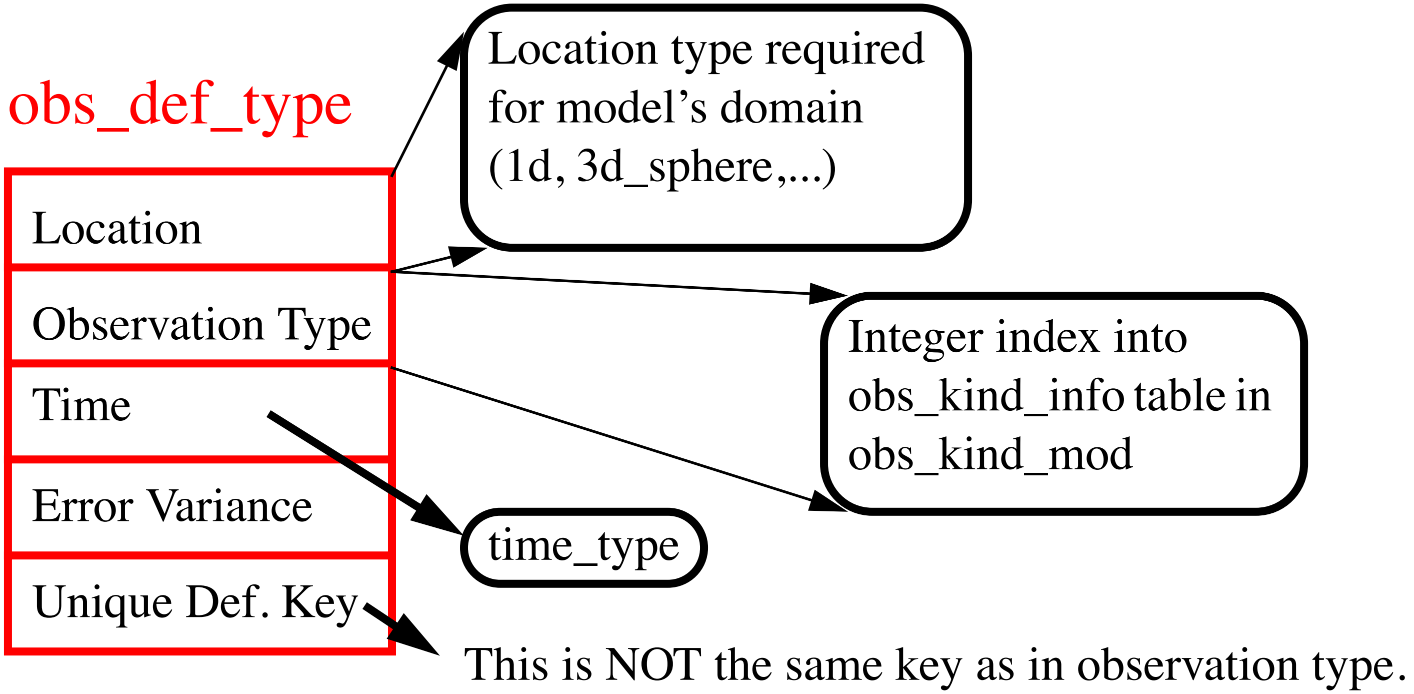


DART filter 'assimilates' until it runs out of observations.
Same for synthetic observation generation with *perfect_model_obs*

Observation Type Details



Observation Type Details



obs_def_type

Location

Observation Type

Time

Error Variance

Unique Def. Key

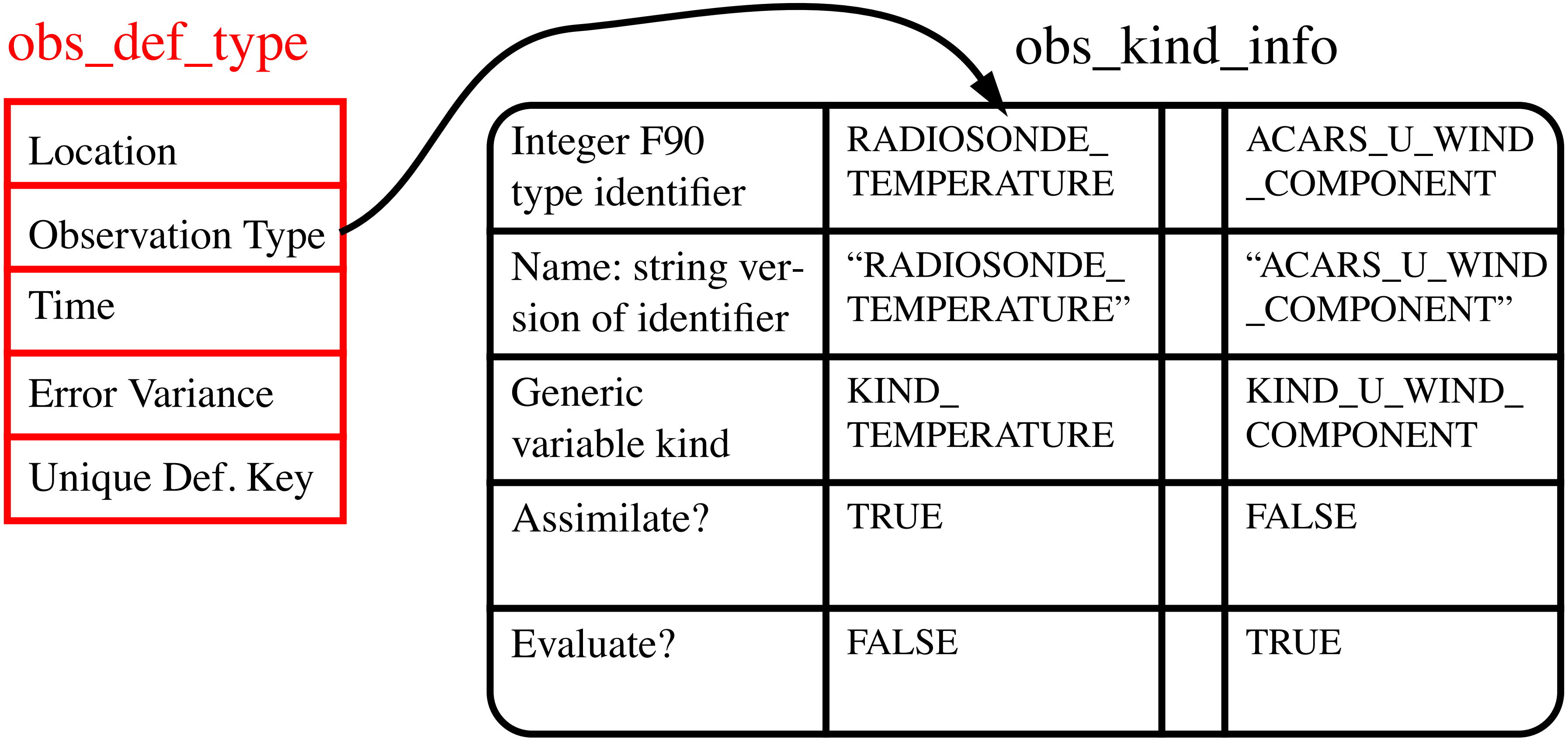
Location type required for model's domain (1d, 3d_sphere,...)

Integer index into obs_kind_info table in obs_kind_mod

time_type

This is NOT the same key as in observation type.

Observation Definition Details



obs_def_type

- Location
- Observation Type
- Time
- Error Variance
- Unique Def. Key

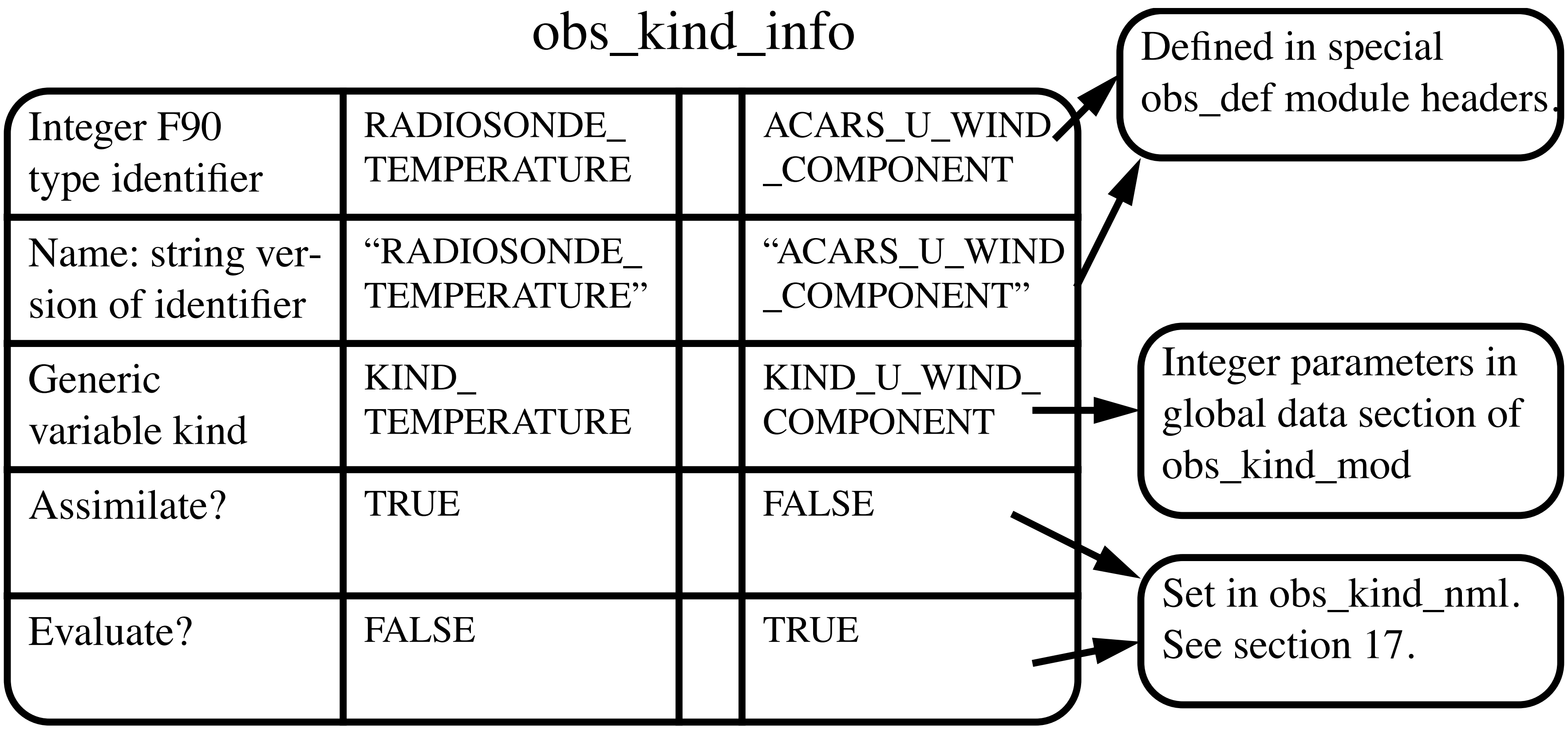
obs_kind_info

Integer F90 type identifier	RADIOSONDE_ TEMPERATURE		ACARS_U_WIND _COMPONENT
Name: string version of identifier	“RADIOSONDE_ TEMPERATURE”		“ACARS_U_WIND _COMPONENT”
Generic variable kind	KIND_ TEMPERATURE		KIND_U_WIND_ COMPONENT
Assimilate?	TRUE		FALSE
Evaluate?	FALSE		TRUE

Example: Observation is a radiosonde temperature

Observation Generic Kinds and Specific Types

obs_kind_info table built by DART preprocess program



obs_kind_info

Integer F90 type identifier	RADIOSONDE_ TEMPERATURE		ACARS_U_WIND _COMPONENT
Name: string version of identifier	“RADIOSONDE_ TEMPERATURE”		“ACARS_U_WIND _COMPONENT”
Generic variable kind	KIND_ TEMPERATURE		KIND_U_WIND_ COMPONENT
Assimilate?	TRUE		FALSE
Evaluate?	FALSE		TRUE

Defined in special obs_def module headers.

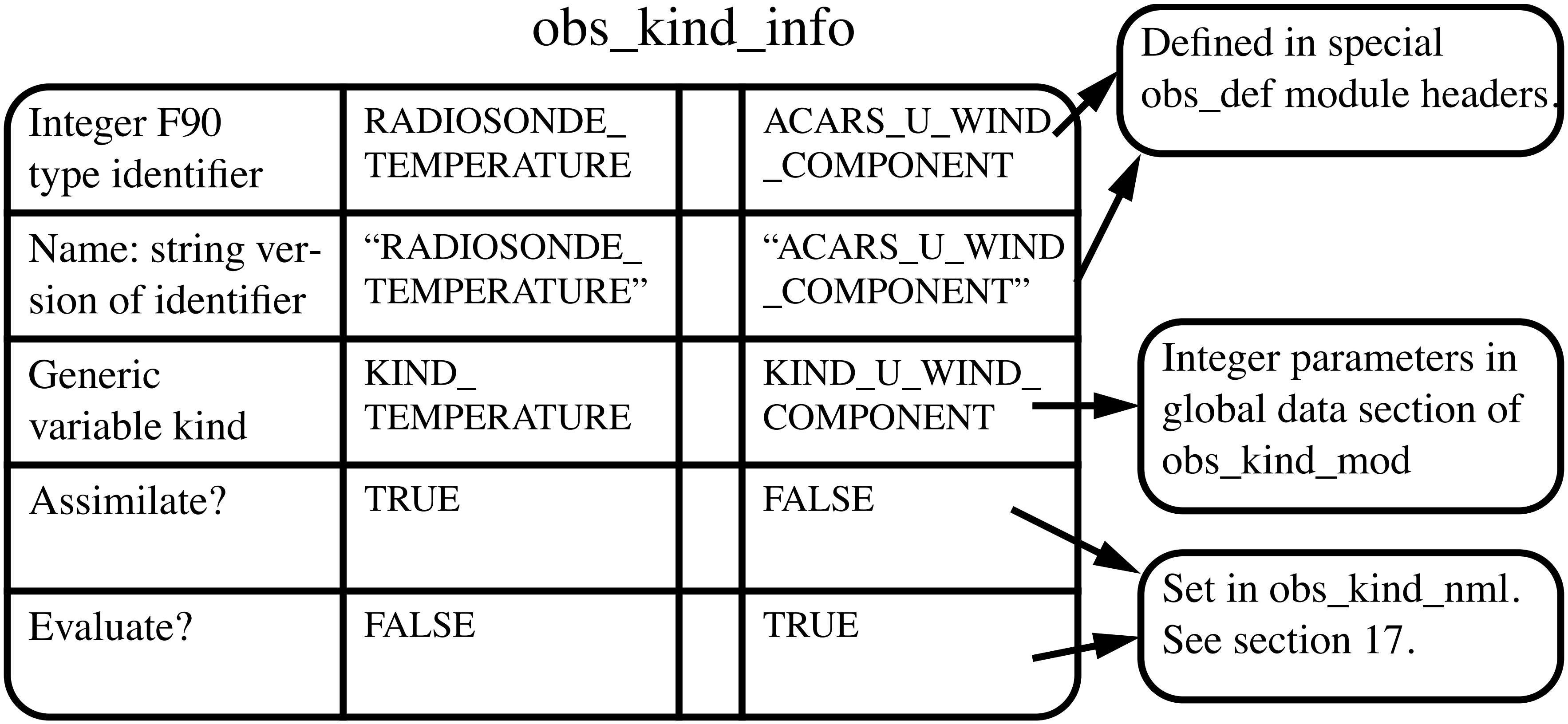
Integer parameters in global data section of obs_kind_mod

Set in obs_kind_nml. See section 17.

Radiosonde temps assimilated, forward operators only for ACARS U

Observation Generic Kinds and Specific Types

Many observation types may share a generic kind.
Example: RADIOSONDE_TEMPERATURE, ACARS_TEMPERATURE...



obs_kind_info

Integer F90 type identifier	RADIOSONDE_ TEMPERATURE		ACARS_U_WIND _COMPONENT
Name: string version of identifier	"RADIOSONDE_ TEMPERATURE"		"ACARS_U_WIND _COMPONENT"
Generic variable kind	KIND_ TEMPERATURE		KIND_U_WIND_ COMPONENT
Assimilate?	TRUE		FALSE
Evaluate?	FALSE		TRUE

Defined in special obs_def module headers.

Integer parameters in global data section of obs_kind_mod

Set in obs_kind_nml. See section 17.

Both have generic KIND_TEMPERATURE.
Model state variables can also be associated with generic kinds.

Observation Generic Kinds and Specific Types

Many observation types may share a generic kind

Example: RADIOSONDE_TEMPERATURE, ACARS_TEMPERATURE

Both have generic KIND_TEMPERATURE.

Model state variables are also associated with generic kinds

Example: CAM/WRF interpolate in T field for all observation types with generic kind KIND_TEMPERATURE.

Models can use the obs_kind_mod:

Have access to all generic kinds.

Also have access to all observation types if needed.

CONFUSING generic kinds and specific observation types is common.

Implementing Observation Definitions in DART

In an *obs_def/obs_def_xxx_mod.f90* file:

1. Give the observation specific type a name. This is where the name is defined.
2. Associate the observation specific type with a generic kind, which must already exist in the DART KIND_xxx list.
3. Optionally specify a keyword to autogenerate needed routines if no specialized handling or additional metadata.

Example:

```
! BEGIN DART PREPROCESS KIND LIST
! AIRS_TEMPERATURE,         KIND_TEMPERATURE,         COMMON_CODE
! AIRS_SPECIFIC_HUMIDITY,  KIND_SPECIFIC_HUMIDITY,  COMMON_CODE
! END DART PREPROCESS KIND LIST
```

If using the autogenerated routines no additional work is needed.

Implementing Observation Definitions in DART

If the forward operator requires additional code, or if this observation specific type has additional metadata, omit the COMMON_CODE keyword and supply additional routines:

Four operations must be supported for each observation type:

1. Compute forward operator given (extended) state vector
2. Read any extra information not in obs_def_type from file (For instance, location and beam angle for radar).
3. Write any extra information not in obs_def_type to file
4. Get any extra information via interactive read of standard in

If additional metadata, suggest two additional routines:

1. get_metadata()
2. set_metadata()

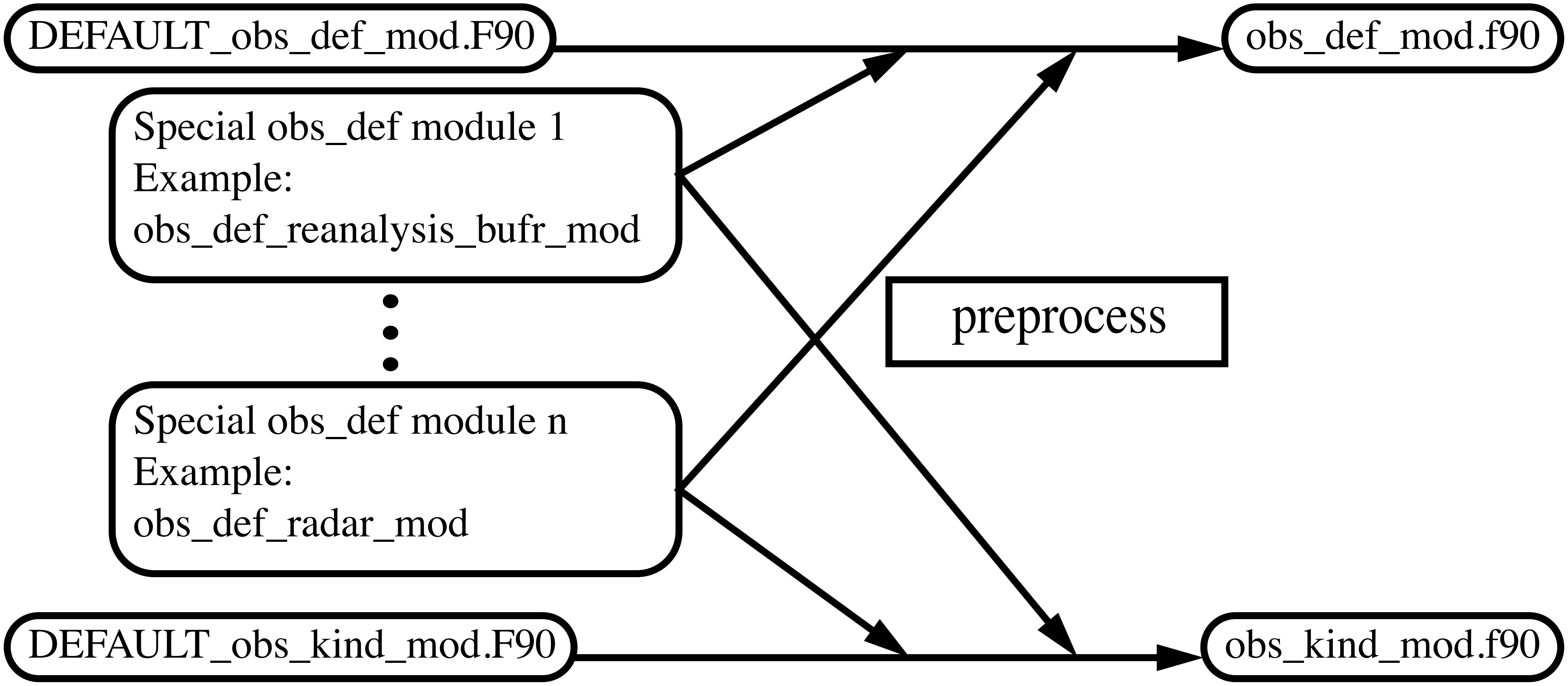
Implementing Observation Definitions in DART

obs_def_xxx_mod.f90 files and *DEFAULT_obs_def_mod.F90* are normal Fortran 90 files with additional specially formatted comments that guide the *preprocess* program.

See the detailed documentation in *obs_def/obs_def_mod.html* (also *obs_kind/obs_kind_mod.html*)

Implementing Observation Definitions in DART

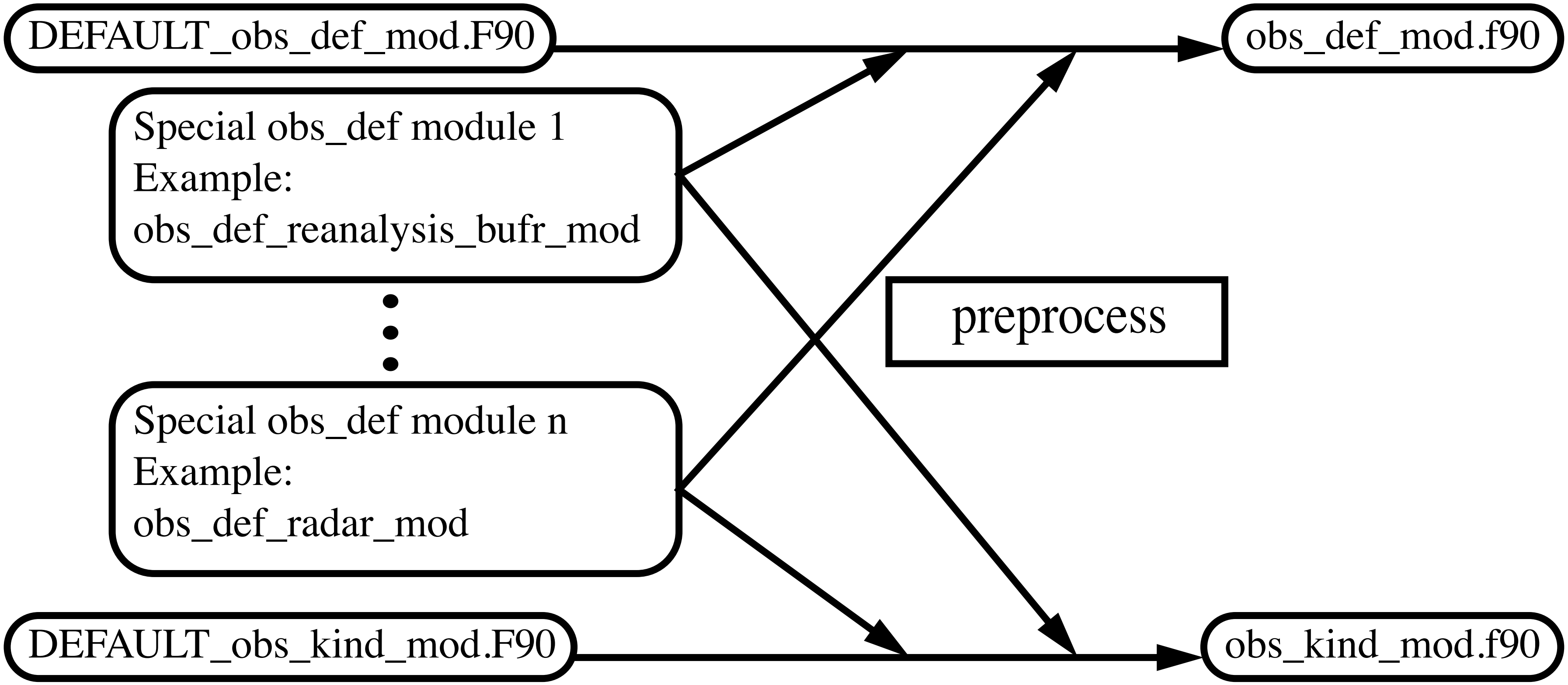
DART preprocess program creates obs_def_mod, obs_kind_mod



Namelist *&preprocess_nml* lists all special obs_def modules to be used.
(Names of DEFAULT F90s and preprocessed f90s can be changed, too)

Implementing Observation Definitions in DART

DART preprocess program creates obs_def_mod, obs_kind_mod



If no special obs_def modules are selected, can do identity obs. only.
DEFAULT modules have special comment lines to help preprocess.

Implementing Basic Observation Definitions in DART

Basic: New observation type with no specialized forward operator code and no extra observation information.

Will call the model interpolate routine to compute the forward operator for each observation type listed.
Needs no extra info in the read/write or interactive create routines.

Requires adding 1 section to one or more obs_def_mod files.

Defines the mapping between each specific observation type and generic observation kind, plus a keyword.

A REQUIRED comment string starts and ends the section.
All lines in the special section must start with F90 comment, !

Implementing Basic Observation Definitions in DART

Define the observation types and associated generic kinds:

```
! BEGIN DART PREPROCESS KIND LIST
! RAW_STATE_VARIABLE, KIND_RAW_STATE_VARIABLE, COMMON_CODE
! END DART PREPROCESS KIND LIST
```

First column is specific type, second is generic kind.

The keyword COMMON_CODE tells DART to automatically generate all required interface code for this new type.

Multiple types can be defined between the special comment lines.

This is all the file needs to contain.

The list of generic kinds is found in DEFAULT_obs_kind_mod.F90.

If not already there, the generic kind must be added to the list.

See obs_def_AIRS_mod.f90 for another example.

Implementing Customized Observation Definitions in DART

Customized: Either the observation type cannot simply be interpolated in a model state vector, and/or there is extra information associated with each observation which must be read, written, and interactively prompted for when creating new observations of this type

Basic observations require only 1 section in the specialized obs_def. Customized ones require 6.

Can have mix of Basic observations (with autogenerated code) and Customized observations (with user-supplied code) in the same file.

REQUIRED comment strings start and end each section.
All lines in special sections must start with F90 comment, !
See obs_def_1d_state_mod.f90 as an example.

Implementing Customized Observation Definitions in DART

Six special sections are required in a special obs_def_mod.

1. Define the observation types and associated generic kinds:

```
! BEGIN DART PREPROCESS KIND LIST
! RAW_STATE_VARIABLE, KIND_RAW_STATE_VARIABLE, COMMON_CODE
! RAW_STATE_1D_INTEGRAL, KIND_1D_INTEGRAL
! END DART PREPROCESS KIND LIST
```

Two observation types defined:

a. RAW_STATE_VARIABLE: generic kind KIND_RAW_STATE_VARIABLE
   All interface code autogenerated by DART
b. RAW_STATE_1D_INTEGRAL: generic kind KIND_1D_INTEGRAL
   User must supply 4 additional interfaces.
   Even if nothing to do, must supply a case statement for each

Implementing Customized Observation Definitions in DART

Six special sections are required in a special obs_def_mod.

2. Use statements required for use of obs_def_1d_state_mod

```
! BEGIN DART PREPROCESS USE OF SPECIAL OBS_DEF MODULE
! ! Comments can be included by having a second ! at
! the start of the line
! use obs_def_1d_state_mod, only : write_1d_integral, &
!            read_1d_integral, interactive_1d_integral, &
!            get_expected_1d_integral
! END DART PREPROCESS USE OF SPECIAL OBS_DEF MODULE
```

This special obs_def module has 4 subroutines which do work.

A special obs_def module can also have its own namelist if needed.

Implementing Customized Observation Definitions in DART

Six special sections are required in a special obs_def_mod.

3. Case statements required to compute expected observation

```
! BEGIN DART PREPROCESS GET_EXPECTED_OBS_FROM_DEF
!   case(RAW_STATE_1D_INTEGRAL)
!       call get_expected_1d_integral(state, location, &
!                                     obs_def%key, obs_val, istatus)
! END DART PREPROCESS GET_EXPECTED_OBS_FROM_DEF
```

Each observation type being defined that does not have the COMMON_CODE keyword must appear in a case.

The autogenerated code calls *interpolate()* from assim_model.
The RAW_STATE_1D_INTEGRAL is more complicated and calls the *get_expected_1d_integral* in the special obs_def module.

Implementing Customized Observation Definitions in DART

Six special sections are required in a special obs_def_mod.

4. Case statements read extra info from an obs_sequence file.

```
! BEGIN DART PREPROCESS READ_OBS_DEF
!   case(RAW_STATE_1D_INTEGRAL)
!       call read_1d_integral(obs_def%key, ifile, fileformat)
! END DART PREPROCESS READ_OBS_DEF
```

The autogenerated code has a case statement and continue.

RAW_STATE_1D_INTEGRAL requires extra information.

This is read with read_1d_integral subroutine.

Extra info stored in obs_def_1d_state_mod, indexed by unique DEFINITION key.

All obs types must have a case statement, even if no extra info.

Implementing Customized Observation Definitions in DART

Six special sections are required in a special obs_def_mod.

5. Case statements write extra info to an obs_sequence file.

```
! BEGIN DART PREPROCESS WRITE_OBS_DEF
!   case(RAW_STATE_1D_INTEGRAL)
!       call write_1d_integral(obs_def%key, ifile, fileformat)
! END DART PREPROCESS WRITE_OBS_DEF
```

Same deal as for read

obs_def_1d_state can read and write whatever it wants
to describe the raw_state_1d_integral observation.

Only requirement is that it can read what it writes!

Implementing Customized Observation Definitions in DART

Six special sections are required in a special obs_def_mod.

6. Case statements to interactively create extra info.

```
! BEGIN DART PREPROCESS INTERACTIVE_OBS_DEF
!   case(RAW_STATE_1D_INTEGRAL)
!       call interactive_1d_integral(obs_def%key, ifile, fileformat)
! END DART PREPROCESS INTERACTIVE_OBS_DEF
```

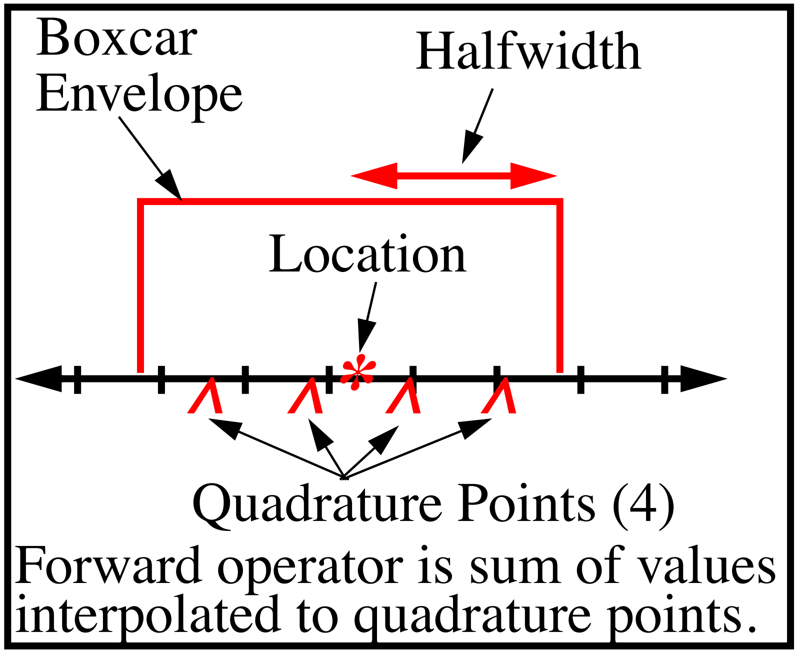
DART uses interactive input from standard in to create type-specific information in a user-extensible form.

It’s nice to be able to do a keyboard create for testing

Standard procedure: construct a text file that drives creation (see section 17)

Implementing Customized Observation Definitions in DART

What is the observation definition ‘extra information’?
obs_def_1d_state_mod example.



Boxcar Envelope
Halfwidth
Location
*
Quadrature Points (4)
Forward operator is sum of values interpolated to quadrature points.

raw_state_1d integral forward operator has 3 parameters:

1. Half-width of envelope,
2. Shape of envelope,
3. Number of quadrature pts.

Interactive creation asks for these 3, stores them with definition key.

Additional values written with each obs separately.

Available obs_def modules in DART

obs_def_1d_state_mod.f90
obs_def_AIRS_mod.f90
obs_def_AOD_mod.f90
obs_def_AURA_mod.f90
obs_def_COSMOS_mod.f90
obs_def_GWD_mod.f90
obs_def_QuikSCAT_mod.f90
obs_def_SABER_mod.f90
obs_def_TES_nadir_mod.f90
obs_def_altimeter_mod.f90
obs_def_cloud_mod.f90
obs_def_cwp_mod.f90
obs_def_dew_point_mod.f90
obs_def_dwl_mod.f90
obs_def_eval_mod.f90
obs_def_goes_mod.f90

obs_def_gps_mod.f90
obs_def_gts_mod.f90
obs_def_metar_mod.f90
obs_def_ocean_mod.f90
obs_def_pe2lyr_mod.f90
obs_def_radar_mod.f90
obs_def_reanalysis_bufr_mod.f90
obs_def_rel_humidity_mod.f90
obs_def_simple_advection_mod.f90
obs_def_sqg_mod.f90
obs_def_tower_mod.f90
obs_def_tpw_mod.f90
obs_def_upper_atm_mod.f90
obs_def_vortex_mod.f90
obs_def_wind_speed_mod.f90

Available obs_def modules in DART

Examples of frequently used obs_def modules in large models:

obs_def_reanalysis_bufr_mod.f90
    Defines all obs likely to be found in BUFR files.

obs_def_ocean_mod.f90
    All obs types from the World Ocean Database

obs_def_radar_mod.f90
    Forward operator code for reflectivity and radial velocity

obs_def_gps_mod.f90
    Simple and integrated forward operators for refractivity obs

obs_def_tower_mod.f90
    Land obs types and forward operators

Using Custom Observation Definitions in DART

1. Compile and run preprocess: specify absolute or relative paths for all required special obs_def modules in *preprocess_nml, input_files*.

2. Compile all other required program units, including obs_def_mod.f90 (only) in the *path_names_file* files. preprocess will add any specialized obs_def code to the obs_def_mod.f90 source file.

3. Select observation types to be assimilated or evaluated in &*obs_kind_nml*.

How and Where to Compute Forward Operators

Keeping models and observation definitions modular is hard.

DART recommendation: models should be able to spatially interpolate their state variables.

Forward observation operators in special obs_def modules should not expect more than this from models.

This may be too idealistic:

1. Models could do complicated forward operators for efficiency.
2. This makes it difficult to link models to DART in F90.

Different version of assim_model could help to buffer this.
Area for ongoing research.

DART Tutorial Index to Sections