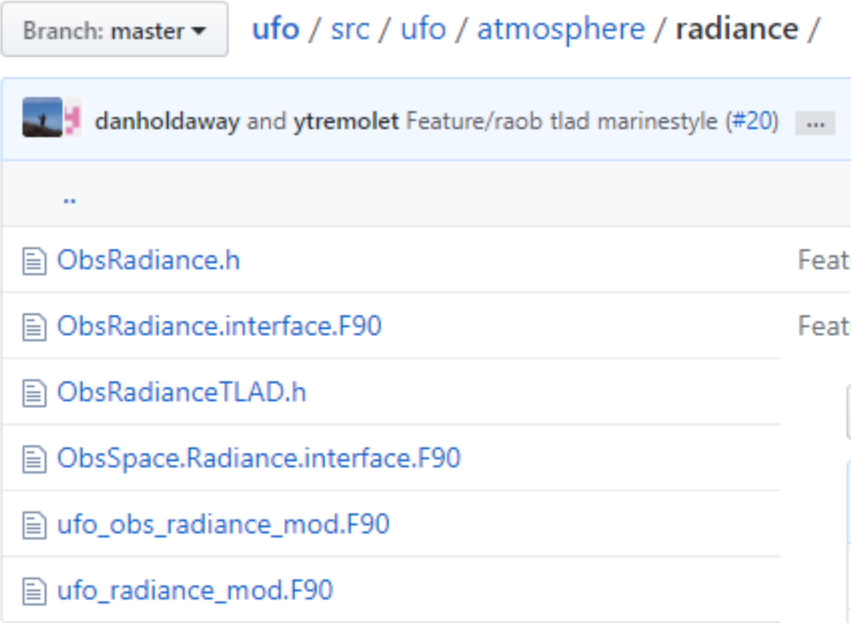
ObsSpace Reorganization

Xin Zhang

JEDI Core Team

5/3/18

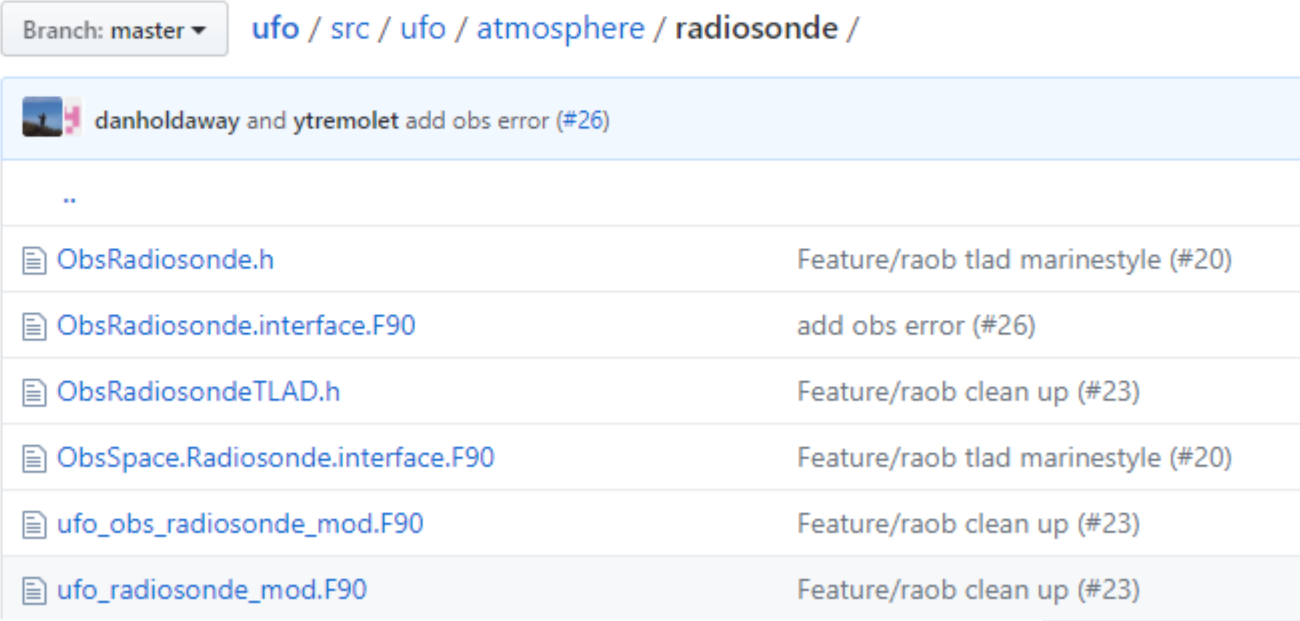
UFO: Current code



Branch: master ▾ ufo / src / ufo / atmosphere / radiance /

danholdaway and ytremolet Feature/raob tlad marinestyle (#20) …

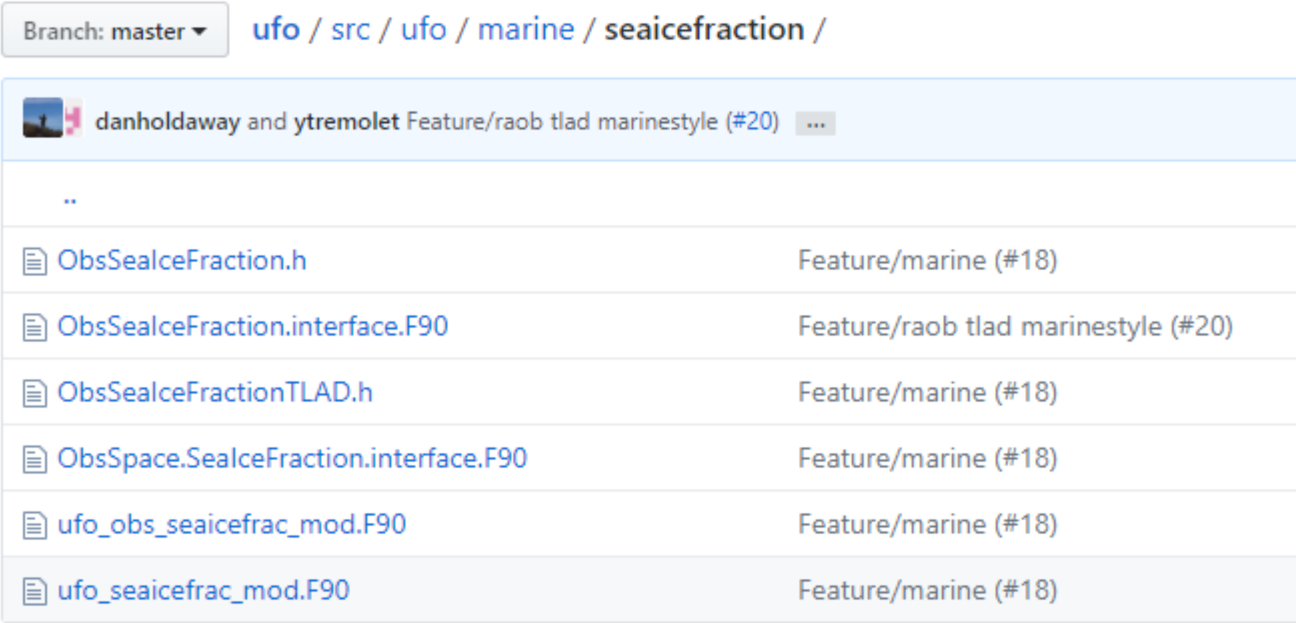
..	
ObsRadiance.h	Feature/raob tlad mai
ObsRadiance.interface.F90	Feature/raob tlad mai
ObsRadianceTLAD.h	
ObsSpace.Radiance.interface.F90	
ufo_obs_radiance_mod.F90	
ufo_radiance_mod.F90	



Branch: master ▾ ufo / src / ufo / atmosphere / radiosonde /

danholdaway and ytremolet add obs error (#26)

..	
ObsRadiosonde.h	Feature/raob tlad marinestyle (#20)
ObsRadiosonde.interface.F90	add obs error (#26)
ObsRadiosondeTLAD.h	Feature/raob clean up (#23)
ObsSpace.Radiosonde.interface.F90	Feature/raob tlad marinestyle (#20)
ufo_obs_radiosonde_mod.F90	Feature/raob clean up (#23)
ufo_radiosonde_mod.F90	Feature/raob clean up (#23)



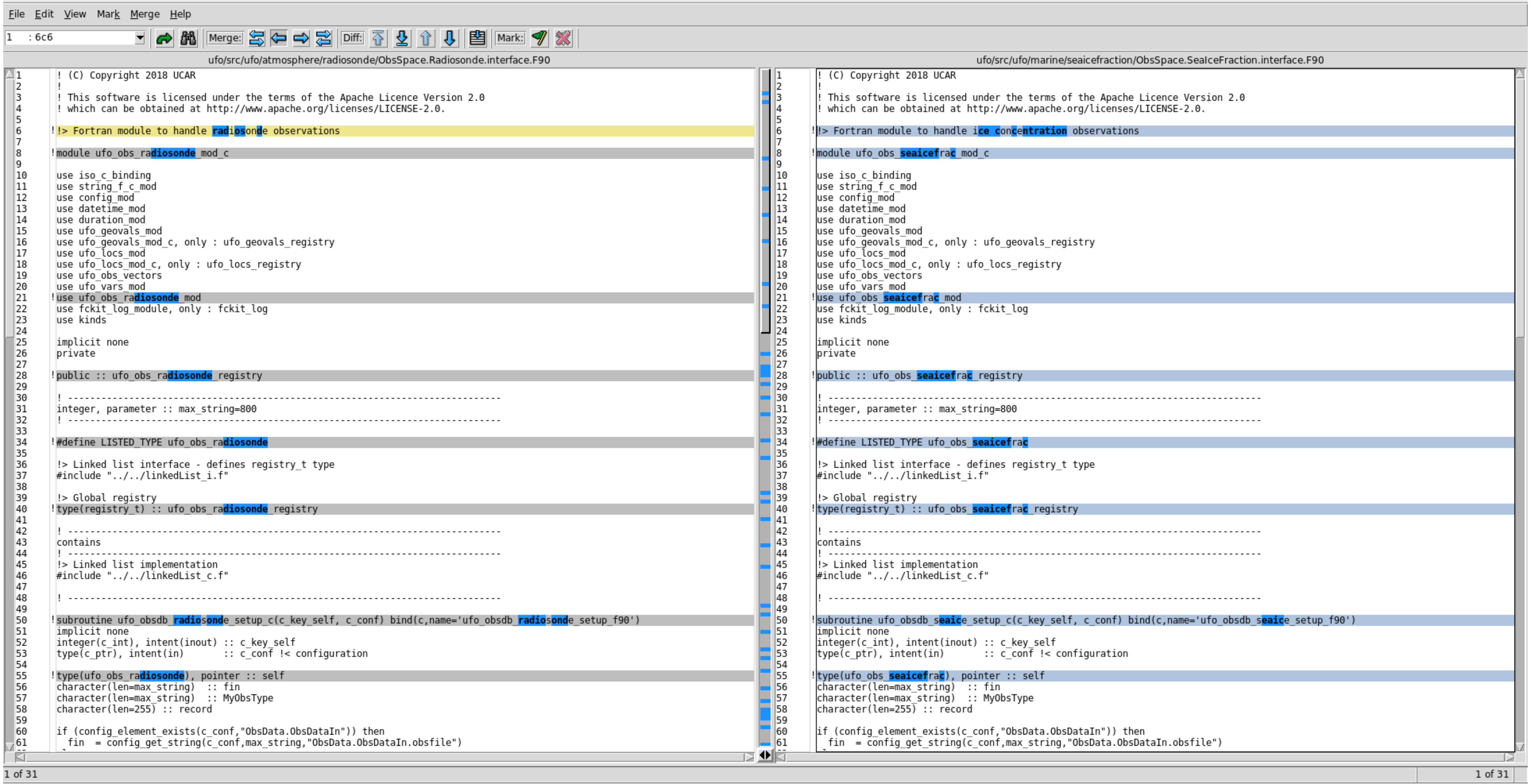
Branch: master ▾ ufo / src / ufo / marine / seaicefraction /

danholdaway and ytremolet Feature/raob tlad marinestyle (#20) …

..	
ObsSeaIceFraction.h	Feature/marine (#18)
ObsSeaIceFraction.interface.F90	Feature/raob tlad marinestyle (#20)
ObsSeaIceFractionTLAD.h	Feature/marine (#18)
ObsSpace.SeaIceFraction.interface.F90	Feature/marine (#18)
ufo_obs_seaicefrac_mod.F90	Feature/marine (#18)
ufo_seaicefrac_mod.F90	Feature/marine (#18)

Similar structures

Almost identical codes



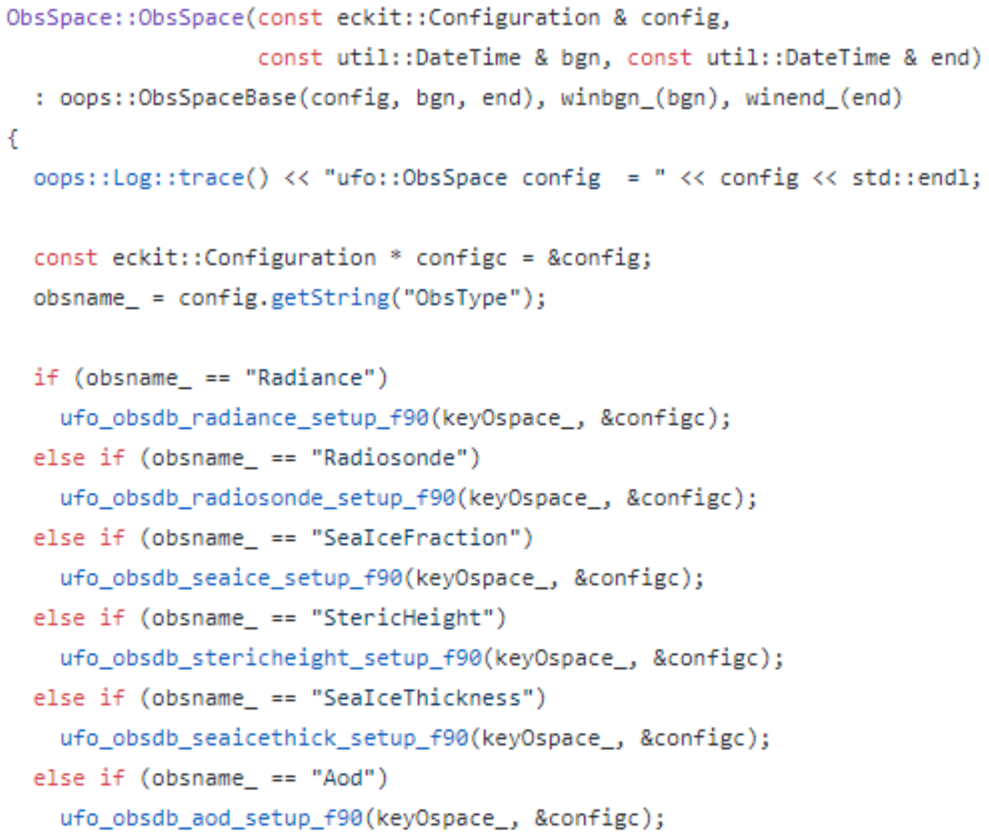
ufo/src/ufo/atmosphere/radiosonde/ObsSpace.Radiosonde.interface.F90

```
! (C) Copyright 2018 UCAR
!
! This software is licensed under the terms of the Apache Licence Version 2.0
! which can be obtained at http://www.apache.org/licenses/LICENSE-2.0.

!> Fortran module to handle radiosonde observations

module ufo_obs_radiosonde_mod_c

use iso_c_binding
use string_f_c_mod
use config_mod
use datetime_mod
use duration_mod
use ufo_geovals_mod
use ufo_geovals_mod_c, only : ufo_geovals_registry
use ufo_locs_mod
use ufo_locs_mod_c, only : ufo_locs_registry
use ufo_obs_vectors
use ufo_vars_mod
use ufo_obs_radiosonde_mod
use fckit_log_module, only : fckit_log
use kinds

implicit none
private

public :: ufo_obs_radiosonde_registry

! ------------------------------------------------------------------------------
integer, parameter :: max_string=800
! ------------------------------------------------------------------------------

#define LISTED_TYPE ufo_obs_radiosonde

!> Linked list interface - defines registry_t type
#include "../../linkedList_i.f"

!> Global registry
type(registry_t) :: ufo_obs_radiosonde_registry

! ------------------------------------------------------------------------------
contains
! ------------------------------------------------------------------------------
!> Linked list implementation
#include "../../linkedList_c.f"

! ------------------------------------------------------------------------------

subroutine ufo_obsdb_radiosonde_setup_c(c_key_self, c_conf) bind(c,name='ufo_obsdb_radiosonde_setup_f90')
implicit none
integer(c_int), intent(inout) :: c_key_self
type(c_ptr), intent(in)       :: c_conf !< configuration

type(ufo_obs_radiosonde), pointer :: self
character(len=max_string)  :: fin
character(len=max_string)  :: MyObsType
character(len=255) :: record

if (config_element_exists(c_conf,"ObsData.ObsDataIn")) then
  fin  = config_get_string(c_conf,max_string,"ObsData.ObsDataIn.obsfile")
```

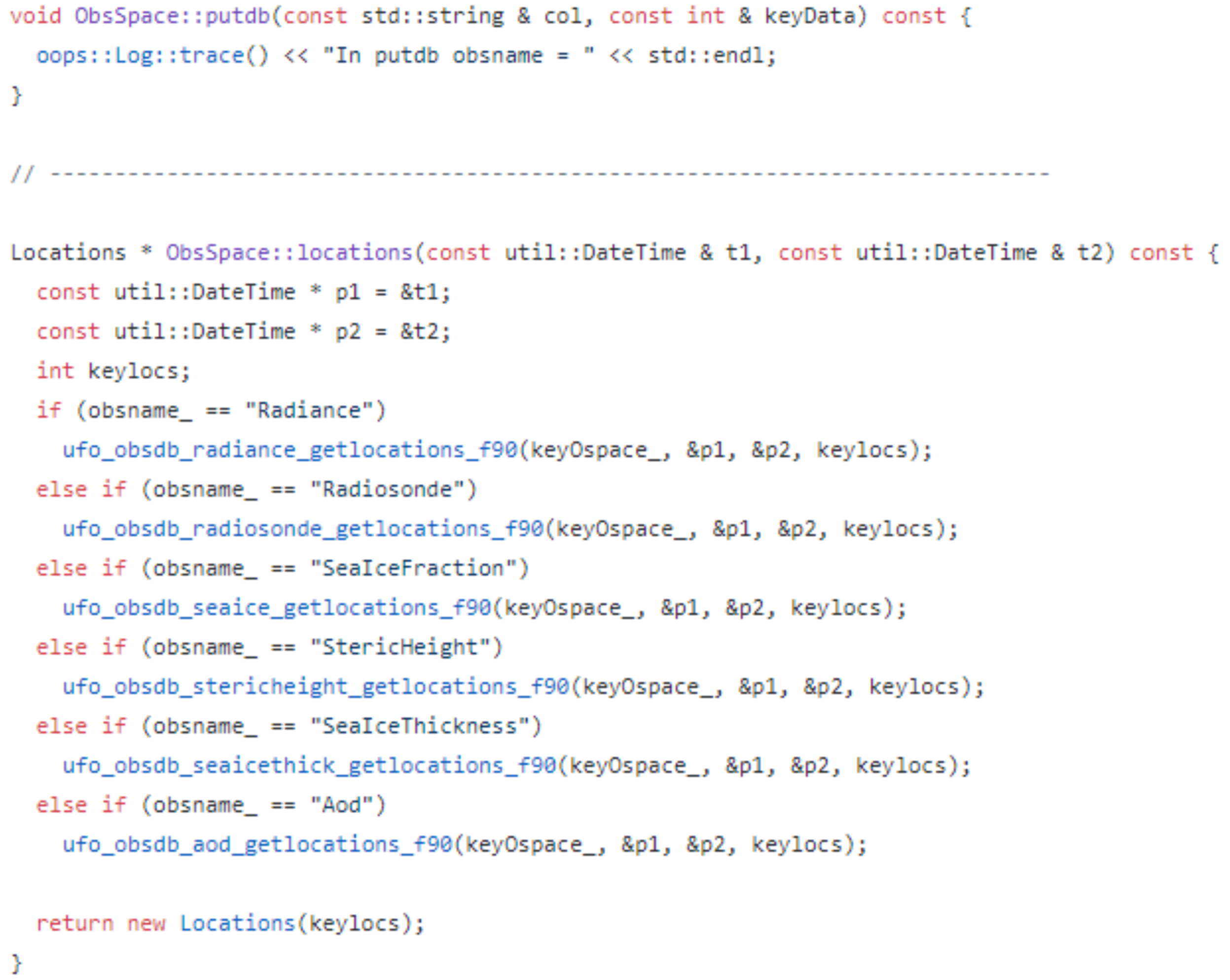
ufo/src/ufo/marine/seaicefraction/ObsSpace.SeaIceFraction.interface.F90

```
! (C) Copyright 2018 UCAR
!
! This software is licensed under the terms of the Apache Licence Version 2.0
! which can be obtained at http://www.apache.org/licenses/LICENSE-2.0.

!> Fortran module to handle ice concentration observations

module ufo_obs_seaicefrac_mod_c

use iso_c_binding
use string_f_c_mod
use config_mod
use datetime_mod
use duration_mod
use ufo_geovals_mod
use ufo_geovals_mod_c, only : ufo_geovals_registry
use ufo_locs_mod
use ufo_locs_mod_c, only : ufo_locs_registry
use ufo_obs_vectors
use ufo_vars_mod
use ufo_obs_seaicefrac_mod
use fckit_log_module, only : fckit_log
use kinds

implicit none
private

public :: ufo_obs_seaicefrac_registry

! ------------------------------------------------------------------------------
integer, parameter :: max_string=800
! ------------------------------------------------------------------------------

#define LISTED_TYPE ufo_obs_seaicefrac

!> Linked list interface - defines registry_t type
#include "../../linkedList_i.f"

!> Global registry
type(registry_t) :: ufo_obs_seaicefrac_registry

! ------------------------------------------------------------------------------
contains
! ------------------------------------------------------------------------------
!> Linked list implementation
#include "../../linkedList_c.f"

! ------------------------------------------------------------------------------

subroutine ufo_obsdb_seaice_setup_c(c_key_self, c_conf) bind(c,name='ufo_obsdb_seaice_setup_f90')
implicit none
integer(c_int), intent(inout) :: c_key_self
type(c_ptr), intent(in)       :: c_conf !< configuration

type(ufo_obs_seaicefrac), pointer :: self
character(len=max_string)  :: fin
character(len=max_string)  :: MyObsType
character(len=255) :: record

if (config_element_exists(c_conf,"ObsData.ObsDataIn")) then
  fin  = config_get_string(c_conf,max_string,"ObsData.ObsDataIn.obsfile")
```

If-else structure, calling similar APIs

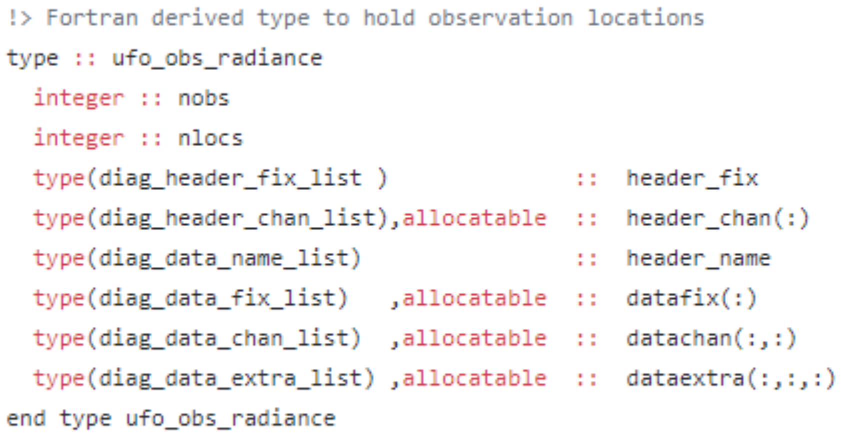


```
ObsSpace::ObsSpace(const eckit::Configuration & config,
                   const util::DateTime & bgn, const util::DateTime & end)
  : oops::ObsSpaceBase(config, bgn, end), winbgn_(bgn), winend_(end)
{
  oops::Log::trace() << "ufo::ObsSpace config  = " << config << std::endl;

  const eckit::Configuration * configc = &config;
  obsname_ = config.getString("ObsType");

  if (obsname_ == "Radiance")
    ufo_obsdb_radiance_setup_f90(keyOspace_, &configc);
  else if (obsname_ == "Radiosonde")
    ufo_obsdb_radiosonde_setup_f90(keyOspace_, &configc);
  else if (obsname_ == "SeaIceFraction")
    ufo_obsdb_seaice_setup_f90(keyOspace_, &configc);
  else if (obsname_ == "StericHeight")
    ufo_obsdb_stericheight_setup_f90(keyOspace_, &configc);
  else if (obsname_ == "SeaIceThickness")
    ufo_obsdb_seaicethick_setup_f90(keyOspace_, &configc);
  else if (obsname_ == "Aod")
    ufo_obsdb_aod_setup_f90(keyOspace_, &configc);
```

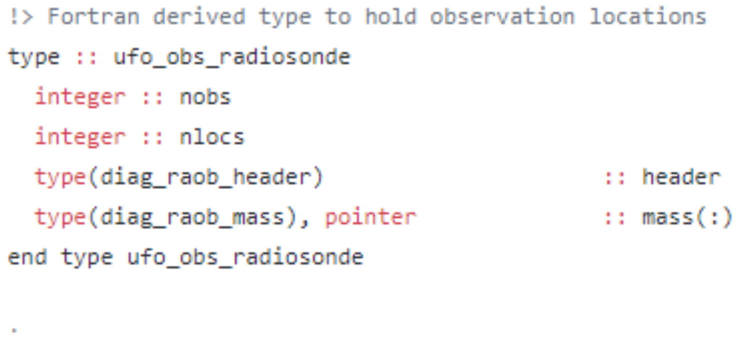


```
void ObsSpace::putdb(const std::string & col, const int & keyData) const {
  oops::Log::trace() << "In putdb obsname = " << std::endl;
}

// -----------------------------------------------------------------------------

Locations * ObsSpace::locations(const util::DateTime & t1, const util::DateTime & t2) const {
  const util::DateTime * p1 = &t1;
  const util::DateTime * p2 = &t2;
  int keylocs;
  if (obsname_ == "Radiance")
    ufo_obsdb_radiance_getlocations_f90(keyOspace_, &p1, &p2, keylocs);
  else if (obsname_ == "Radiosonde")
    ufo_obsdb_radiosonde_getlocations_f90(keyOspace_, &p1, &p2, keylocs);
  else if (obsname_ == "SeaIceFraction")
    ufo_obsdb_seaice_getlocations_f90(keyOspace_, &p1, &p2, keylocs);
  else if (obsname_ == "StericHeight")
    ufo_obsdb_stericheight_getlocations_f90(keyOspace_, &p1, &p2, keylocs);
  else if (obsname_ == "SeaIceThickness")
    ufo_obsdb_seaicethick_getlocations_f90(keyOspace_, &p1, &p2, keylocs);
  else if (obsname_ == "Aod")
    ufo_obsdb_aod_getlocations_f90(keyOspace_, &p1, &p2, keylocs);

  return new Locations(keylocs);
}
```

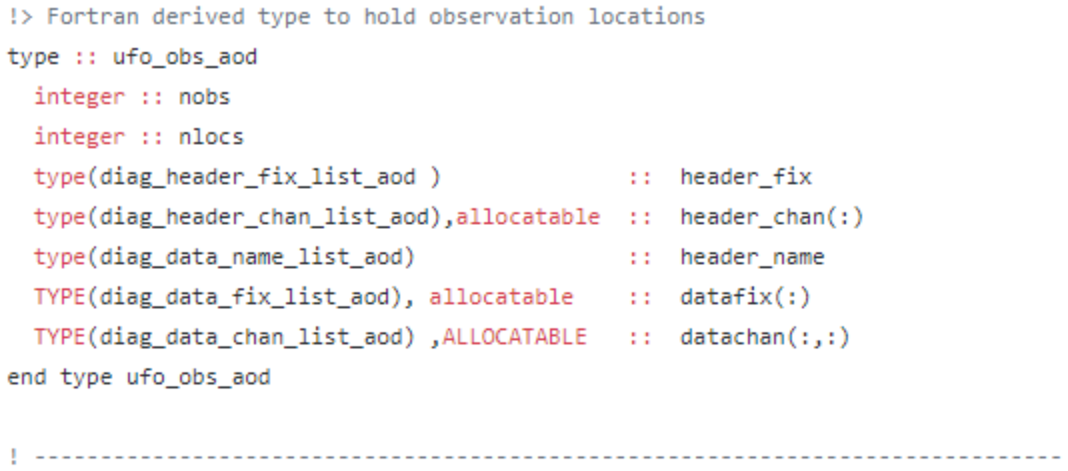
Similar UFO ObsSpace data structure



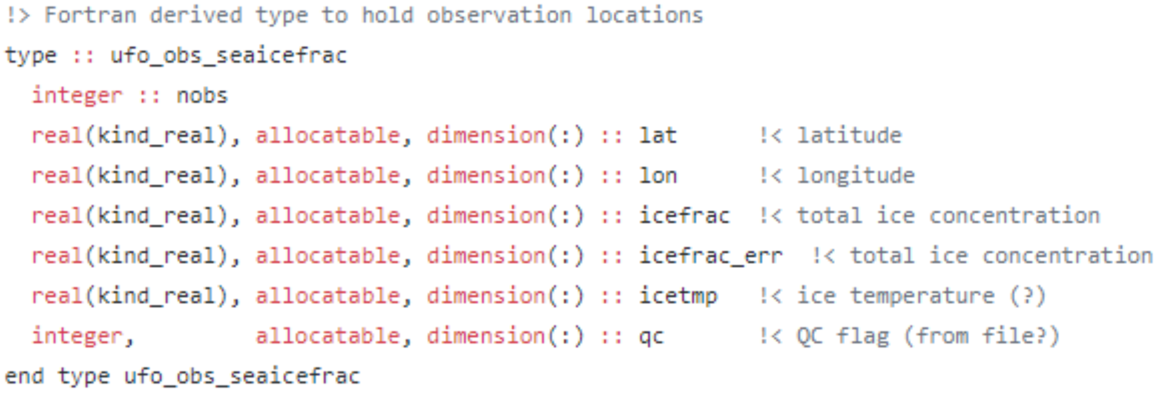
```
!> Fortran derived type to hold observation locations
type :: ufo_obs_radiance
  integer :: nobs
  integer :: nlocs
  type(diag_header_fix_list )             ::  header_fix
  type(diag_header_chan_list),allocatable ::  header_chan(:)
  type(diag_data_name_list)               ::  header_name
  type(diag_data_fix_list)   ,allocatable ::  datafix(:)
  type(diag_data_chan_list)  ,allocatable ::  datachan(:,:)
  type(diag_data_extra_list) ,allocatable ::  dataextra(:,:,:)
end type ufo_obs_radiance
```



```
!> Fortran derived type to hold observation locations
type :: ufo_obs_radiosonde
  integer :: nobs
  integer :: nlocs
  type(diag_raob_header)                     :: header
  type(diag_raob_mass), pointer              :: mass(:)
end type ufo_obs_radiosonde

.
```



```
!> Fortran derived type to hold observation locations
type :: ufo_obs_aod
  integer :: nobs
  integer :: nlocs
  type(diag_header_fix_list_aod )              ::  header_fix
  type(diag_header_chan_list_aod),allocatable  ::  header_chan(:)
  type(diag_data_name_list_aod)                ::  header_name
  TYPE(diag_data_fix_list_aod), allocatable    ::  datafix(:)
  TYPE(diag_data_chan_list_aod) ,ALLOCATABLE   ::  datachan(:,:)
end type ufo_obs_aod

! -----------------------------------------------------------------------------
```

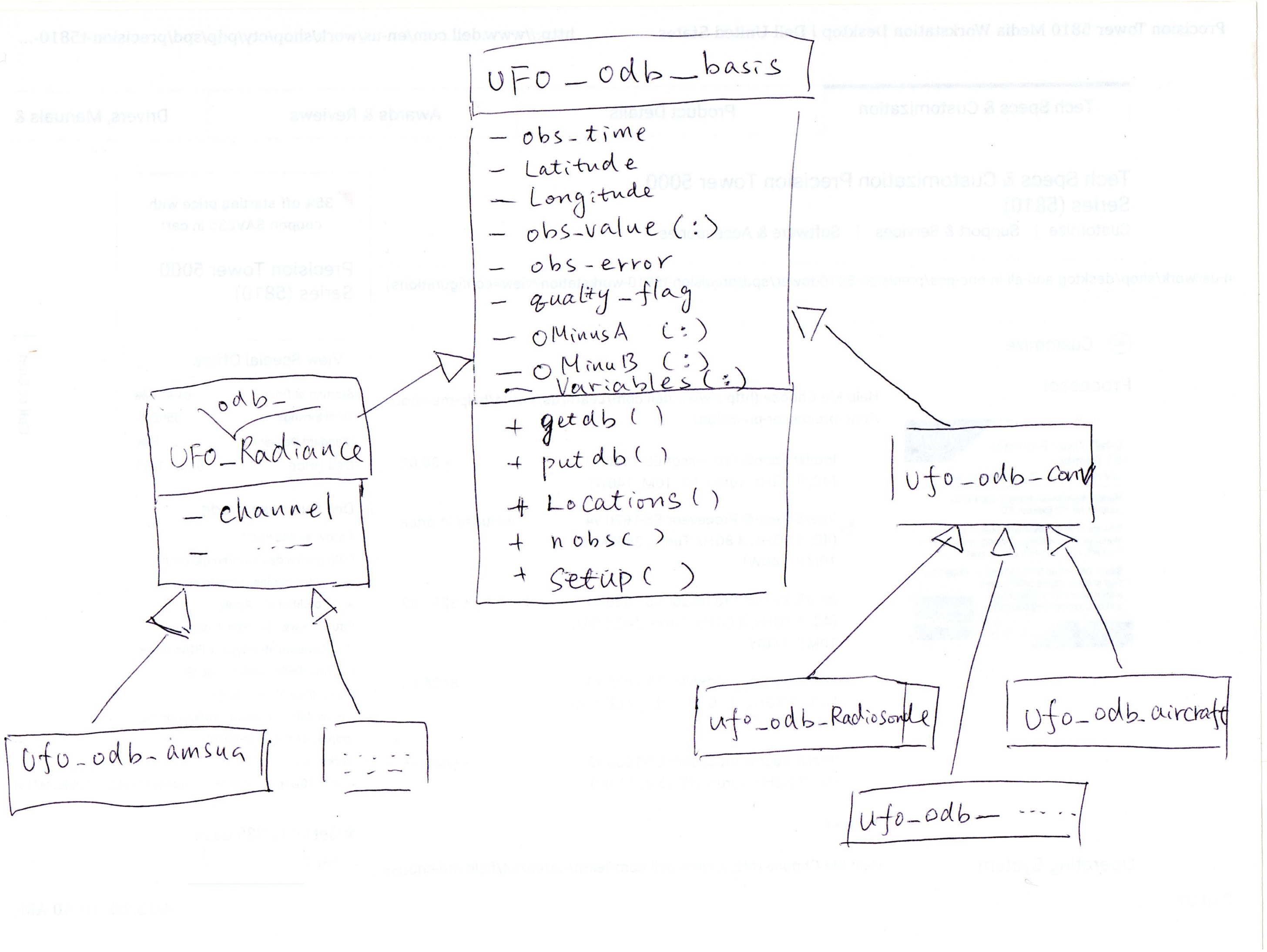


```
!> Fortran derived type to hold observation locations
type :: ufo_obs_seaicefrac
  integer :: nobs
  real(kind_real), allocatable, dimension(:) :: lat      !< latitude
  real(kind_real), allocatable, dimension(:) :: lon      !< longitude
  real(kind_real), allocatable, dimension(:) :: icefrac  !< total ice concentration
  real(kind_real), allocatable, dimension(:) :: icefrac_err !< total ice concentration
  real(kind_real), allocatable, dimension(:) :: icetmp   !< ice temperature (?)
  integer,         allocatable, dimension(:) :: qc       !< QC flag (from file?)
end type ufo_obs_seaicefrac
```

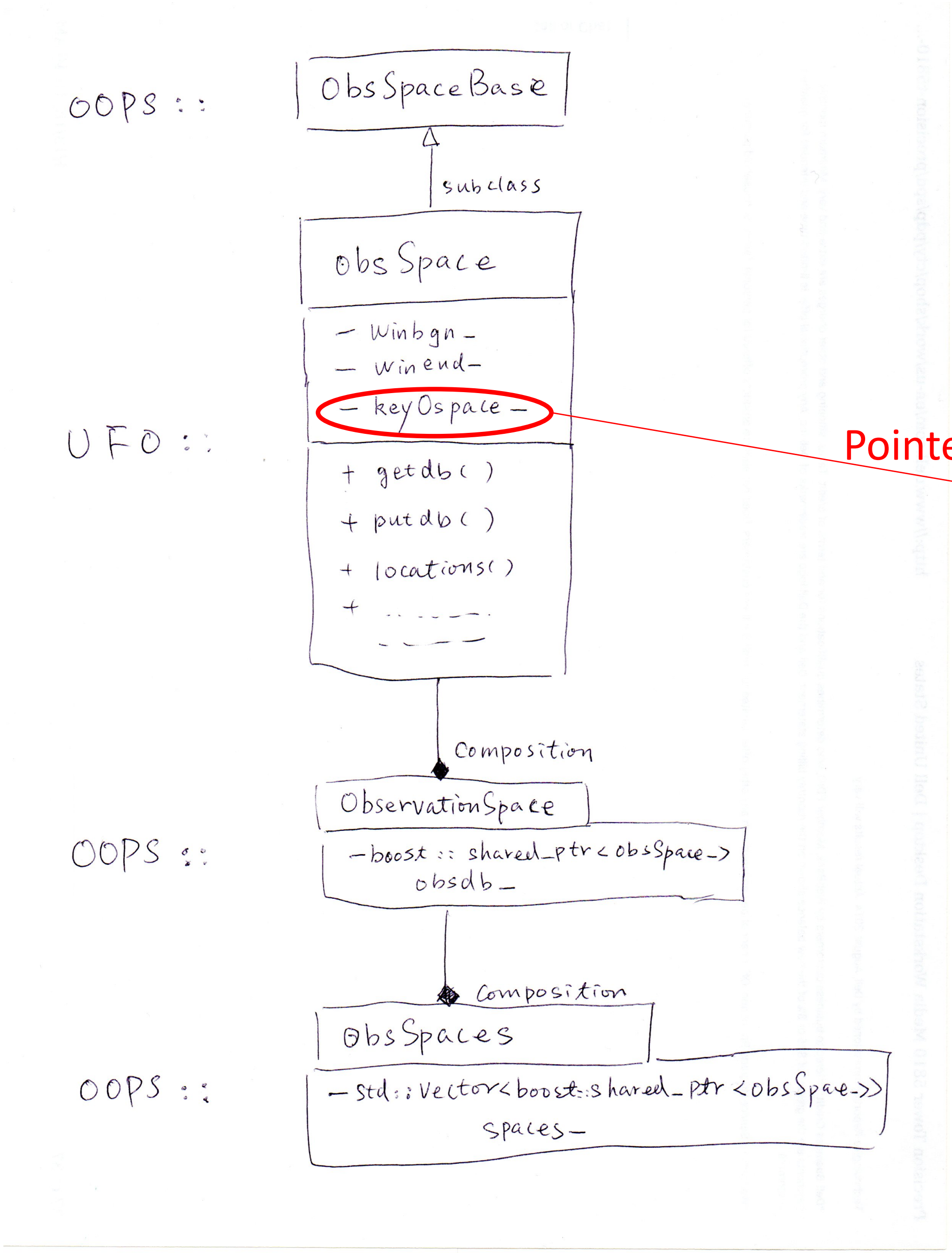
Motivation:

- Reduce the duplicated subroutines
- Simplify the APIs
- Re-design ObsSpace data structure

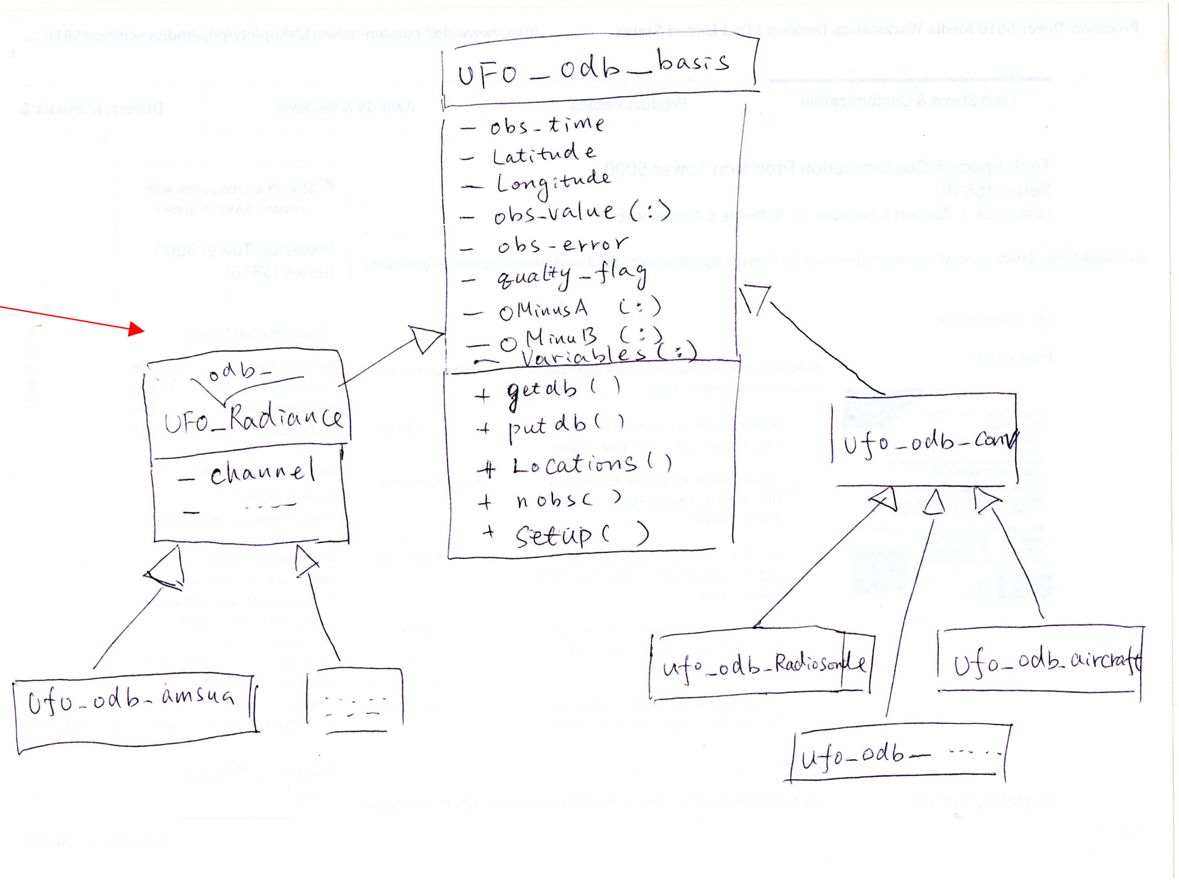
How: re-design the UFO ObsSpace data structure



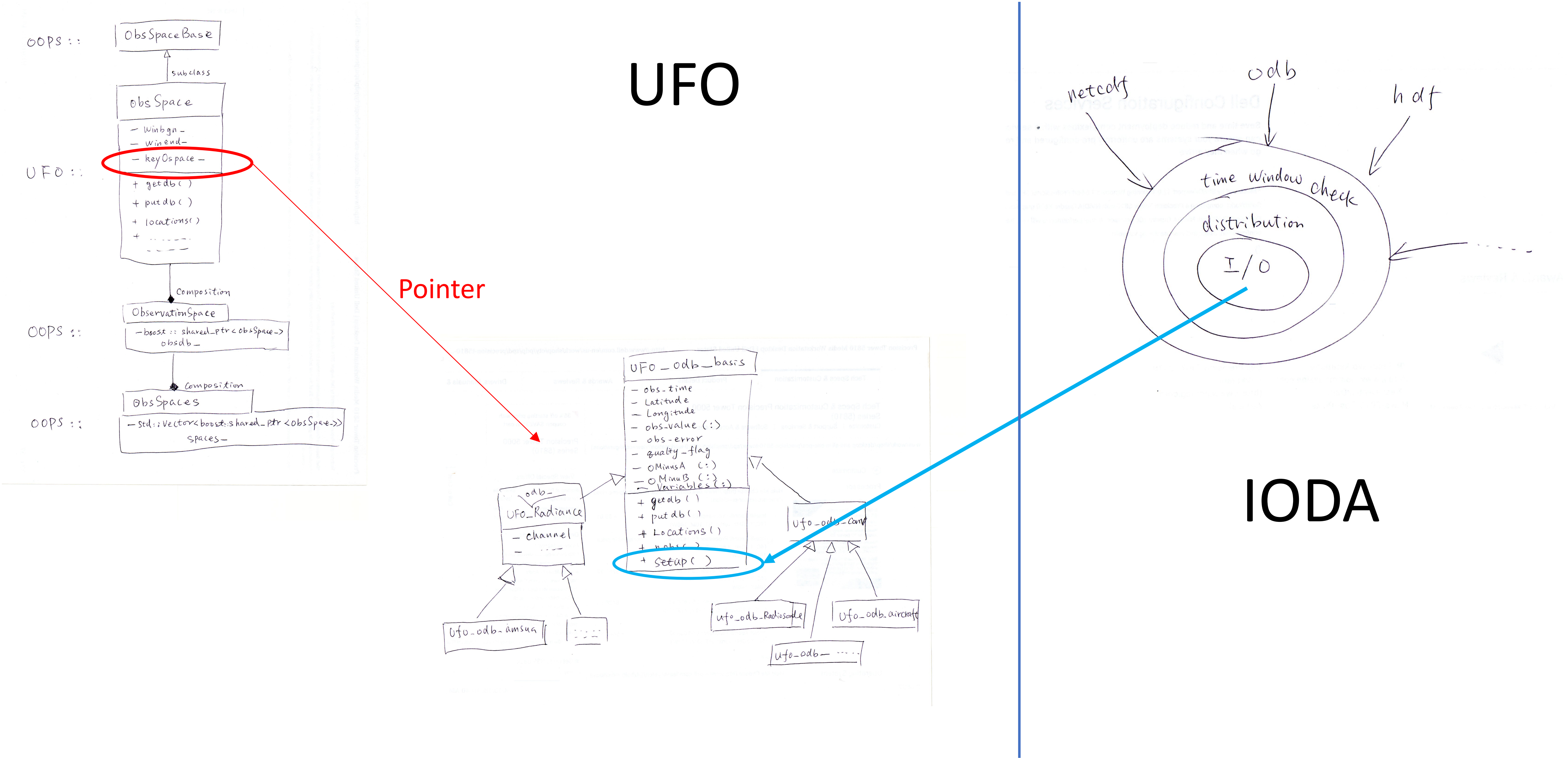
Observation Space



Pointer



Observation Space associated with IODA

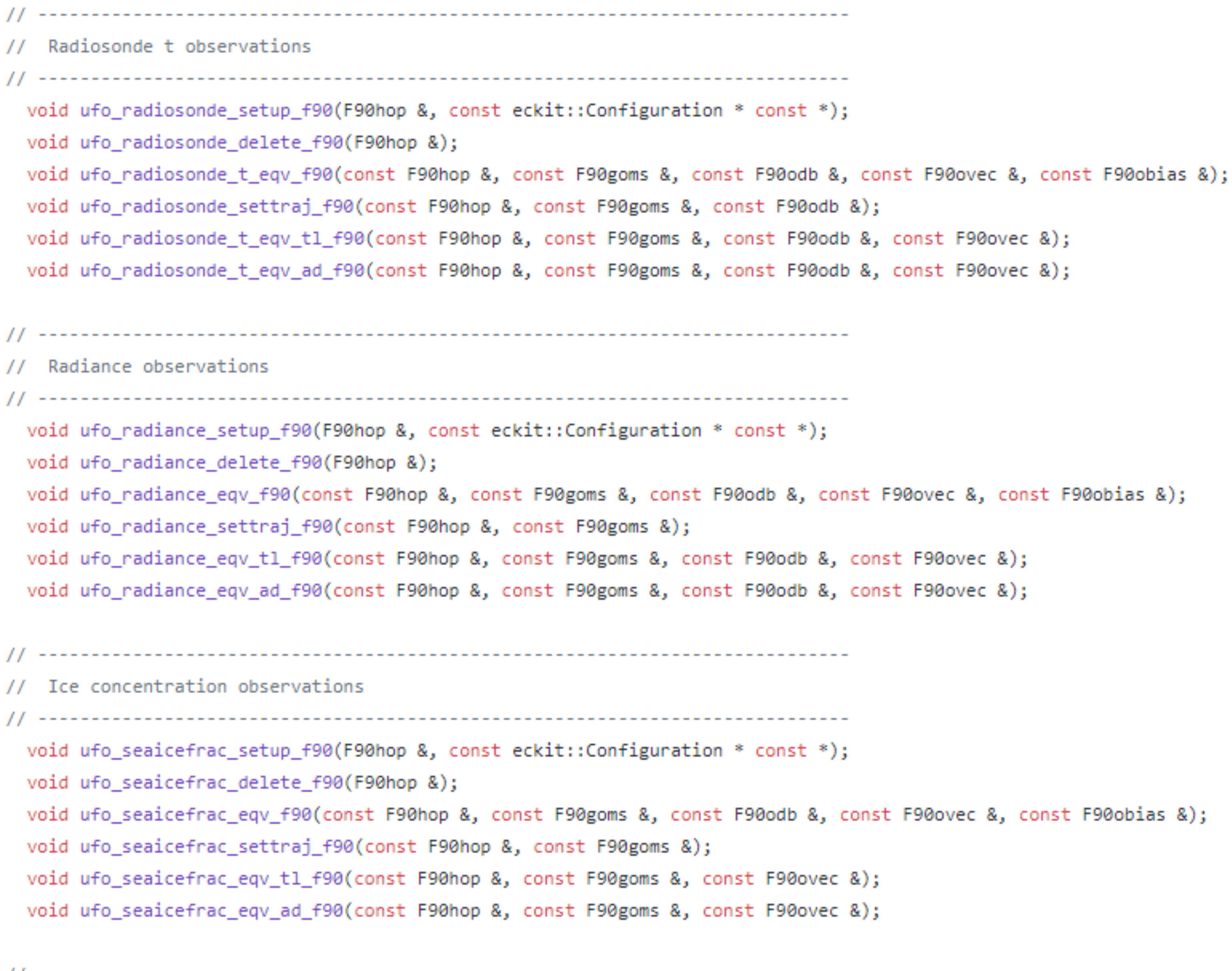

UFO

Pointer

IODA

Backup:

Obs Operator has similar issues; Although it already use the factory design pattern, but it does not leverage the inheritance approach, which can expose an unified interface from different family member.



```
// ----------------------------------------------------------------------------
//  Radiosonde t observations
// ----------------------------------------------------------------------------
  void ufo_radiosonde_setup_f90(F90hop &, const eckit::Configuration * const *);
  void ufo_radiosonde_delete_f90(F90hop &);
  void ufo_radiosonde_t_eqv_f90(const F90hop &, const F90goms &, const F90odb &, const F90ovec &, const F90obias &);
  void ufo_radiosonde_settraj_f90(const F90hop &, const F90goms &, const F90odb &);
  void ufo_radiosonde_t_eqv_tl_f90(const F90hop &, const F90goms &, const F90odb &, const F90ovec &);
  void ufo_radiosonde_t_eqv_ad_f90(const F90hop &, const F90goms &, const F90odb &, const F90ovec &);

// ----------------------------------------------------------------------------
//  Radiance observations
// ----------------------------------------------------------------------------
  void ufo_radiance_setup_f90(F90hop &, const eckit::Configuration * const *);
  void ufo_radiance_delete_f90(F90hop &);
  void ufo_radiance_eqv_f90(const F90hop &, const F90goms &, const F90odb &, const F90ovec &, const F90obias &);
  void ufo_radiance_settraj_f90(const F90hop &, const F90goms &);
  void ufo_radiance_eqv_tl_f90(const F90hop &, const F90goms &, const F90odb &, const F90ovec &);
  void ufo_radiance_eqv_ad_f90(const F90hop &, const F90goms &, const F90odb &, const F90ovec &);

// ----------------------------------------------------------------------------
//  Ice concentration observations
// ----------------------------------------------------------------------------
  void ufo_seaicefrac_setup_f90(F90hop &, const eckit::Configuration * const *);
  void ufo_seaicefrac_delete_f90(F90hop &);
  void ufo_seaicefrac_eqv_f90(const F90hop &, const F90goms &, const F90odb &, const F90ovec &, const F90obias &);
  void ufo_seaicefrac_settraj_f90(const F90hop &, const F90goms &);
  void ufo_seaicefrac_eqv_tl_f90(const F90hop &, const F90goms &, const F90ovec &);
  void ufo_seaicefrac_eqv_ad_f90(const F90hop &, const F90goms &, const F90ovec &);

//
```