GSI Polymorphic Guess Interpolator
Major Use-Cases as Schematic Code

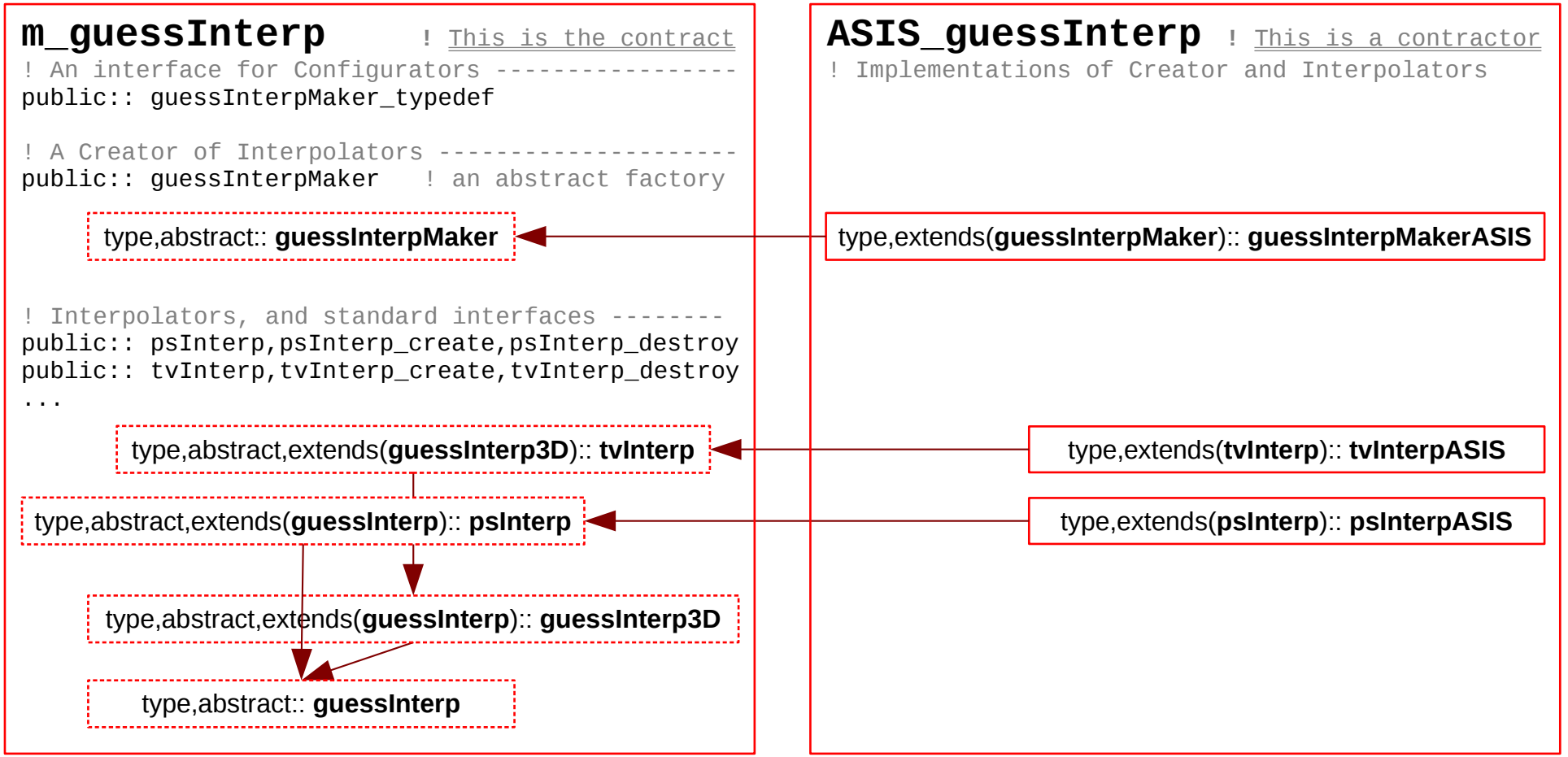
```
! UC-1: Interpolations with distributed observations !

time_forward_loop: do ... ! time_sections of background_states
  ...
 call OBS_setuprhsall(obs_dstr){
  intent(in):: obs_dstr(:) ! (ndat)
   ...
  obs_input_source_loop: do is=1,size(obs_dstr)
    ...
   call X_setup(data=obs_dstr(is)%data(:,:), ...){
    intent(in):: data(:,:) ! (nreal,ndata)
          !==== Step 1: declarations ====
    class(psInterp),pointer:: iges_ps    ! this is 2-d
    class(tvInterp),pointer:: iges_tv    ! this is 3-d
          !==== Step 2: constructions ====
    call psInterp_create(iges_ps)
    call tvInterp_create(iges_tv)
     ...
    obs_data_loop: do i=1,size(data,2)
          !==== Step 3: time-dimension verifications ====
     time=data(itime,i)
     in_curbin = iges_ps%dtime_check(time) .and. &
                 iges_tv%dtime_check(time)
     if(.not. in_curbin) cycle obs_data_loop
          !==== Step 4: interpolations ====
     elat=data(ilat,i)
     elon=data(ilon,i)
     call iges_ps%ginterp(psges,elat,elon,time)
     plev=data(ilev,i)
     call iges_tv%ginterp(tvges,elat,elon,plev,time)
      ...
    enddo obs_data_loop
          !==== Step 5: destructions ====
    call psInterp_destroy(iges_ps)
    call tvInterp_destroy(iges_tv)
   } ! end call X_setup()
  enddo obs_input_source_loop
   ...
 } ! end call OBS_setuprhsall()
 ...
enddo time_forward_loop
```

```
! UC-2: Inquiring PE destinations w.r.t. the guess !

call OBS_PEinquire(obs_orig,iPE_dest){
 intent(in ):: obs_orig(:) ! (ndat)
 intent(out):: iPE_dest(:) ! (ndat)
  ...
 obs_input_source_loop: do is=1,size(obs_orig)
  nobs=size(obs_orig(is)%data,2)
  allocate (iPE_dest(is)%iPEs(nobs))  ! for example ...
   ...
  call X_PEinquire(data=obs_orig(is)%data, &
                   iPEs=iPE_dest(is)%iPEs, ...){
   intent(in ):: data(:,:) ! (nreal,ndata)
   intent(out):: iPEs(  :) ! (     ,ndata)
         !==== Step 1: declarations ====
   class(psInterp),pointer:: iges_ps    ! this is 2-d
   class(tvInterp),pointer:: iges_tv    ! this is 3-d
         !==== Step 2: constructions ====
   call psInterp_create(iges_ps,inquiriesOnly=.true.)
   call tvInterp_create(iges_tv,inquiriesOnly=.true.)

   obs_data_loop: do i=1,size(data,2)
         !==== Step 3: PE-inqueries ====
    elat=data(ilat,i)
    elon=data(ilon,i)
    call iges_ps%inquire(elat,elon,iPE=iPE_ps)
    call iges_tv%inquire(elat,elon,iPE=iPE_tv)
    ASSERT(iPE_ps==iPE_tv)
    iPEs(i)=iPE_ps
   enddo obs_data_loop

         !==== Step 4: destructions ====
   call psInterp_destroy(iges_ps)
   call tvInterp_destroy(iges_tv)

  } ! end call X_PEinquire()
   ...
 enddo obs_input_source_loop
 ...
} ! end call OBS_PEinquire()
```

GSI Polymorphic Guess Interpolator
Requirements and Design Considerations

- Respect current GSI use-cases
  - There are other use-cases than the two given earlier.
  - Interfaces are built around physical quantities (latitude, longitude, ps, tv, wind, etc.),
    - Made no assumption to implementation details (grid definitions, parallel partitions, choices of model variables), in the generic specific guess-grid;
    - Some minimum requirements are given below;
    - Implementation details are all left to specific implementations, as extensions.
  - Can be expanded to support background ensemble use-cases.
  - Continue to support time-forwarding background processing.
- Assumptions to a guess-state component,
  - It is horizontally distributed. (Convert a spectral grid to physical space at initialization?)
  - Horizontal, vertical, and temporal dimensions are separable (lat-lon do not have to be).
  - Support PE inquiries for a given lat-lon pair of targeted variable types
    - They are potential interpolation requests;
    - Observations will be distributed according to the results of these PE inquiries.
  - ***Support properly localized interpolation requests***
    - Look out for locations in between grid partitions
  - Support of multiple grid/partitions is up to implementations
    - e.g. differences between cube-guess, met-guess, chem-guess,.
    - certain consistency in co-location may apply.

GSI Polymorphic Guess Interpolator
Type Extension Hierarchy of **`m_guessInterp`**



GSI Polymorphic Guess Interpolator
An Implementation Road Map

1. Implement module ***`m_guessInterp`***, which does not do anything real, but a self syntactic verification.
   - Some naming issues are still in consideration.
2. Add a module ***`ASIS_guessInterp`***, a place for details of ***`m_guessinterp`***, by incrementally moving in GSI specific interpolation implementations, away from `setup()`, with necessary improvements.
3. After all related code movements are done, GSI is ready to implement a different ***`guessInterp`*** extension, with other required implementations.
   - `OBS_alltoallv(obs_orig,iPE_dest,obs_dstr,iPE_orig)`