XIAS

XIOS Interface for Arpege-climat and Surfex

S.Sénési – August 2016

Document **draft,** version 0.4

Warning : The kind of readers who think that they can use a complex system without spending some time studying it can try to read only the text in red.

Table des matières

1. Scope, audience and structure of this document

This document reports on the interfacing of Arpege-Climat and Surfex to XIOS ; XIOS provides an alternate way to output model diagnostics, with some differences with respect to legacy output schemes for these models, which were considered useful for some applications.

The document is intended for three categories of readers :

- 'plain' model users, whose objective is simply to run with XIOS, and to configure the output diagnostics for their model runs, in order to save resources w.r.t. a brute force copy of pre-existing output settings; they can also discover here to what extent they can post-process model outputs directly during the simulation, without having to modify model code
- model developpers, who must take care of XIOS output scheme when introducing new diagnostics in one of the models; this is quite simple
- the happy taxpayers who will take care of all that (provided it proves to be useful) in the long run (as e.g. when phasing Arpege-Climat with a new cycle of Arpege/IFS), and who should understand the details of the interfacing

This document **does not** yet fully cover the following questions :

- how do I run CNRM-CM using XIOS, and getting exactly the same output as without XIOS ?
- how do I run Surfex using XIOS, and getting exactly the same output as without XIOS ?

This due to the fact that XIOS offers much more flexibility that legacy outptut systems, and so, there is a need for further discussions with various stakeholders beforehand, based on the material of this version of the document.

After a common introduction, there is a section for each reader category, in the same order as above. We try to explain enough basics about XIOS while avoiding to overlap with the role of its own documentation.

2. Rationale, and XIOS basics

Release V8.1 of Surfex and version 6.3 of Arpege-Climat (which includes a beta version of that Surfex release) have been interfaced to XIOS, the Xml configurable Input/Output Server, at its release r909. XIOS [1] (see https://forge.ipsl.jussieu.fr/ioserver/wiki and preferably the tutorial there) was developped by IPSL/LSCE in order to provide both high performance output for massively parallel simulations, an easy configuration of model outputs and of some inline post-processing. The three latter features are instrumental in the context of CMIP exercises

The performance is achieved by releasing the client models (Arpege and Surfex) of the input/ouput operations : MPI processes (called 'XIOS servers') are dedicated to the reading/writing, and their number can be adjusted to the flow rate of I/O. Server-less operation is also possible.

The client models do deliver the fields to be (possibly) written to XIOS using a simple interface, actually through an XIOS-provided function call with two arguments : a string which is for XIOS an identifier for the field, and a multi-dimension array representing the part of the physical field for a geophysical variable corresponding to the part of the domain which is processed by the current model's MPI process. From that stage, XIOS work is shared between the process running the client model (for all tasks, such as time averaqes, which do not imply gathering fields for the whole domain), and other tasks which do so, or which imply an actual I/O operation ; these latter tasks are hanlded by XIOS servers. So, delivering a field to Xios does not imply each time calling an MPI routine or an I/O routine.

XIOS actually manages the delivered fields based on a configuration file, in XML syntax, which allows for a structured description of axes, domains, grids, fields, files and filters (see below), and to finely control and describe the content of output files. Some post-processing can also be described that way. An xml file can reference another xml file, which allows for structuring the configuration files.

There is also an API (i.e. an Application Program Interface - said otherwise : a library) which allows to configure XIOS on the same topics directly from models code. The interfacing of Arpege and Surfex makes heavy use of the API for releasing the user with part of the burden of lenghty configuration files. This is especially true for Surfex (see section 'user guide')

In XIOS terminology, the configuration elements are as follows :
- 'contexts', wich are namespaces; Arpege and Surfex share a single namespace; in a coupled model such as CNRM-CM6, Nemo and Gelato do share another namespace
- '**identifiers**', which are chosen by model developers or by you in xml files; **they must never collide** within a 'context' (i.e. be re-used for identifying two distinct elements)
- '**fields**' are either directly delivered by the client models or are the result of applying 'filters' (see below) to other 'fields'. They are identified by a string called a '**field_ref**'. All fields must be declared (through config file or through the API).
- 'filters' allow to post-process 'fields'; an important category of filters for climate are time filters, which allow to average, and to compute maximum/minimum of fields across time; 'filters' also include artihmetic operations and spatial interpolations
- '**files**' are description of the desired output files : each 'file' has a single '**output freq(uency)**' and includes a number of 'fields';
- NetCDF file attributes and variable attributes are set using XIOS 'variables' which are respectively attached to 'files' and to 'fields' in a 'file'
- 'axes' and 'domains' describe the geometry of the simulation; a 'grid' is the combination of one (horizontal) 'domain' and 0 or more (vertical or not) 'axis'

Limitations :
- With version r909, XIOS managed formats are limited to NetCDF (either NetCDF4 or NetCDF classic). Grib format may be devleopped in some later XIOS version.
- For using more than one server, Xios must be linked to a parallel version of the netCDF library.
- Because of limitation of the PNetDCF library, it is not possible to write NetCDF4 format compressed files in parallel (one single server must be used).
- When processing very small grids with Surfex, XIOS proves to be slow w.r.t. the legacy NetCDF scheme

3. User guide

3.1. For both models

a) Supported model grids

XIOS can be activated for all geometries currently handled by both models (except for the MUSC-model geometry). Tests included the reduced gaussian grid, an Aladin case, and the Surfex 'latlon' regular grid. For Aladin, the output is defined on the C+I zone

b) Installing XIOS

You usually do not have to install XIOS yourself : Surfex distribution package embarks and compiles Xios for you ; Arpege is usually provided as a binary linked to all necessary libraries ; the XIOS server binary is colocated with these versions of XIOS libraries. If you need to install it, see the installation directives

c) XIOS inputs

XIOS needs a root configuration file : this file is in XML syntax, must be named iodef.xml in the run directory, and can reference other XML files using the 'src=' construct.
Herafter shows a simple, working, example for a Surfex Offline run using Oasis and not using any Xios server ; the definition of Surfex fields and how to output them is here deferred to a companion configuration file ./surfex.xml .

```
<?xml version="1.0"?>
<simulation>
  <context id="surfex" src="./surfex.xml" />
  <context id="xios">
     <variable_definition>
        <variable_group id="coupling">
           <variable id="using_server" type="bool">false</variable>
           <variable id="using_oasis" type="bool">true</variable>
           <variable id="oasis_codes_id" type="string">surfex</variable>
        </variable_group>
     </variable_definition>
  </context>
</simulation>
```

Samples of xml files for surfex and arpege show in the respective relevant sections below

When setting using_server to 'true', you must launch one or more MPI task running the xios server , such as in

```
      mpirun  : -np 10 ./xios_server : -np 150 ./arpege
```

There is no need to quote the number of servers anywhere else, nor to ensure consitecny between this server or other parameters (except sometime for performance, see last pages of the tutorial)

d) Outputs

Syntax for describing required outputs

Output description goes in the 'file definition' section of the xml file**;** a sample shows in § 'Surfex case' below ; see the XIOS tutorial for details beyond the ones described there and below.

Output frequency

For each output file defined in XIOS config file, a single time period applies for all temporal operations applied to the delivered fields before writing (see examples below). This is set using the 'output_freq' file attribute, which you may change at will. Hence, there is no tight link between the frequency of fields delivery to XIOS and the time resolution of the variables in diagnostic files, (except that it would not make sense to have a freq_output shorter than the diagnostic delivery period). XIOS syntax for setting time periods include the following, self-explaining patterns : 1"y", "12mo", "1d', "24h", "3600s", "1.5h" , "1h30m", "1m", "60s" ; note : "2ts" means : two model timesteps.

Instant versus averaged field, and sampling

Each field is declared (by the model code or by an xml file), using XIOS attribute freq_op, as being either an instant field or an averaged field (the full list of options actually is : instant, average, minimum, maximum, once and cumulate) . By default, when outputing an instant variable in a daily file by a run starting at 0h, you will get one value at 0h each day, the first one being valid for the end of the first day of simulation ; when requesting an averaged field in a daily file, you will get for the first field the average over a sample beginning with the value after first model time step and ending with the value at 0h next day ;
Whatever the freq_op option, the sampling period is governed by the rules described below for each model

You can also define more complex conditions, such as outputing once every month a daily averaged temperature : applying a filter can be done with a given period (the filetr has a freq_op attributes), which defines the actual sample period of its ouput.
See example in § ...
See also the tutorial

Time dimensions (and mix of)

The NetCDF time variable it is fully managed and described by XIOS in a CF-convention compatible way in the output files, including time bounds for averaged variables ;

Time dimensions mix : an output file may mix time-averaged field and instant fields; in that case, the NetCDF variables for such fields will all have the same time dimension named 'time_counter', but will also have a 'coordinates' attribute quoting either 'time_instant' or 'time_centerd'; both time variables, with dimension 'time counter' will be present in the NetCDF file. This is a bit more tricky than what the usual post-processing tools handle, but is the only CF compatible way to deal with such a case. See an example as appendix xxx. CDO 1.7 cannot deal correctly with this situation, so it could be handy to separate such variables in different files.

Saving storage disk space

A number of XIOS xml elements have a toggle attribute named 'enable' : for a field in a file, for a file as a whole, and also for a field in a field_group, a file_group or in field_definitions. This allows various levels of action for actually choosing whether to output a field :

- a disabled file will not be created whatever the enabling of its fields of field groups
- the enabled attribute value for a field in a file_definition has higher priority than in the field_definition it refers to, or than the group it is included in ('facing the child, the parent gives up')
- grouping can be nested to an arbitrary number of levels

Because XIOS allows to very easily define multiple output files with varied frequencies and varied diagnostic fields lists (see further below), you can save storage resources provided you sit for a while and think seriously at what your actual needs are (which variables, at which frequency)
Example ....

e) Performance

...

3.2. Surfex case (either Offline or embedded in Arpege)

a) Activating XIOS output scheme

This is done using value 'XIOS' for namelist variable CTIMESERIES_FILETYPE (either in NAM_IO_OFFLINE or NAM_IO_SURF_ARO). You then need to provide an XIOS configuration file named iodef.xml (see above and also § Eclis). The CSELECT and LSELECT parameters are then neglected.

b) Surfex output files definjtion example

The following configuration file content illustrates how to reproduce usual Surfex output files for a typical run with a few selected ISBA daily diagnostics, when no use is made of Xios time-averaging capacity. We assume that such a file is sourced in an iodef.xml such as the one above. Here :

- the sampling period is set to 24h (48 time steps for this run) ;
- two files are defined, with a daily frequency, which names and content mimics usual Surfex output files ;
- the desired diagnostics in each file are listed ;
- a « field group » is used in order to set the type of time operation applied to each member of the group ; here we set it to 'instant' , on purpose
-

```
 <context id="surfex">

 <variable_definition>
   <!-- set largest possible sampling frequency which allows to match 'XTSTEP_OUTPUT=86400.' when time step is 1/2h -->
   <variable id="timesteps_between_samples" type="int">48</variable>
 </variable_definition>

 <file_definition >
   <file_group output_freq="24h" >
     <file id="ISBA_DIAG_CUMUL.OUT" >
              <field_group id="cumulated_fields" operation="instant"  >
                <field field_ref="EVAPC_ISBA" />
              </field_group>
     </file>

     <file id="ISBA_DIAGNOSTICS.OUT" >
              <field_group id="snapshot_fields" operation="instant"  >
                <field field_ref="WGI_ISBA" />
                <field field_ref="TS_ISBA" />
              </field_group>
     </file>
   </file_group>
 </file_definition>
</context>
```

c) No need to declare usual diagnostic fields

All Surfex usual diagnostics that your run will produce are automatically known to XIOS based on a declaration done through the API. This apply to those diagnostics written by write_surf().

Diagnostics are then automatically declared to Xios, using their Surfex name as an identifier and also as the NetCDF variable name, and with NetCDF attributes long name and units taken from the Fortran code (specifically from the HCOMMENT variable passed to write_surf, but not yet from the string in init_outfn). They are also automatically enabled, and their sampling period is set as explained below in § time sampling. Their missing value is set to Surfex one. The kind of Xios time operation is set to 'instant', which matches the habits for Surfex. You may override all these attributes by setting them explicilty in the xml configuration file, such as with

```
<field_definition>
   <field id="qsurf" name="surface_air_humidity "long_name="Surface Air humidity"  unit=kg/kg"
                              operation="average" freq_op="30m" enable="false" default_value="-999.""/>
</field_definition>
```

If you use an identifier in XIOS configuration files which is not the name of a declared diagnostic (e.g. not written by a write_surf call, nor explicitly declared in xml), XIOS will raise an error if it is requested in a file (or as input to a 'filter')

In section 'Model developer Guide', we address the case of 'non-usual' diagnostics, i.e. diagnostics which are not handled by a call to write_surf in a write_diag_xx module.

Please note that the PROVAR_TO_DIAG namelist toggle is not active (to be checked ... xxx)

d) Time sampling of Surfex fields by XIOS

You can flexibly choose which diagnostic is averaged over which period before being written ; this can also be a time maximum or time minimum operation. We describe here which is the time period for building the sample which undergoes such a time operation.

Diagnostic fields are delivered to XIOS by routine write_diag_surf, which means that they are delivered with a period frequency which is equal to, or a mutliple of, XTSTEP_OUTPUT for the offline case, or NFRSFXHIS for Surfex embedded in Arpege.

The Arpege-embedded case presently needs that write_diag_surf is called at every timestep (NFRSFXHIS=1)

For the Offline case, you have to choose between two options : either let XIOS compute time averaged fluxes, or compute it as a post-processing based on Surfex-time-cumulated fluxes (as usual when not using XIOS). For the first case, you have to decide for a small value of XTSTEP_OUTPUT in order to get a sampling frequency consistent with the desired accuracy for the geophysical variable showing the quickest changes. You can set it as small as the Surfex timestep.

The other XIOS variable which drives time sampling is named « timesteps_between_samples », such as in :

```
<context id="surfex">
 <variable_definition>
   <variable id="timesteps_between_samples" type="int">1</variable>
 </variable_definition>
....
```

and acts in addition to the mechanism described above, for all variables except those which have an explicit « freq_op ». It's default value is 1. Hence, for an Offline example, if time_step = 1h, XTSTEP_OUTPUT=3h, and timesteps_between_samples=2 ; then if variable TS_ISBA has no freq_op set , it will be sampled every 6h (the smallest multiple of 2*1h and 3h) ; and if variable RUNOFF_ISBA has a freq_op set to 1h , it will be sampled every 3h

e) Getting a comprehensive output

You can get an output of all Surfex diagnotics at any frequency by enabling a special file named 'surfex_cselect' inserting an xml sequence such as shown below in the file_definition section

```
  <file id="surfex_cselect" name="surfex_all_fields" output_freq="1ts" enabled=".TRUE." >  </file>
```

f) Dimensions and dimension names

NetCDF dimension names for the horizontal - case of 1D and curvilinear grids

XIOS logic for managing the masks associated with the four tiles (Land, Sea, Lakes, Town) is such that the name of the dimension variable(s) for diagnostics vary from on tile to another; this should however be harmless for well-coded NetCDF processing.

NetCDF additional dimensions - axes other than horizontal

A Surfex namelist toggle, LALLOW_ADD_DIM has been introduced in namelists NAM_IO_SURF_ARO and NAM_IO_OFFLINE; if it set to true, a number of diagnostic fields which have an additional dimension (w.r.t. the horizontal dimension(s)) will be written as multi-dimensional NetCDF variables (rather than be disguised with mutliple variable names, such as e.g. when using a suffix bearing the patch number). Said otherwise, you will get e.g. TG(lat, lon, layer) rather than TG_ISBA_01(lat,lon), TG_ISBA_02(lat,lon) ...TG_ISBA_28(lat,lon).
This scheme is however limited to one additional dimension, and to these diagnostics for which the write_diag_xx routines have been modified accordingly for delivering the multi-dimensional field (see section 'Model developer guide') ; this actually includes today only variables SWD, SWU, TG, WG and WGI, for which these respective dimensions are pre-defined : swband, ground_layer, ground_ice_layer.
Further, when the additionnal dimension is the patch number, the loop with suffix is enforced anyway.

Xios interface can presently manage the case of two additional dimensions (in addition to horizontal ones), but only when the second one is patch number ; in that case, it will loop on patch number writing e.g. snow_content_01(lat, lon, snow_layer) (where 1 is the patch number)

g) Sub-grids and re-gridding fields

XIOS allows to output fields both on a sub-grid and after some regridding.
xxx

3.3. Arpege/Aladin case

a) How to activate XIOS

Outputing with XIOS from Arpege relies on selecting the XIOS output scheme in Surfex (see above). This can happen independantly of FullPOS settings, and hence in addition to FullPOS output, so, take care of disabling FullPOS (by setting NFRPOS to 0, or to AUTO is using Eclis) except on purpose. (Note that setting LFPOS to false does not work with Arpege-Climat). You must presently set NFSRSFXHIS=1 for a correct behaviour of XIAS (this is done by Eclis). You cannot have Arpege use XIOS while Surfex uses another diagnostics output format

b) Arpege diagnostics which are known to XIOS

A number of Arpege fields are declared and sent automatically from Arpege code (which does not mean that they are automatically written in output file). The list of the corresponding identifiers depends on the Arpege code you use (an initial list shows as appendix xxx), and you cannot quote in the xml file an identifier which is not declared by that Arpege code (XIOS will complain and stop the run). If you struggle to find the actual list, you may grep the arp/ directory modules for 'call mse_xios_send' and 'call sfx_xios_declare_field'.
These diagnostics are declared to XIOS with an 'operation' attribute, and a set of other attributes, which have been chosen by the model developper, as e.g. 'operation=average' for fluxes and 'operation=instant' for pressure at sea level. You may override it using xml declarations (see example in Surfex § above)

c) Time sampling

Diagnostic fields are delivered to XIOS (mainly by routine aplpar) based on the value of variable timesteps_between_samples (described above for Surfex) and of their own freq_op value (if set) ; small values allow to ensure a high frequency time sampling for time-averaged variables, and also allows to access high frequency (1 per timestep) for instant variables, for debugging purposes (which can be much useful when combined with the capacity to define output fields restricted to a sub-grid, see xxx).

For a number of diagnostics (e.g. surface level pressure), for which there is an additional computation cost and/or obviously no need for a high frequency, a freq_op value is set in the code (as e.g. 6h for sea level proessure, or the value of radiation scheme call timestep for radiative fluxes), which can be overriden by setting a value in the xml file for this field

d) Example of a XIOS configuration file for Arpege

3.4. Troubleshooting

XIOS provide quite systematically a detailed error message, both in model run stderr and, if activated, in dedicated files, one per client model process (named xios_client_xx.err) and one per server (named xios_server_xx.err). Using Eclis, a few of these files are copied upon crash in the relance directory

A typical error message has much debug information, but the most useful part for XIOS users always show as exemplified in bold face below

```
[26] > Error [CObjectFactory::GetObject(const StdString & id)] : In file '/home/gmgec/mrgu/senesi/SAVE/XIOS/XIOS-902-CM6bO2/src/object_factory_impl.hpp', line 78 -> [ id = zqsat, U = field ] object was not found.
[26]
[26] application called MPI_Abort(MPI_COMM_WORLD, -1) - process 26
```

From the author's experience the most common errors are :

- XIOS is very sensitive to the syntax of xml files. It is recommended to use an editor which highlights xml syntax, and to modify files with care.
- XIOS may hang the model run very early, without any warning, in a number of cases of wrong syntax (e.g. : a duplicated closing bracket « /> »)
- XIOS may hang if a field has been declared with a wrong domain (e.g. a too large one), either due to the xml declaration or to a wrong call in the model code.
- XIOS reports errors in files named like xios_server_00.err or xios_client_02.err ; the error messages are usually quite explicit and clear . On some occasions, the error is rather reported in the stderr of the model run.
- ...

3.5. Use with ECLIS

ECLIS, the CNRM-CM run environment, is described at http://www.cnrm-game-meteo.fr/cm/spip.php?article1. ECLIS version 6.9 or above is useful to smoothly run Arpege and or Surfex with XIOS, and is needed to run CNRM-CM with XIOS in Apege, through the following new parameters :

- IOSOUTPATT as a pattern for listing XIOS output files to be processed by :
- IOSPOST for telling which script will process XIOs outputs ; such a script is run in step3 of the simulation and is supposed to :
  - possibly process all files in the current directory, and
  - to echo on its stdout the list of files that should be archives (and only that !)
- OTHER_FILES for tellin which secondary xml files are to be copied in run directory

It also includes additional features for :

- identifying if Surfex activates XIOS (from its namelist ; this works correctly only if CTIMESERIES_FILETYPE has the same value in NAM_IO_OFFLINE and NAM_IO_SURF_ARO)
- modifying N1POS=AUTO in Arpege NAMCT1 namelist (depending on the above)
- modifying using_server = AUTO in XIOS namelist (depending on the chosen number of XIOS processes)
- modifying using_coupler= AUTO in XIOS namelist (depending on quoting a coupler namelist)

...

3.6. Post-processing with XIOS operations and filters

XIOS allows to perform time statistics (i.e. average,min, max), fields arithmetics, data selection (e.g. zoom), and interpolation (either horizontal or vertical). This is further explaned in XIOS documentation (see the tutorial) and a few examples show in annex xxx

4. Model developper guide

4.1. Surfex :

There are two ways for adding a diagnostic for output by XIOS : the Surfex legacy way, i.e. adding it in a write_diag_xxx routine, or the XIOS quick way. Only the former ensures compatibility with all Surfex output formats. Hence, the 'quick way' should be used only for development or personal purpose, as it may may be useful for prototyping new diagnostics.

There is no special care when adding a diagnostic field the legacy way, except :

- to ensure that the HREC field name do not collide with the Xios name of another field, neither in Surfex code, nor in Arpege code, nor in 'standard' xml files
- to ensure that the field description beared by the 5th argument to write_surf (usually in variable YCOMMENT) is well-formed, with the field units between parenthesis at the end
- to take care of any new non-horizontal dimension (see further below)

The quick way for adding a diagnostic, or more genrally for outputting any array, is to call XIOS_SEND_BLOCK from any routine in Surfex physics such as this example in routine coupling_seafluxn :

```
CALL SFX_XIOS_SEND_BLOCK("zqsat",ZQSAT)
```

In addition, if you want to output it, beyond referencing it in a fiee_definition, you **must** declare this diagnostic to XIOS either by :

- inserting a relevant line in the field_definition section of the xml configuration file, which describe the tile/domain and if applicable the vertical dimension, as e.g :

```
      <field id="zqsat"  name="zqsat_from_seaflux" domain_ref= »SEA »/>
```

- or in the code by a call to sfx_xios_declare_field , e.g. at the end of sfx_xios_setup_ol and sfx_xios_setup_aro, such as :

```
    CALL SFX_XIOS_DECLARE_FIELD('zqsat',HDOMAIN='SEA',HCOMMENT='Q_at_saturation (1)' , KFREQOP=180)
```

  argument KFREQOP is optionnal and set the XIOS sampling period, if none si declared for this field in the xml files

The tile/domain names to use are : FULL, SEA, NATURE, MASK TOWN. Default is FULL

The reason for this additional constraint with Surfex is explained in section 'Technical Guide'

a) Handling non-horizontal dimensions :

One may wish to allow the Surfex user to take advantage of NetCDF capacity to handle multi-dimension arrays when outputing an existing or a new diagnostic which has an additional dimension (e.g. a vertical axis, like for ground layers or seaice categories) ; in that case one should :

- write or modify the sequence of code in routine write_diag_xxx in a way similar to the boldface part in :

```
IF (LALLOW_ADD_DIM)  THEN
  YRECFM='TG_ISBA'
  YCOMMENT='Soil temperature (K)'
  CALL WRITE_SURF(DGU, U, &
      HPROGRAM,YRECFM,ZTG,IRESP,HCOMMENT=YCOMMENT,
              HNAM_DIM2='tg_layer')
ELSE
  DO JLAYER=1,IWORK
    WRITE(YLVL,'(I4)') JLAYER
    YRECFM='TG'//ADJUSTL(YLVL(:LEN_TRIM(YLVL)))
    YRECFM=YRECFM(:LEN_TRIM(YRECFM))//'_ISBA'
    YCOMMENT='X_Y_'//YRECFM//' (K)'
    CALL WRITE_SURF(DGU, U, &
       HPROGRAM,YRECFM,ZTG(:,JLAYER),IRESP,HCOMMENT=YCOMMENT)
END DO
ENDIF
```

- optionally, declare the dimension name (which is passed as HNAM_DIM2 arg to WRITE_SURF, here above : 'tg_layer') ; this can be done by calling routine set_axis in sfx_xios_setup_ol and sfx_xios_setup_aro ; at that stage, you can also declare the axis as a coordinate (by providing an array of values such as ground layer depth in meters), if it makes sense ;
- at run stage, the user must put LALLOW_ADD_DIM in the relevant namelist (NAM_IO_SURF_ARO or NAM_IO_OFFLINE)

You should reuse an axis name only on fields which share the same size for the corresponding dimension. A number of additional dimension names have been initially introduced (see the User guide section above). If you do not provide a value for HNAM_DIM2, Xios will generate a dynamical dimension name ; you may (but do not need to) declare new dimension names and the corresponding dimension size by calling routine set_axis in sfx_xios_setup. At that stage, you can also provide the coordinate values for this axis (e.g. the central wavelengths of short wave bands), however no provision was yet made to also provide coordinate bounds. If the coordinates are not the same for the Offline and the Arpege case, you should rather put distinct calls both in sfx_xios_setup_aro and sfx_xios_setup_ol.
You should modify modd_xios to define there the variables holding new dimension names. AT a later stage, if the other Surfex output schemes can make use of dimension names, this section of modd_xios could be moved elsewhere.

As an alternate way, you can declare a new dimension as an axis in Xios configuration files. See XIOS documentation

4.2. Arpege

a) Simple case

Adding a diagnostic usually requires only to deliver it from Arpege code to XIOS using function mse_xios_send such as at the bottom of aplpar :

```
CALL MSE_XIOS_SEND('zg'  ,PFIELD2=PAPHIF(KIDIA:KFDIA,:))
```

You may use either this simple form, which has only two arguments (field_id as first arg, and field array as arg PFIELD or PFIELD2) or take this opportunity to feed XIOS with more meta-data for the field, namely a description and units, assembled in the CDCOMMENT argument of the call as 'description (units)' such as in :

```
CALL MSE_XIOS_SEND('ps'  ,PAPRS(KIDIA:KFDIA,KLEV),CDCOMMENT='surface_air_pressure (Pa)')
```

However, in this latter case, the call must be actually issued also during at step 0 whatever the run configuration. Otherwise, the field will not be correctly decalred to XIOS (see next §)

It is recommended to use as :
- field identifier : the CMIP6-standardized variable name, as found by using query facility at http://clipc-services.ceda.ac.uk/dreq/mipVars.html
- description : the long name of the variable in the CF convention, if it exists :
  - follow the link on the variable name if one is found by the search above, and look at bullet 1.8 CF Standard Names
  - or refer to http://cfconventions.org/Data/cf-standard-names/31/build/cf-standard-name-table.html)

If applicable, you can also prescribe a default value for the sampling period of the field by XIOS using argument KFREQOP expressed as a number of minutes, such as in :

```
  CALL MSE_XIOS_SEND('zg',PFIELD2=..., CDCOMMENT='geopot... (J kg-1)' , KFREQOP=180)
```

All this apply to fields which first dimension is the index in the Arpege 1D grid (so physically 2d fields ; use argument keyword PFIELD1 or put array as second argument), and possibly with 1 additional dimension (use argument keyword PFIELD2)

Fields showing other grids are not yet managed.

The call can be issued from any routine in the time loop, either from routine where the space domain is splitted in NPROMA blocks or where they are gathered (this last case not fully tested yet)

For field with additional dimensions, please see Surfex sub-section one or two pages above.

b) Tricky case

A restriction occurs in some cases : if the call to MSE_XIOS_SEND is not systematically executed during step 0 (as e.g. in actqsat, or depending on the test mse_xios_field_is_active- which is false at step 0), then the arguments related to declarations, i.e. CDCOMMENT and KFREQOP cannot be forwarded to XIOS during its init phase, and are neglected. In order to bypass this limitation, you must a call to SFX_XIOS_DECLARE in routine MSE_XIOS_DECLARE such as :

```
CALL SFX_XIOS_DECLARE_FIELD('psl',HCOMMENT='air_pressure_at_sea_level (Pa)', KFREQOP=360)
```

c) Saving on diagnostic computing time

On some occasions, computing diagnostics can be expansive in CPU. In order to trigger such computation only for those timesteps actually requested by the runtime xml configuration files, on may use function mse_xios_field_is_active, to query the interface for the need actual need for the field at this timtestep. The only argument is the Xios field name. This function first queries Xios to know if the field is configured to be output in some files, and next analyzes if the current timetsep is consistant with period requested for this field (i.e. argument KFREQOP of the call to mse_xios_send_field or to sfx_xios_declare_field, or attribute FREQOP in the xml file) Example, from aplpar :

```
IF (MSE_XIOS_FIELD_IS_ACTIVE('clwc')) THEN
  ZTMP=0.
  DO ILEV=1,KLEV
    ZTMP(KIDIA:KFDIA) = ZTMP(KIDIA:KFDIA) + PQLI(KIDIA:KFDIA,ILEV)*PDELP(KIDIA:KFDIA,ILEV)/RG
  ENDDO
  CALL MSE_XIOS_SEND('clwc',ZTMP(KIDIA:KFDIA))
ENDIF
```

5. Technical guide

5.1. Interface design

Some reasons behind the interface design :
- because Surfex in Offline mode must be autonomous in using XIOS, it has been interfaced first
- when Surfex is run in Arpege/Aladin, Surfex interface to XIOS must cope with all geometries for these models; hence it must be able to inherit grid description from Arpege; the Surfex interface must also in that case cope with the NPROMA vector-slicing scheme of Arpege, which is not handled natively by Xios (which assumes that fields for the whole of the MPI task are delivered).
- given that Arpege is no more used in climate runs without Surfex, and given the capabilites of XIOS/Surfex interface, there was no use to develop an autonomous XIOS/Arpege interfacing;

Hence, Arpege uses :
- all XIOS setups done by Surfex (including the definition for XIOS domain corresponding to the whole grid -'FULL', and including the calendar and timestep setups);
- routine sfx_xios_declare_fields for declaring fields
- routine sfx_xios_send_block for delivering fields
- when needed, routine set_axis for declaring a new axis to XIOS

Scanning Arpege geometries for declaring it to XIOS has a lot in common with wat is needed for declaring it to Oasis; this common part stand in routine aroini_surf. The description of Surfex masks for the various tiles is in mse/internals/sfx_xios_setup_aro (or in surfex/OFFLIN/sfx_xios_setup_ol for the Offline case). The call tree of XIOS related routines in Arpege and Offline SUrfex shows as appendix xxx, with a number of comments

In the Surfex legacy NetCDF output option, there is an adhoc scheme for initializing NetCDF attributes for diagnostic fields (i.e. variable name, long name, and dimension names) : because there is no general table of such attributes built at model init stage, and because this information is just hard-coded and scattered across Surfex code in physical scheme-specific diagnostic writing routines, this legacy scheme uses the diagnostic write routine as a dummy write, at the time of the first actual write, for capturing the NetCDF attributes. The same scheme has been applied in Offline and Online interface to XIOS. **This implies that the termination of XIOS declaration (xios_close_context_definition) has, for now, to be delayed until the stage of writing the first diagnostics set (call to aro_surfdiag), which happens only after the end of the first model time step 0. This has complicated a lot the logics for the sequence of calls to XIOS functions, and even the code of some new functions. This could be re-engineered if the principle was set that all XIOS field_definitions should lay out of the code, in xml files. However, this would miss an opprotunity : that the declaration of field characteristics (name, units, operatipn) be written close to the code which computes the field, and hence, be consistent with the code.**

5.2. Code changes

The 75 new or changed routines are :

arp/phys_dmn/aplpar.F90
arp/phys_dmn/suphmse.F90
arp/programs/master.F90
mse/externals/aro_surf_diag.F90
mse/externals/aroini_surf.F90
mse/externals/end_oasis3_sfx.F90
mse/externals/ini_oasis3_sfx.F90
mse/externals/mse_xios_field_is_active.F90
mse/externals/mse_xios_declare.F90
mse/externals/mse_xios_send.F90
mse/externals/send_oasis3_sfx.F90
mse/internals/aroopen_namelist.F90
mse/internals/read_namelists_io.F90
mse/internals/sfx_xios_readnam_aro.F90
mse/internals/sfx_xios_setup_aro.F90
mse/module/modd_io_surf_aro.F90
surfex/SURFEX/close_aux_io_surf.F90
surfex/SURFEX/close_namelist.F90
surfex/SURFEX/coupling_seafluxn.F90
surfex/SURFEX/dealloc_surf_atmn.F90
surfex/SURFEX/detect_field.F90
surfex/SURFEX/end_io_surfn.F90
surfex/SURFEX/get_default_namn.F90
surfex/SURFEX/get_luout.F90
surfex/SURFEX/get_size_fulln.F90
surfex/SURFEX/init_io_surfn.F90
surfex/SURFEX/init_read_data_cover.F90
surfex/SURFEX/init_surf_atmn.F90
surfex/SURFEX/modd_sfx_oasis.F90
surfex/SURFEX/modd_surf_atm_gridn.F90
surfex/SURFEX/modd_xios.F90
surfex/SURFEX/mode_pos_surf.F90
surfex/SURFEX/open_aux_io_surf.F90
surfex/SURFEX/open_namelist.F90
surfex/SURFEX/read_all_namelists.F90
surfex/SURFEX/read_buffer.F90
surfex/SURFEX/read_default_flaken.F90
surfex/SURFEX/read_default_surf_atmn.F90
surfex/SURFEX/read_grid.F90
surfex/SURFEX/read_surf.F90
surfex/SURFEX/set_axis.F90
surfex/SURFEX/set_surfex_filein.F90
surfex/SURFEX/sfx_oasis_end.F90
surfex/SURFEX/sfx_oasis_init.F90
surfex/SURFEX/sfx_xios_check_field.F90
surfex/SURFEX/sfx_xios_check_field_2d.F90
surfex/SURFEX/sfx_xios_declare_field.F90
surfex/SURFEX/sfx_xios_set_domain.F90
surfex/SURFEX/sum_on_all_procs.F90
surfex/SURFEX/test_nam_var_surf.F90
surfex/SURFEX/write_cover_tex_end.F90
surfex/SURFEX/write_cover_tex_start.F90
surfex/SURFEX/write_diag_misc_isban.F90
surfex/SURFEX/write_diag_pgd_isban.F90

surfex/SURFEX/write_diag_seb_flaken.F90	surfex/SURFEX/write_surf.F90	surfex/SURFEX/sfx_xios_send_block.F90
surfex/SURFEX/write_diag_seb_isban.F90	surfex/SURFEX/write_surf_xios.F90	surfex/SURFEX/sfx_xios_setup.F90
surfex/SURFEX/write_diag_seb_seafluxn.F90	surfex/SURFEX/writesurf_gr_snow.F90	surfex/GELATO/modi_glt_vhdiff_r.F90
surfex/SURFEX/write_diag_seb_surf_atmn.F90	surfex/SURFEX/writesurf_isban.F90	surfex/OFFLIN/modn_io_offline.F90
surfex/SURFEX/write_grid.F90	surfex/SURFEX/writesurf_seafluxn.F90	surfex/OFFLIN/offline.F90
	surfex/SURFEX/writesurf_seaicen.F90	surfex/OFFLIN/prep.F90
	surfex/SURFEX/writesurf_watfluxn.F90	surfex/OFFLIN/sfx_xios_readnam_ol.F90
	surfex/SURFEX/writesurf_atm_confn.F90	surfex/OFFLIN/sfx_xios_setup_ol.F90

5.3. Structure of calls implying XIOS

The following call tree can be helpful for a deep understanding of XIAS and for its maintenance

```
Appels et routines de l'interfacage de XIOS à Surfex en Offline ou dans Arpege/Aladin

Note : le présent texte est organisé pour faciliter l'accès aux
fichiers sources qui sont cités, que ce soit
  - sous Emacs avec la fonction 'file-find-at-point',
  - ou sous vi avec la focntion 'gf'
dès lors que le présent fichier est situé dans le répertoire src/local
du pack Arpege qui représente les sources de l'ajout XIOS, ou dans le
répertoire src/main d'un pack maitre qui les contient

-Offline

surfex/OFFLIN/offline.F90
  -> surfex/SURFEX/sfx_oasis_init.F90
       -> xios_initialize
  -> surfex/OFFLIN/sfx_xios_setup_ol.F90
     -> surfex/SURFEX/sfx_xios_setup.F90
       -> xios_context_initialize, xios_set_current_context(sfx), xios_solve_inheritance()
       -> surfex/SURFEX/sfx_xios_set_domain.F90
                   definition des domaines 'full' et tiles, via xios_set_domain_attr
   -> xios_update_calendar (avant appel de diag_surf_atm)
   -> surfex/SURFEX/write_surf.F90 : idem Arpege , cf infra
   -> xios_close_context_definition.F90 (après le 1° appel à write_diag_surf_atm)
   -> xios_finalize.F90


Arpege

arp/programs/master.F90
  -> mse/externals/ini_oasis3_sfx.F90
    -> mse/internals/sfx_xios_readnam_aro.F90 : lecture flag de namelist
    -> surfex/SURFEX/sfx_oasis_init.F90 :
        xios_initialize, init Oasis, et init MPI (via Oasis ou XIOS)
  -> surfex/SURFEX/sfx_oasis_end.F90
     -> xios_finalize.F90


Surfex dans Arpege (MSE)

mse/externals/aroini_surf.F90
  -> intialisation du tableau d'index pour XIOS (similaire à Oasis)
  -> mse/internals/sfx_xios_setup_aro.F90
    -> surfex/SURFEX/sfx_xios_setup.F90
       -> xios_context_initialize, xios_set_current_context(sfx),xios_solve_inheritance
       -> xios_set_timestep, xios_define_calendar
       -> surfex/SURFEX/sfx_xios_set_domain.F90
                   definition des domaines 'full' et tiles, via xios_set_domain_attr

mse/externals/aro_surf_diag.F90, dans la boucle sur les blocs pour écrire
  => sur bloc 0 (phase inactive/ de comptage (LCOUNTW))
      => si on est phase init_xios : write_surf bidon pour declarer les champs :
      !   -> write_surf_atm_n.F90 (et write_diag_surf_atm_n)
      !      -> surfex/SURFEX/init_io_surfn.F90 : positionne YXIOS_DOMAIN
      !           (utile en phase de déclaration des champs à XIOS)
      !      -> surfex/SURFEX/write_surf.F90 (en boucle sur les champs)
      !        -> surfex/SURFEX/write_surf_xios.F90 :
      !          -> surfex/SURFEX/sfx_xios_declare_field.F90
      !   -> xios_close_context_definition.F90 (et met fin a la phase d'init xios)
      => sinon (i.e. si l'init d'Xios est fini)
      !   -> xios_set_current_context (quand LCOUNTW)
      !   -> xios_update_calendar
  => sur autres blocs
        -> surfex/SURFEX/write_surf_atmn.F90 et write_diag_surf_atm_n
```

```
            via une cascade d'appels propre à Surfex , on arrive à
            -> surfex/SURFEX/write_surf.F90
              -> surfex/SURFEX/write_surf_xios.F90 : si champ OK pour XIOS :
                       -> surfex/SURFEX/sfx_xios_send_block.F90
                          -> xios_send_field, automatique en fin de boucle sur les blocs
 -> xios_context_finalize (au dernier pas de temps)

arp/phys_dmn/suphmse.F90  (pourrait etre en fin de suphmf ou de aroini_surf)
    -> mse/externals/mse_xios_declare.F90

Dans la physique Surfex, on peut utiliser xios_send_block:
surfex/SURFEX/sfx_xios_send_block.F90
    -> surfex/SURFEX/sfx_xios_declare_field.F90
       -> xios_send_field
Exemple, dans coupling_seafluxn : CALL SFX_XIOS_SEND_BLOCK("zqsat",ZQSAT,CDOMAIN="SEA")

Dans la physique Arpege, on peut utiliser :
mse/externals/mse_xios_field_is_active.F90  (pour savoir si un diag est à calculer)

mse/externals/mse_xios_send.F90  ( pour livrer un champ)
    -> surfex/SURFEX/sfx_xios_send_block.F90
       -> surfex/SURFEX/sfx_xios_declare_field.F90
       -> xios_send_field
```

5.4. Compilation

The XIOS code has been included in Surfex repository and in Surfex build system in order to simplify the installation of an XIOS-enabled Surfex. This implied designing specific FCM configuration files, and letting them inherit the value of compile-time varibale (compiler, Netcdf libraries ...)

For both models (Arpege and Surfex), the code for activating the interface is controlled using a pre-processing flag WXIOS, which could be ultimately discarded once the above Xios build si fully operative on all Surfex platforms, and a dummy library of XIOS API is created for Arpege.

5.5. OpenMP

XIOS is yet not thread-safe; therefore, calls to the XIOS API are protected by an "OMP SINGLE" directive

5.6. Input using XIOS

XIOS is here yet used only for the output. Nevertheless, it includes features for reading files in a distributed way, and with read-ahead capability; this feature could be much useful for reading evolving Lateral Boundary Conditions or nudging fields, including for normalizing the format of such data to NetCDF. All processing filters can be applied to input streams. Using XIOS input scheme for restart files could also be considered

5.7. Tests extent :

xxx inclure README

xxx : Arpege vs Arpege-Climat

Appendix 1 : Default Surfex configuration file

Appendix : mutliple time coordinates

This is an example of