

How to configure output files in ORCHIDEE

Presentation by Josefine Ghattas, IPSL
ORCHIDEE Training course 30 november 2017

Files created by ORCHIDEE

Restart files

- Containing all state variables in ORCHIDEE at the last time step of the execution
- These files are needed as input to start next iteration
- driver_rest_out.nc, sechiba_rest_out.nc, stomate_rest_out.nc

Diagnostic output files

- Optional files containing variables from ORCHIDEE
- One file per frequency, different operations possible
- As many files as wanted, as many variables as wanted
- For example sechiba_history.nc, stomate_history.nc,...

Output diagnostic files by XIOS

Thanks to

- Yann Meurdesoif, LSCE, the main developer of XIOS
- Arnaud Cael, LSCE who did the first implementation of XIOS in ORCHIDEE

XIOS is a tool developed for the IPSL modèles to obtain better performances and more flexible management of output files. In a near future, XIOS will also handle reading of input files in ORCHIDEE.

In ORCHIDEE:

- `src_parallel/xios_orchidee.f90` : all interfacing to XIOS
- `src_xml` : directory with all xml files for running with XIOS
- Flag **XIOS_ORCHIDEE_OK** to activate running with XIOS
- Preprocessing key **XIOS** to enable linking to XIOS

Previous method with IOIPSL

IOIPSL is a fortran library developed at IPSL, used since long time in ORCHIDEE for

- reading input files (call flinopen, call flinget)
- reading and writing restart files (call restget, restput)
- reading parameter files run.def (call getin)
- writing output files with diagnostic variables (call histdef, call histwrite)

(*) The default method for writing output files is now changed to use XIOS. The method using IOIPSL is still left in ORCHIDEE but new variables are not added. This possibility will be removed in coming version (date not yet decided).

(**) The method for reading and interpolating input files will be change to use XIOS in a near future.

xios_orchidee_send_field

Example from thermosoil_main:

```
USE xios_orchidee
```

```
REAL(r_std), DIMENSION (kjpindex)      :: soilflx  
REAL(r_std), DIMENSION (kjpindex)      :: surfheat_incr  
REAL(r_std), DIMENSION (kjpindex, ngrnd) :: ptn  
...  
  
CALL xios_orchidee_send_field("ptn", ptn)  
CALL xios_orchidee_send_field("Qg", soilflx)  
CALL xios_orchidee_send_field("DelSurfHeat", surfheat_incr)
```

Syntax: **CALL xios_orchidee_send_field(field_id, field)**

field_id: a unique identifier, the same id is set in the field definition in parameter file field_def_orchidee.xml which must be present at run time
CHARACTER(len=*)

field: the variable to send to XIOS. The variable is on landpoint grid, it can have 1 or 2 supplementary axis:
REAL(r_std), DIMENSION(kjpindex) or
REAL(r_std), DIMENSION(kjpindex,:)

Convention in ORCHIDEE : use the same name for the id as the variable name

xml parameter files

To run ORCHIDEE with XIOS all diagnostic output files are configured through xml files. Following 4 files need to be present at each execution :

- **iodef.xml** Main input file for XIOS
- **context_orchidee.xml** Grid and axis information, include field and file def
- **field_def_orchidee.xml** => **Definition for each variable send in ORCHIDEE**
=> Only change if added new variable in ORCHIDEE
- **file_def_orchidee.xml** => **Specify all output files and their variables**
=> **Change to set your output level**
=> **Remove variables, change levels, change freq...**
- **file_def_input_orchidee.xml** => **Specify all input files read by XIOS**
=> **This file is needed but reading of files are currently deactivated. Do not modify!**

The above xml files are stored in ORCHIDEE/src_xml directory.

file_def_orchidee.xml

```
<!-- ===== -->
<!-- file_def_orchidee.xml : Definition of output files -->

<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">

  <!-- Sechiba file 1 -->
  <file id="sechiba1" name="sechiba_history" output_level="11" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="evap" level="1"/>
    <field field_ref="coastalflow" level="1"/>
    <field field_ref="riverflow" level="2"/>
    <field field_ref="temp_sol_C" level="2"/>
    ...
  </file>

  <!-- Sechiba file 2 -->
  <file id="sechiba2" name="sechiba_out_2" output_level="2" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="mrsos" level="1"/>
    <field field_ref="mrro" level="2"/>
    ...
  </file>

  <!-- Stomate file 1 -->
  <file id="stomate1" name="stomate_history" output_level="10" output_freq="86400s">
    <field field_ref="RESOLUTION_X" level="1"/>
    <field field_ref="RESOLUTION_Y" level="1"/>
    <field field_ref="CONTFRAC_STOMATE" level="1"/>
  </file>
</file_definition>
```

file_def_orchidee.xml

```
<!-- ===== -->  
<!-- file_def_orchidee.xml : Definition of output files -->
```

```
<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">
```

```
<!-- Sechiba file 1 -->
```

```
<file id="sechiba1" name="sechiba_history" output_level="11" output_freq="1d" enabled=".TRUE.">
```

```
<field field_ref="Areas" level="1"/>
```

```
<field field_ref="LandPoints" level="1"/>
```

```
<field field_ref="...
```

```
<field field_ref="...
```

```
<field field_ref="...
```

```
<field field_ref="...
```

```
<field field_ref="...
```

```
<field field_ref="...
```

```
</file>
```

Information about all files written by ORCHIDEE

type "one_file" or "multiple_file" : XIOS will gather information from all processes on a single output file or not

```
<!-- Sechiba file 2 -->
```

```
<file id="sechiba2" name="sechiba_out_2" output_level="2" output_freq="1d" enabled=".TRUE.">
```

```
<field field_ref="Areas" level="1"/>
```

```
<field field_ref="LandPoints" level="1"/>
```

```
<field field_ref="Contfrac" level="1"/>
```

```
<field field_ref="mrsos" level="1"/>
```

```
<field field_ref="mrro" level="2"/>
```

```
</file>
```

```
<!-- Stomate file 1 -->
```

```
<file id="stomate1" name="stomate_history" output_level="10" output_freq="86400s">
```

```
<field field_ref="RESOLUTION_X" level="1"/>
```

```
<field field_ref="RESOLUTION_Y" level="1"/>
```

```
<field field_ref="CONTRAC_STOMATE" level="1"/>
```

```
</file>
```

```
</file_definition>
```


file_def_orchidee.xml

```
<!-- ===== -->
<!-- file_def_orchidee.xml : Definition of output files -->

<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">

  <!-- Sechiba file 1 -->
  <file id="sechiba1" name="sechiba_history" output_level="11" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="evap" level="1"/>
    <field field_ref="coastalflow" level="1"/>
    <field field_ref="riverflow" level="2"/>
    <field field_ref="temp_sol_C" level="2"/>
    ...
  </file>

  <!-- Sechiba file 2 -->
  <file id="sechiba2" name="sechiba_out_2" output_level="2" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="mrsos" level="1"/>
    <field field_ref="mrro" level="2"/>
    ...
  </file>

  <!-- Stomate file 1 -->
  <file id="stomate1" name="stomate_history" output_level="10" output_freq="86400s">
    <field field_ref="RESOLUTION_X" level="1"/>
    <field field_ref="RESOLUTION_Y" level="1"/>
    <field field_ref="CONTFRAC_STOMATE" level="1"/>
  </file>
</file_definition>
```

file_def_orchidee.xml

```
<!-- ===== -->
<!-- file_def_orchidee.xml : Definition of output files -->

<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">
```

```
<!-- Sechiba file 1 -->
```

```
<file id="sechiba1" name="sechiba_history" output_level="11" output_freq="1d" enabled=".TRUE.">
```

```
<field field_ref="Areas" level="1"/>
<field field_ref="LandPoints" level="1"/>
<field field_ref="Contfrac" level="1"/>
<field field_ref="evap" level="1"/>
<field field_ref="coastalflow" level="1"/>
<field field_ref="riverflow" level="2"/>
<field field_ref="temp_sol_C" level="2"/>
```

```
...
</file>
```

```
<!-- Sechi
```

```
<file id="
```

```
<field f
```

```
<field f
```

```
<field f
```

```
<field f
```

```
<field f
```

```
...
</file>
```

```
<!-- Stoma
```

```
<file id="
```

```
<field f
```

```
<field f
```

```
<field f
```

```
</file>
```

```
</file_defin
```

Information line about one file

name	filename, suffix .nc will be added to the filename
output_level	"x" : all variables listed below with level less or equal to x will be written
output_freq	"1d", "1800s", "1ts", "1mo", "3h", "1y" : frequency for the file
enabled	".TRUE." / ".FALSE." : create the file, true is default

file_def_orchidee.xml

```
<!-- ===== -->
<!-- file_def_orchidee.xml : Definition of output files -->

<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">

  <!-- Sechiba file 1 -->
  <file id="sechiba1" name="sechiba_history" output_level="11" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="evap" level="1"/>
    <field field_ref="coastalflow" level="1"/>
    <field field_ref="riverflow" level="2"/>
    <field field_ref="temp_sol_C" level="2"/>
    ...
  </file>

  <!-- Sechiba ... -->
  <file id="sechiba2" name="sechiba2" output_level="11" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="evap" level="1"/>
    <field field_ref="coastalflow" level="1"/>
    <field field_ref="riverflow" level="2"/>
    <field field_ref="temp_sol_C" level="2"/>
    ...
  </file>

  <!-- Stomate ... -->
  <file id="stomate" name="stomate" output_level="11" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="evap" level="1"/>
    <field field_ref="coastalflow" level="1"/>
    <field field_ref="riverflow" level="2"/>
    <field field_ref="temp_sol_C" level="2"/>
    ...
  </file>
</file_definition>
```

A line per variable added in the file

field_ref	reference id as set in field_def_orchidee.xml file
level	“x” : the variable is only written if this level is less or equal of output_level set at the file description line above.
name / long_name	“new_name” : name of the variable in the output file. If it is not set, the name set in field_def_orchidee.xml will be used.
enabled	“.TRUE.” / “.FALSE.” : write the variable, true is the default.
operation	can be added, overwrites settings in field_def “average”, “min”, “max”, “instant”

Add a new variable in ORCHIDEE

1) Add in the ORCHIDEE module where the variable is calculated:

CALL xios_orchidee_send_field("new_var",new_var)

2) In **field_def_orchidee.xml** : add declaration of the variable

3) In **file_def_orchidee.xml** : add the variable in all file sections where you want to write it

*) If the variable is only calculated for a specific option, add an exception in **xios_orchidee_init**. This avoid that the variable will be initialized in the output file without being written if you keep the same .xml files.

Running in attached mode

Requirements for running ORCHIDEE with XIOS in attached mode:

- **1 executable: orchidee_ol**
- 5 xml files : iodef.xml, context_orchidee.xml,
field_def_orchidee.xml,
file_def_orchidee.xml, file_def_input_orchidee.xml
- Parameter file: run.def
- Input netcdf files: forcing_file.nc, PFTmap.nc, ...

It is possible to run on 1 or several MPI

Note: You can copy xml files from ORCHIDEE/src_xml

Running with server

Requirements for running ORCHIDEE with XIOS using server:

- **2 executables** : **orchidee_ol** and **xios_server.exe**
- And all other input files as before:

Note : the 2 executables must be launched together in Multiple Program Multiple Data (MPMD) mode. MPI is used for the communications between the executables.

For example:

For ORCHIDEE in offline mode, 2-degree global resolution, we advice

- 31 MPI for orchidee_ol executable
- 1 MPI for xios_server.exe

Use at least 2 MPI for orchidee_ol and 1 for xios_server.exe

Running with server

Requirements for running ORCHIDEE with XIOS using the server

- **2 executables** : orchidee_ol and xios_server.exe

And other input files as before:

- 5 xml files : iodef.xml, context.xml, file_def_orchidee.xml, file_def_orchidee_server.xml, file_def_orchidee_client.xml
- Parameter file : run_ol_server.xml
- Input netcdf files

- More efficient, we want this method
- Not so easy to write the run script
- Different on each platform

⇒ USE LIBIGCM

For

For C... solution, we advice

- 31 M...
- 1 MF...

Use at least 2 for orchidee_ol and 1 for xios_server.exe

Using libIGCM configurations server mode by default

Default mode is using 1 server XIOS in libIGCM configurations (ORCHIDEE_trunk, LMDZOR_v6, IPSLCM6)

config.card

- Component IOS represents XIOS
- Set number of cores MPI for each executable with 1MPI for the xios server.

```
#=====
#D-- ListOfComponents -
[ListOfComponents]
#D- For each component, Name of component, Tag of component
SRF= (sechiba, orchidee_trunk)
SBG= (stomate, orchidee_trunk)
OOL= (orchidee_ol, orchidee_trunk)
IOS= (xios, XIOS)

#D-- Executable -
[Executable]
#D- For each component, Real name of executable
SRF= ("", "")
SBG= ("", "")
OOL= (orchidee_ol, orchidee_ol, 31MPI)
IOS= (xios_server.exe, xios.x, 1MPI)

...

#D-- IOS -
[IOS]
WriteFrequency=""
Restart= n
RestartDate=
RestartJobName=
RestartPath=
^ ^ ^
```

Number of core MPI
per executable

Using libIGCM configurations

control of output

- All xml files except iodef.xml are copied from source directory ORCHIDEE/src_xml
- Use **WriteFrequency** in **config.card** to control the frequency and activation of predefined files

or

- Change directly in **ORCHIDEE/src_xml/file_def_orchidee.xml**
Keyword AUTO is changed by libIGCM but can be changed manually as well.

Questions?

Learn better by do it your self
during the hands on session !