

Standard Running Environment (SRE) within IMDI

ISSMES 07

Heinrich Widmann

3. September 2007

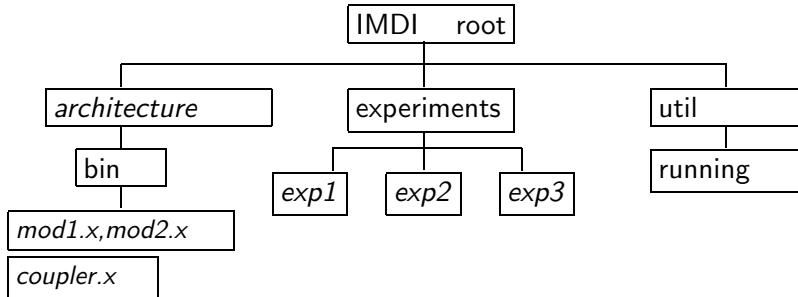
Outline

- 1 Introduction to SRE
 - Components of SRE
 - Standardized Directory Structure
 - Design of an IMDI Experiment
 - Experiment Flow within SRE
- 2 Performing an Experiment
 - Setup and Configuration
 - Generation of Tasks
 - Running a Model
- 3 IT Environment
 - SRE File Systems
 - Models and Platforms supported by SRE
- 4 Summary

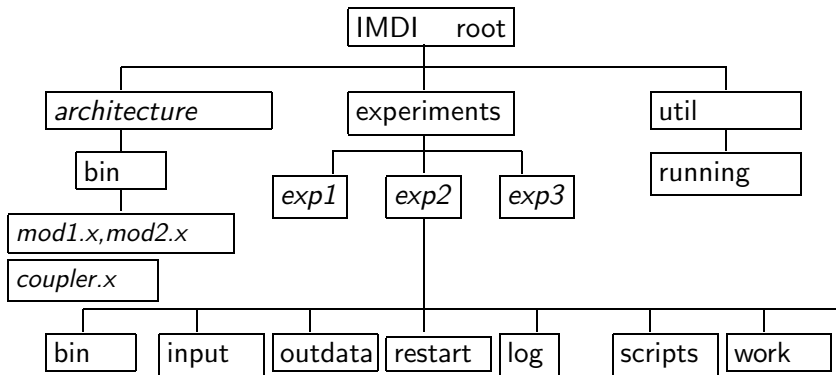
Components of SRE

- Standardized Directory Structure contains
 - the executables
 - the input, output and restart files
 - tools and header files for script generation
 - setup files for experiment configuration
 - directories to run the experiment
- m4 and ksh scripts to generate scripts from a
 - common base of site and model specific header files
- Computing and archiving facilities

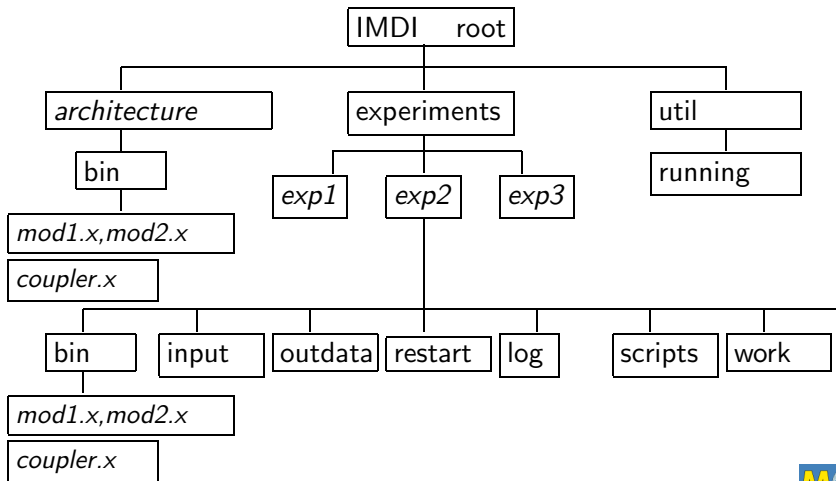
Standardized Directory Structure



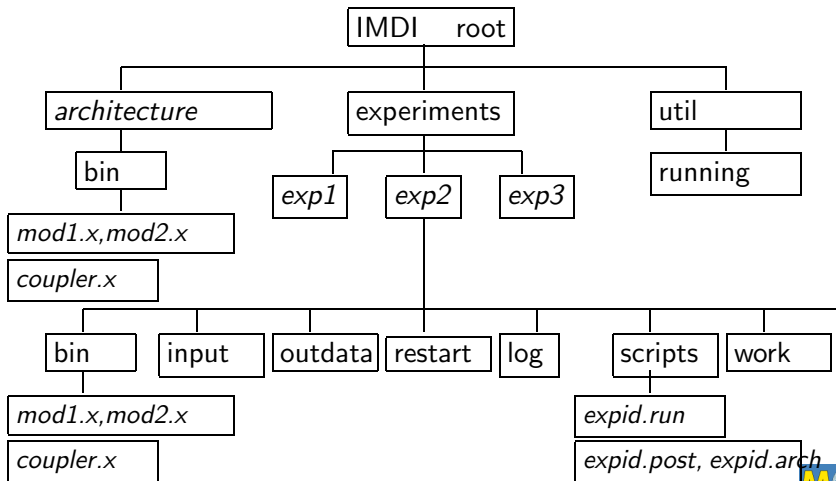
Standardized Directory Structure



Standardized Directory Structure



Standardized Directory Structure

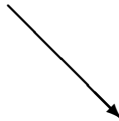


Design of an IMDI Experiment

- 1 **Retrieving sources** of *models* and *IMDI* from SVN
- 2 **Compiling the model** e.g. within *SCE*
 - Optionally : Modify source code
 - Create compile scripts and makefiles (*SCE*)
 - Compile the model components \Rightarrow **executables**
- 3 **Performing an experiment** within *SRE*
 - Configure and set up your experiment
 - Create tasks and run scripts
 - Run experiment
- 4 **Analysing** results

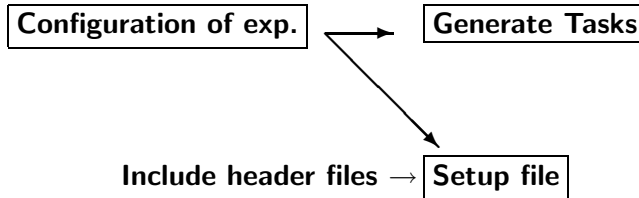
Experiment Flow within SRE

Configuration of exp.

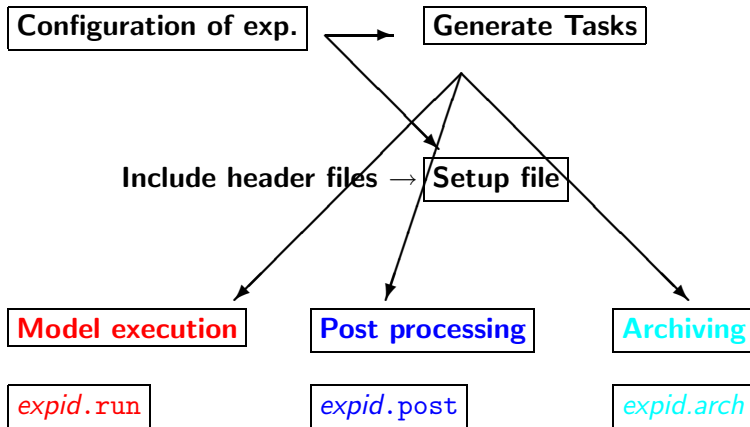


Include header files → **Setup file**

Experiment Flow within SRE



Experiment Flow within SRE



Setup and Configuration

- \$ `cd imdiroot/util/running/tools/`
- \$ `./Create_TASKS.frm --help`
- \$ `./Create_TASKS.frm cplmod explD [node]`
 - ⇒ m4 macro processor generates **setup file**
`imdiroot/util/running/setup/setup_cplmod_explD`
 - ⇒ `check_setup_cplmod_explD` checks the consistency of the setup file
- Change settings by editing the setup file for your needs, i.e. **configure your experiment**

Structure of the setup

The setup file is composed of several header files, partly depending on

- the coupled model combination (**cplmod**)
- the component model (**model**) or
- the site (**node**).

config_experiments.h
config_**cplmod**.h
config_timecontrol.h
config_postprocessing.h
config_postprocessing_**model**.h
config_archive.h
config_visualization.h
config_visualization_**model**.h
config_mpi_**node**.h
config_filesystem.h
config_site_**node**.h
config_commands_**node**.h
config_setup_**cplmod**.h

Generation of the Tasks

- ① `./Create_TASKS.frm cplmod explD [node]` : first call \Rightarrow only **setup file** is created (see above)
- ② `./Create_TASKS.frm cplmod explD [node]` : subsequent call(s), i.e. setup file already created
 - \Rightarrow m4 macro processor generates **tasks**, i.e. corresponding \rightarrow *scripts* by including again *model* and *site* specific header files
 - Model integration \rightarrow *explD.run*
 - Output data postprocessing \rightarrow *explD.post*
 - Archiving of raw and processed output \rightarrow *explD.arch*

Running a Model

- Submit the runscript in the job directory:
 - `$ cd imdirroot/experiments/expid/scripts`
 - check existence of run scripts, i.e. `expid.run`, `expid.post` and `expid.arch`
 - `$ rm expid.date`, if exists and for initial run
 - `$ qsub expid.run`
Request 49733.siox3 submitted to queue: pipe.
 - Control and monitor your run

Example : Experiment *sus0001* - Settings

Main settings in the setup file and in the run scripts

- `expid=sus0001`
- `cplmod=cosmos-asob`
- `queue=default` # **change to queue s4 !!**
- `email=your@email` # change to your email address
- `initial_date=0800-01-01` (initial date of experiment)
- `final_date=0820-12-31` (final date of experiment)
- \Rightarrow Integration periode $IP = 21$ years
- `nyear=1` (yearly runs \Rightarrow run periode $RP = 1$ year)

Example : Experiment *sus0001* - Workflow

```
$ qsub sus0001.run =>
```

IP = 21 RPs = 21 years

→ model time



Example : Experiment *sus0001* - Workflow

```
$ qsub sus0001.run ⇒
```

sus0001.run(RP1)

IP = 21 RPs = 21 years

model time



Example : Experiment *sus0001* - Workflow

```
$ qsub sus0001.run ⇒
```



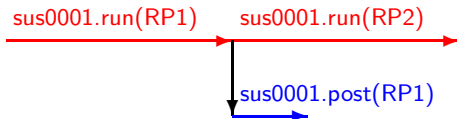
IP = 21 RPs = 21 years

model time



Example : Experiment *sus0001* - Workflow

```
$ qsub sus0001.run =>
```

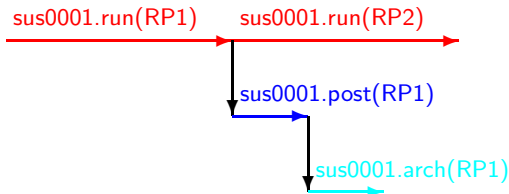


IP = 21 RPs = 21 years

model time

Example : Experiment *sus0001* - Workflow

```
$ qsub sus0001.run =>
```

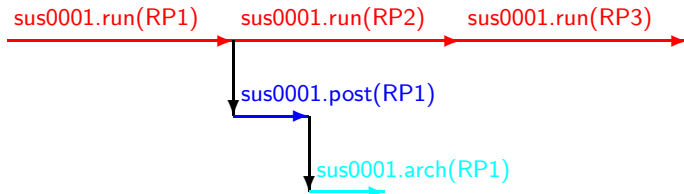


IP = 21 RPs = 21 years

model time

Example : Experiment *sus0001* - Workflow

```
$ qsub sus0001.run =>
```

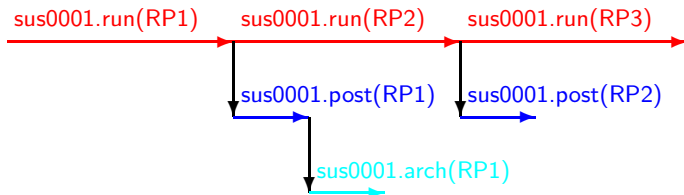


IP = 21 RPs = 21 years

model time

Example : Experiment *sus0001* - Workflow

```
$ qsub sus0001.run =>
```

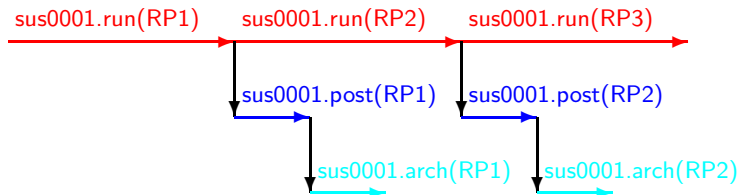


IP = 21 RPs = 21 years

model time

Example : Experiment *sus0001* - Workflow

```
$ qsub sus0001.run =>
```



IP = 21 RPs = 21 years

model time

Monitoring your run(s) and jobs

Use `qstat` to get the actual status of your job(s)

```
$ qstat
```

```
RequestID ReqName  UserName  Queue  Pri  STT  S  Memory  CPU  Elapse  R  H  M  Jobs  
-----  
49733.siox3 sus0001. k204019 d0 0  QUE - 0.00B 0.00 0 Y Y Y 1
```

Monitoring your run(s) and jobs

Use `qstat` to get the actual status of your job(s)

```
$ qstat
```

```
RequestID ReqName  Username Queue Pri STT S Memory CPU Elapse R H M Jobs  
-----  
49733.siox3 sus0001. k204019 d0 0 QUE - 0.00B 0.00 0 Y Y Y 1
```

```
$ qstat
```

```
RequestID ReqName  Username Queue Pri STT S Memory CPU Elapse R H M Jobs  
-----  
49733.siox3 sus0001. k204019 d0 0 RUN - 0.00B 0.00 4 Y Y Y 1
```

Monitoring your run(s) and jobs

Use `qstat` to get the actual status of your job(s)

```
$ qstat
RequestID ReqName  UserName  Queue Pri STT S Memory CPU Elapse R H M Jobs
-----
49733.siox3 sus0001. k204019 d0 0 QUE - 0.00B 0.00 0 Y Y Y 1
$ qstat
RequestID ReqName  UserName  Queue Pri STT S Memory CPU Elapse R H M Jobs
-----
49733.siox3 sus0001. k204019 d0 0 RUN - 0.00B 0.00 4 Y Y Y 1
49734.siox3 sus0001. k204019 iseri 0 RUN - 26.47M 4.82 885 Y Y Y 1
$ qstat -f jobid
$ qstat -Q
```

Monitoring your Experiment

Check the SRE logfiles to see the status of your experiment

```
$ less sus0001.log  
Sat Sep 1 13:41:33 DST 2007 : Beginning of Experiment sus0001  
Sat Sep 1 13:41:33 DST 2007 : 1 08000101 0:49733.siox3 - start  
Sat Sep 1 15:06:56 DST 2007 : 1 08010101 0:49733.siox3 - done
```

Monitoring your Experiment

Check the SRE logfiles to see the status of your experiment

```
$ less sus0001.log
Sat Sep 1 13:41:33 DST 2007 : Beginning of Experiment sus0001
Sat Sep 1 13:41:33 DST 2007 : 1 08000101 0:49733.siox3 - start
Sat Sep 1 15:06:56 DST 2007 : 1 08010101 0:49733.siox3 - done
$ less sus0001.date
08010101 2
```

SRE File Systems (FS)

Process	SRE FS	Description	for ISSMES@DKRZ
Execution	home	submit jobs	<i>imdiroot/experiments</i>
	work	temporary file system	<i>\$WRKSHR/cosmos-issmes/experiments</i>
Postproc.	data	medium term storage	<i>\$WRKSHR/cosmos-issmes/experiments</i>
Archiving	archive_in	input data archive	<i>/pool/SX-6/COSMOS</i>
	archive	long term storage	<i>\$PRJHOME/arch/uid/cosmos-issmes/experiments</i>

Models supported by SRE

(Coupled) Model	Model components			
	Atmosphere	Biosphere	Ocean	BGC Ocean, Chemistry or ...
COSMOS-ASOB	ECHAM5	JSBACH	MPIOM	HAMOCC
COSMOS-ASO	ECHAM5	JSBACH	MPIOM	
COSMOS-AOB	ECHAM5		MPIOM	HAMOCC
COSMOS-AO	ECHAM5		MPIOM	
COSMOS-A	ECHAM5			
TOYCLIM	TOYATM		TOYOCE	TOYCHE
CLM	Standalone regional climate model			

Platforms supported by SRE

Site/OS (<i>node</i>)	Architecture	Comments
ds (cross)	NEC SX6	cross system provided by DKRZ
linux86	Linux PCs	e.g. workstations at ZMAW
linux-x64	64 Bit Linux	linux cluster <i>tornado</i> at ZMAW (in progress)
sunos	Sun Solaris	SUN server at ZMAW (in work)
crayx1	CRAY	other sides (in work)
aix	IBM Linux	e.g. at ECMWF (in work)

Summary and Outlook

- The Standard Run Environment (SRE)
 - provides a suite of tools to setup, run and monitor experiments of coupled models
 - is easy to use (look and feel)
 - is adaptable to new models and platforms
- Outlook (ongoing work)
 - Postprocessing for MILLENNIUM project (extra postprocessing resp. data server)
 - 'Automatic' data base filling
 - New monitoring task (*expid.mon*) : timeseries of choosen variables are plotted per run periode and put on webserver during run time

Documentation

- imdiroot/util/running/doc/prism_rep05.pdf
- <http://www.mad.zmaw.de/imdi> (at bottom of page)
- <http://www.dkrz.de/dkrz/services/docs/intro> (e.g. for NQS queueing system and other HPC facilities provided by DKRZ)

Thank you for your attention !