The Common Community Physics Package

Dustin Swales[1,2], Dom Heinzeller[4,7], Ligia Bernardet[1,2], Soren Rasmussen[2,5], Michael Kavulich[2,5], Courtney Peverley[6], Grant Firl[1,2,3], Tracy Hertneky[2,5], Lulin Xue[2,5]

[1]NOAA Global Systems Laboratory, [2]Developmental Testbed Center, [3]Colorado State University Cooperative Institute for Research in the Atmosphere, [4]U.S. Naval Research Laboratory / [7]UCAR Cooperative Programs for the Advancement of Earth System Science, [5]NSF NCAR-Research Applications Laboratory, [6]NSF NCAR-Climate and Global Dynamics Laboratory

Overview

The Common Community Physics Package (CCPP) consists of a library of physical parameterizations, the ***CCPP Physics***, and a software infrastructure, the ***CCPP Framework***, used to couple the physics to a host application. It serves as a core component of multiple numerical weather prediction systems, including NOAA's Unified Forecast System and the US Navy's NEPTUNE. Furthermore, additional modeling centers are actively adopting the CCPP, such as NSF NCAR with its CAM-SIMA model.

Fostering Collaboration with Common Infrastructure

The CCPP Physics library contains physical parameterizations that are used across a wide range of forecast applications (e.g., SPS -> S2S). Physics innovations developed for one application can be adapted for any other CCPP enabled application.

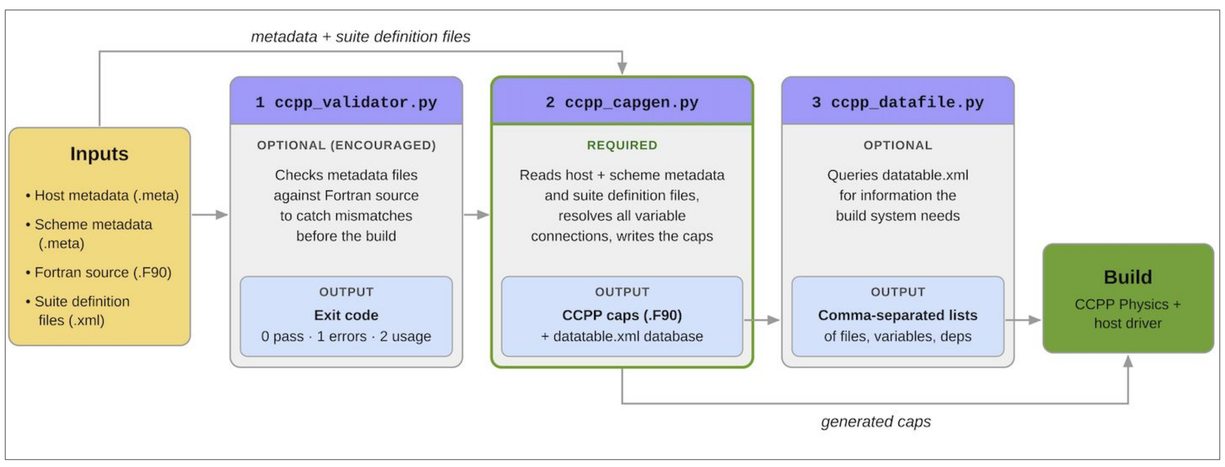


Figure 1. The three primary CCPP Capgen scripts. `ccpp_validator.py` (optional) checks metadata against Fortran source. `ccpp_capgen.py` (required) generates the CCPP caps and `datatable.xml`. `ccpp_datafile.py` (optional) queries `datatable.xml` for the file lists the host build system uses.

Metadata

Each ***CCPP compliant scheme*** comes with an ***accompanying metadata file*** that describes the arguments in detail. On the host side, each variable used by the CCPP must also come with supporting metadata.

These metadata files, along with the source files, are ***validated*** against each other using a host-provided physics configuration file (.xml). Upon completion, the Framework creates the ***"physics driver(s)"*** or ***"cap(s)"*** called from within the atmospheric model.

Suite Definition Files

The "recipe" for the physics driver is controlled by the CCPP ***Suite Definition File (SDF)***, an .xml file containing the physics groups, schemes, ordering, and, if provided, subcycling, to be written into the autogenerated cap.

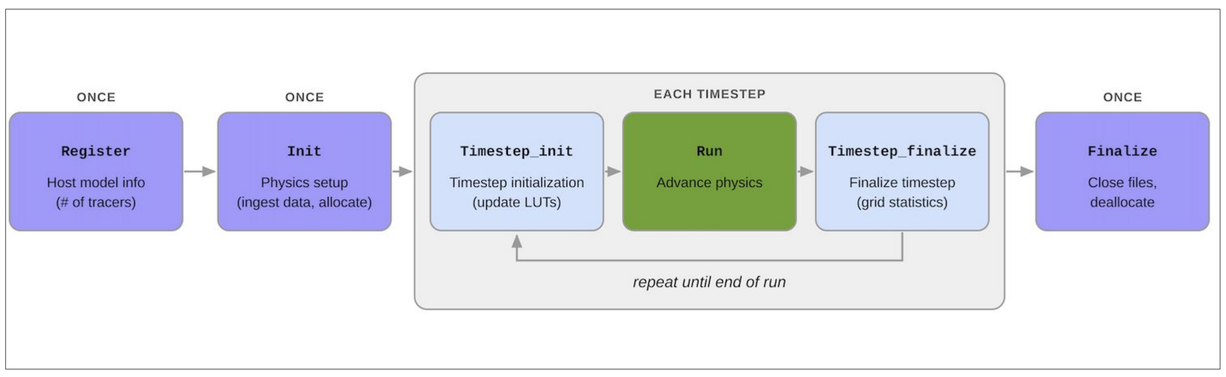


Figure 2. The six CCPP runtime phases. Register, Init and Finalize (purple) are each called once per run, for host model registration, physics setup and cleanup. Timestep_init, Run and Timestep_finalize are called every timestep (gray box) and repeat until the end of the run.

UFS_MPAS_regional

```
<suite name="UFS_MPAS_regional">
  <group name="time_vary">
    <scheme>time_vary_pre</scheme>
  </group>

  <group name="radiation">
    <scheme>rrtmg_lw</scheme>
    <scheme>rrtmg_sw</scheme>
  </group>

  <group name="physics">
    <subcycle loop="2">
      <scheme>sfc_layer</scheme>
      <scheme>lsm_noahmp</scheme>
    </subcycle>
    <scheme>pbl_mynnedmf</scheme>
    <scheme>GWD</scheme>
    <scheme>Convection</scheme>
    <scheme>Photochem</scheme>
    <scheme>update_state</scheme>
  </group>

  <group name="microphysics">
    <subcycle loop="2">
      <scheme>mp_tempo</scheme>
    </subcycle>
    <scheme>update_state</scheme>
  </group>
</suite>
```

UFS_FV3_regional

```
<suite name="UFS_FV3_regional">
  <group name="time_vary">
    <scheme>time_vary_pre</scheme>
  </group>

  <group name="radiation">
    <scheme>rrtmg_lw</scheme>
    <scheme>rrtmg_sw</scheme>
  </group>

  <group name="physics_ps">
    <subcycle loop="2">
      <scheme>sfc_layer</scheme>
      <scheme>lsm_noahmp</scheme>
    </subcycle>
    <scheme>pbl_mynnedmf</scheme>
    <scheme>GWD</scheme>
    <scheme>update_state</scheme>
  </group>

  <group name="physics_ts">
    <scheme>Photochem</scheme>
    <scheme>update_state</scheme>
    <scheme>Convection</scheme>
    <scheme>update_state</scheme>
    <scheme>mp_tempo</scheme>
    <scheme>update_state</scheme>
  </group>
</suite>
```

Host driver: coupling code

```
IF (do_init) then
  call ccpp_register()
  call ccpp_init()
  do_init=.false.
END IF

call ccpp_physics_timestep_init()
call ccpp_physics_run(group_name="radiation")

IF (dycore_FV3) then
  call ccpp_physics_run(group_name="physics_ps")
  call ccpp_physics_run(group_name="physics_ts")
  call advection(dycore='FV3')
END IF

IF (dycore_MPAS) then
  call ccpp_physics_run(group_name="physics")
  call advection(dycore='MPAS')
  call ccpp_physics_run(group_name="microphysics")
END IF
```

Figure 3. Suite definition files for `UFS_MPAS_regional` (left) and `UFS_FV3_regional` (middle), and the host driver code that runs their groups through `ccpp_physics_run` (right)

Features for Interoperability

The framework can handle inconsistencies between the host and scheme metadata by applying ***automatic conversions*** when creating the caps. The conversions handle differences in units (e.g., microns -> meters), type (e.g., R8 -> R4), and vertical ordering convention (e.g., top_at_one -> bot_at_one).

Automatic physics ***memory management***. Data passed between schemes (ephemeral) is allocated by the Caps. There is no need to define and allocate fields on the host model side. These are known as ***CCPP Suite Variables.***

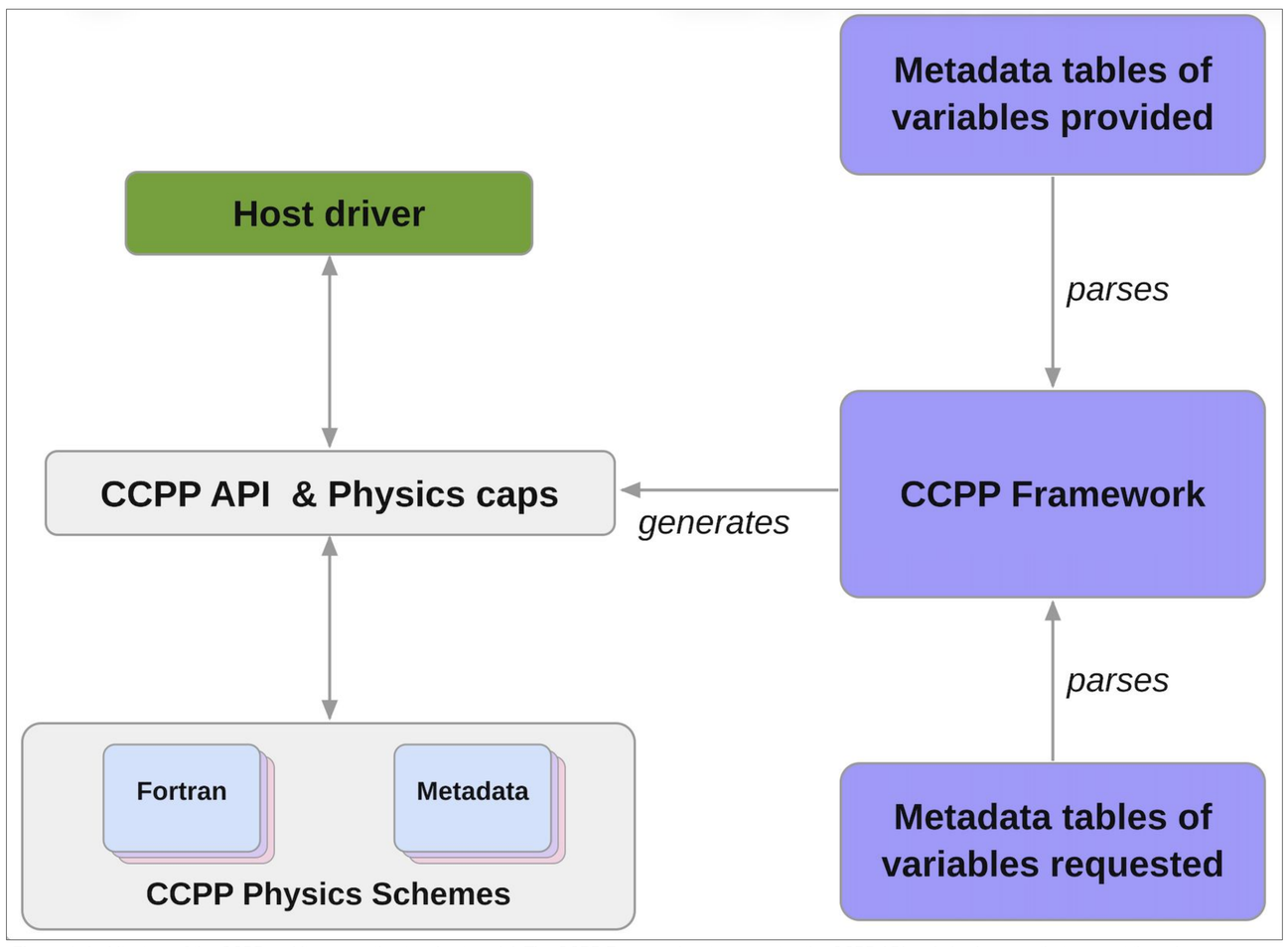


Figure 4. Architecture of the CCPP and its connection to a host model. The CCPP Framework auto-generates the CCPP API and physics caps that connect the schemes in CCPP Physics with the host model (Heinzeller et. al. 2023)

Atmospheric Constituents and Chemistry Coupling

The CCPP Framework automatically handles the ***memory management for constituents/tracers***. If a physics scheme requires a given constituent (or tracer), that constituent must be registered during that scheme's ***register phase.*** The constituent type is useful for inline physics-to-chemistry coupling, where the number of chemical constituents can be large and vary across forecast applications.

CCPP Single Column Model

Accompanying the CCPP Physics and Framework is the Single Column Model (CCPP SCM). The SCM is a lightweight host model connected to the CCPP Physics.

From a development perspective, the ***SCM provides a lower entry point*** to development compared to the fully coupled 3D Earth System Models. The SCM uses prescribed forcing (e.g., no dynamics), which makes it a great research tool for parameterization development and ***hierarchical development.***

Questions or Comments: **dustin.swales@noaa.gov**

Bernardet et al. (2024): https://doi.org/10.1175/BAMS-D-23-0227.1
Heinzeller et al. (2023): https://doi.org/10.5194/gmd-16-2235-2023

CCPP Technical documentation: https://ccpp-techdoc.readthedocs.io/en/latest/index.html
CCPP Single Column Model: https://github.com/NCAR/ccpp-scm
CCPP Physics code: https://github.com/NCAR/ccpp-physics
CCPP Framework code: https://github.com/NCAR/ccpp-framework