Introduction to Using the CCPP

GEOS-Chem@NCAR

Steve Goldhaber, Andrew Conley
NCAR

2020-01-08

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

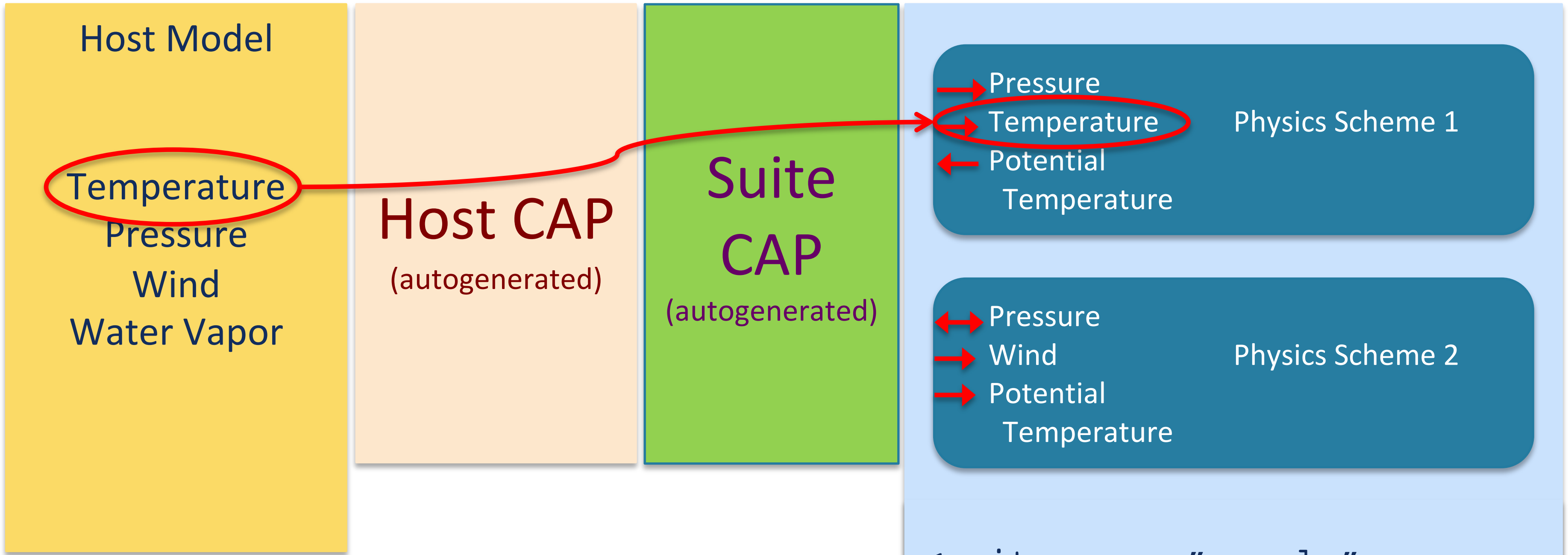
Outline

- What is the CCPP?
- What does the CCPP Framework do?
- How does a host model use the CCPP?
- How do I convert a chemistry scheme to be CCPP compliant?

What is the CCPP?

- The Common Community Physics Package (CCPP) is a system for supporting the development and use of portable physics schemes (parameterizations) and physics suites. It also refers to a library of these portable schemes and suites.
- **CCPP Framework**: This is the metadata parsing, analysis, and code generation software. It is jointly developed and managed by NOAA and NCAR under a formal agreement.
- **CCPP Host model**: This is any host model which can call CCPP physics suites. At NCAR, CAM and MPAS-A are being developed as CCPP host models.
- **CCPP physics scheme / suite**: A physics parameterization or suite of parameterizations that has been ported to the CCPP system.

What does the CCPP Framework do?

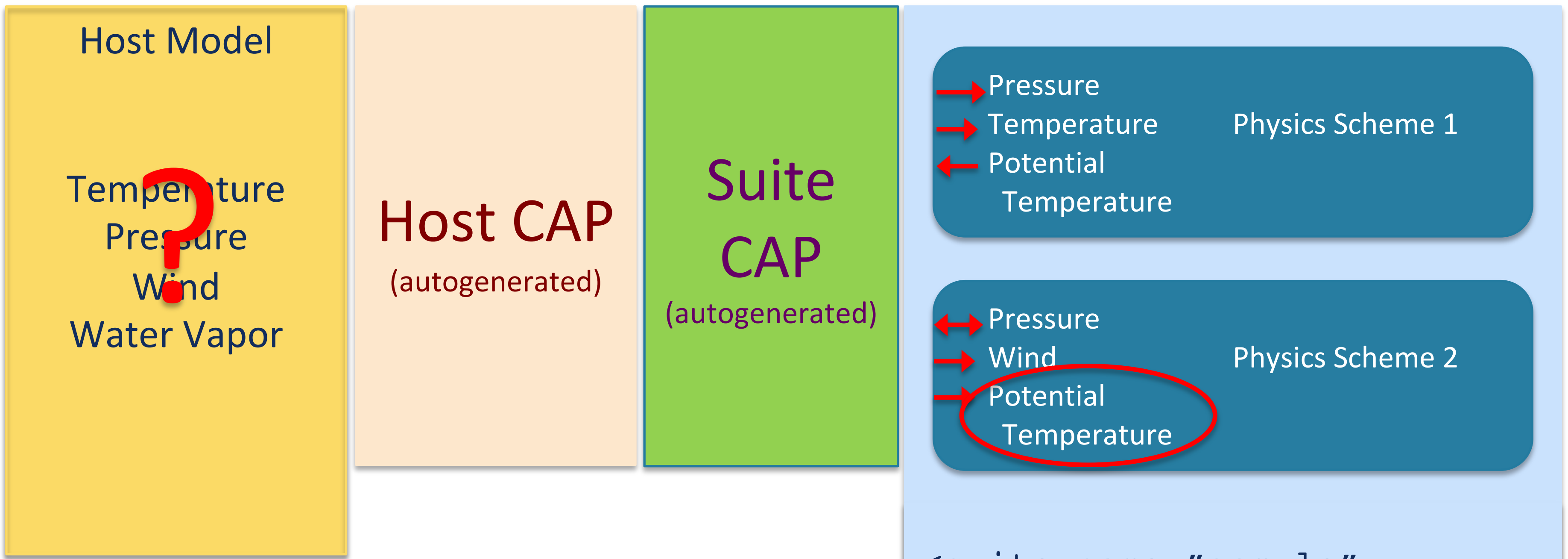


```
<suite name=”sample”
            version="1.0">
  <group name="physics">
    <scheme>scheme1</scheme>
    <scheme>scheme2</scheme>
  </group>
</suite>
```

Framework satisfies Temperature input of Physics Scheme 1 pass passing the host model Temperature

NCAR
UCAR
Steve Goldhaber: goldy@ucar.edu

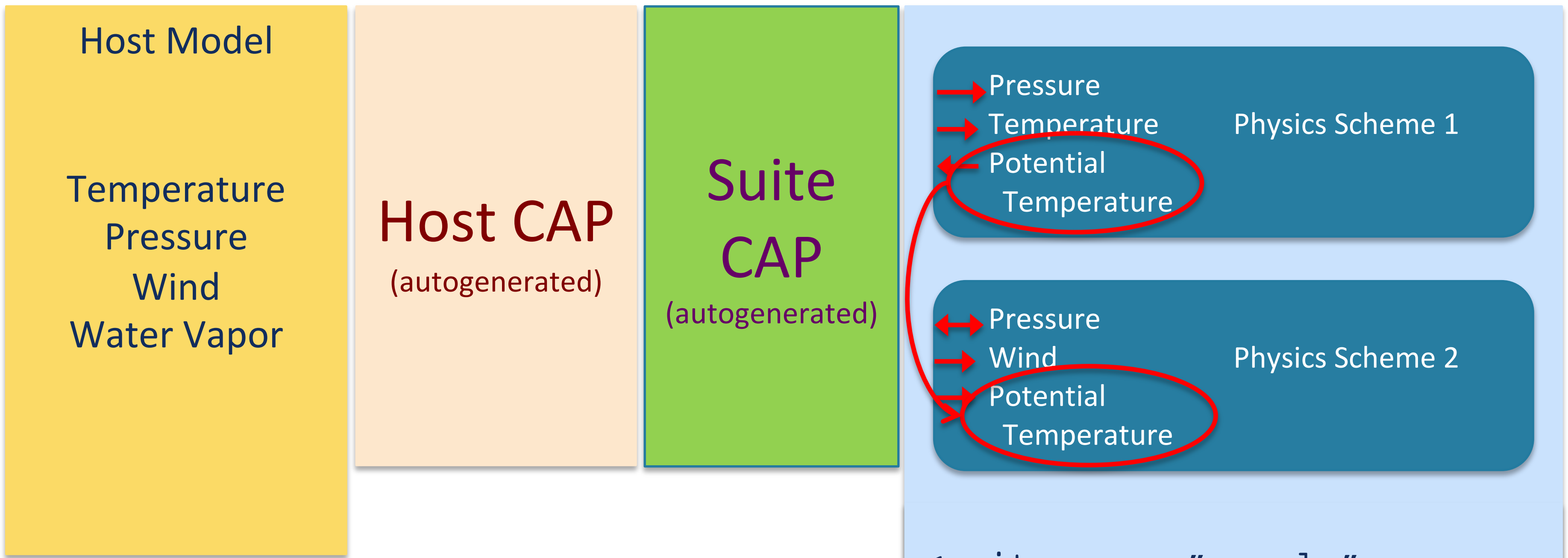
What does the CCPP Framework do?



```
<suite name="sample"
             version="1.0">
  <group name="physics">
    <scheme>scheme1</scheme>
    <scheme>scheme2</scheme>
  </group>
</suite>
```

The host model does not have a potential temperature field required by Physics Scheme 2.

What does the CCPP Framework do?



```
<suite name=”sample”
            version="1.0">
  <group name="physics">
    <scheme>scheme1</scheme>
    <scheme>scheme2</scheme>
  </group>
</suite>
```

Framework provides the potential temperature from the output of Physics Scheme 1

CCPP Metadata – The Magic Sauce

```
[ccpp-arg-table]
  name = host_data
  type = module
[ T ]
  standard_name = temperature
  units = K
  type = real | kind = kind_phys
  dimensions = (horizontal_dimension,
                vertical_layer_dimension)
```

```
[ccpp-arg-table]
  name = Physics_Scheme_1
  type = scheme
[ temp ]
  standard_name = temperature
  state_variable = true
  units = K
  type = real | kind = kind_phys
  dimensions =
             (horizontal_loop_extent,
            vertical_layer_dimension)
  intent = in
```

The Metadata table on the left describes part of the Host Model's data.
The Metadata table on the right describes part one of the arguments to Physics Scheme 1. Other types of table include host model module data and Derived Data Types (DDTs).

CCPP Metadata – The Magic Sauce

```
[ccpp-arg-table]
  name = host_data
  type = module
[ T ]
  standard_name = temperature
  units = K
  type = real | kind = kind_phys
  dimensions = (horizontal_dimension,
                vertical_layer_dimension)
```

```
[ccpp-arg-table]
  name = Physics_Scheme_1
  type = scheme
[ temp ]
  standard_name = temperature
  state_variable = true
  units = K
  type = real | kind = kind_phys
  dimensions =
             (horizontal_loop_extent,
             vertical_layer_dimension)
  intent = in
```

The Host Model calls this variable “T”.
Physics Scheme 1 calls this variable “temp”.
These names are independent.

CCPP Metadata – The Magic Sauce

```
[ccpp-arg-table]
  name = host_data
  type = module
[ T ]
  standard_name = temperature
  units = K
  type = real | kind = kind_phys
  dimensions = (horizontal_dimension,
               vertical_layer_dimension)
```

```
[ccpp-arg-table]
  name = Physics_Scheme_1
  type = scheme
[ temp ]
  standard_name = temperature
  state_variable = true
  units = K
  type = real | kind = kind_phys
  dimensions =
               (horizontal_loop_extent,
              vertical_layer_dimension)
  intent = in
```

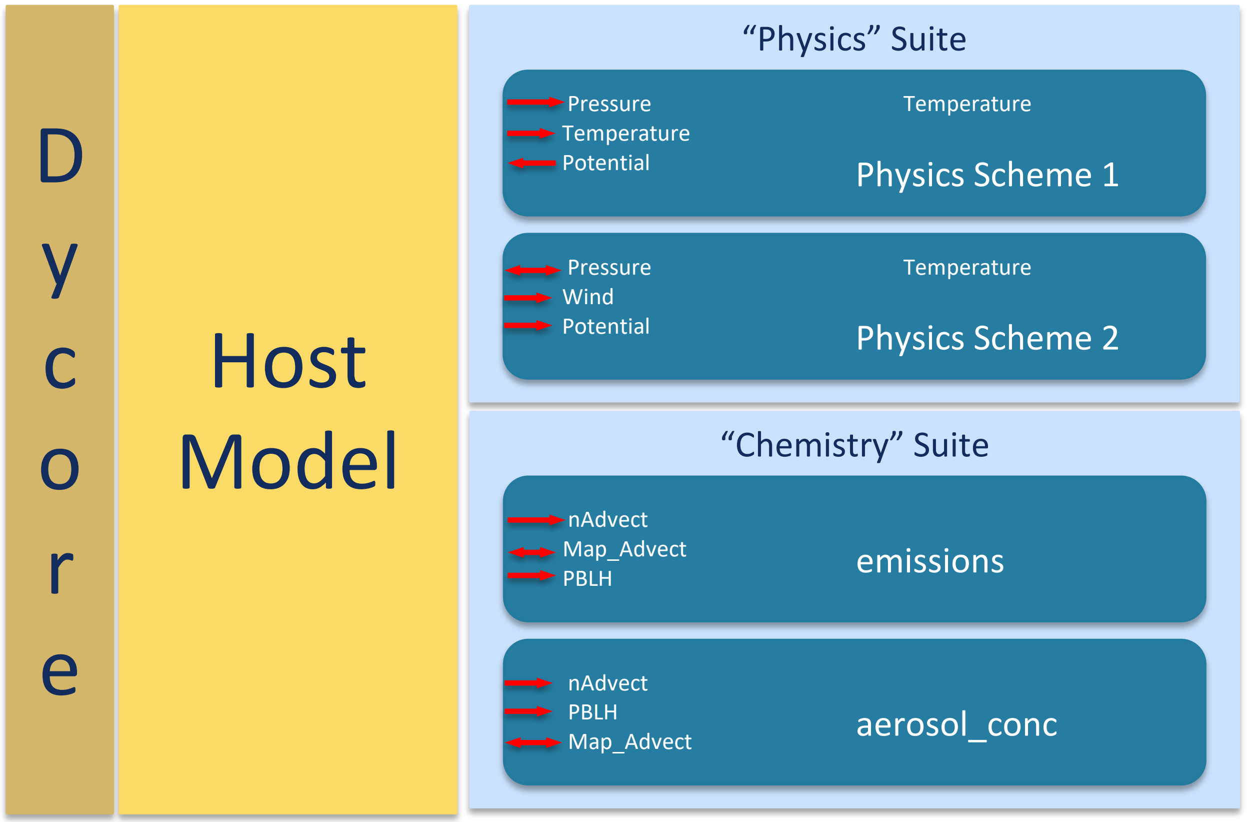
The standard name is the key indicator of the physical properties of a field.
The CCPP Framework knows these two fields represent the same physical quantity because they have the same standard name.

Outline

- What is the CCPP?
- What does the CCPP Framework do?
- How does a host model use the CCPP?
- How do I convert a chemistry scheme to be CCPP compliant?

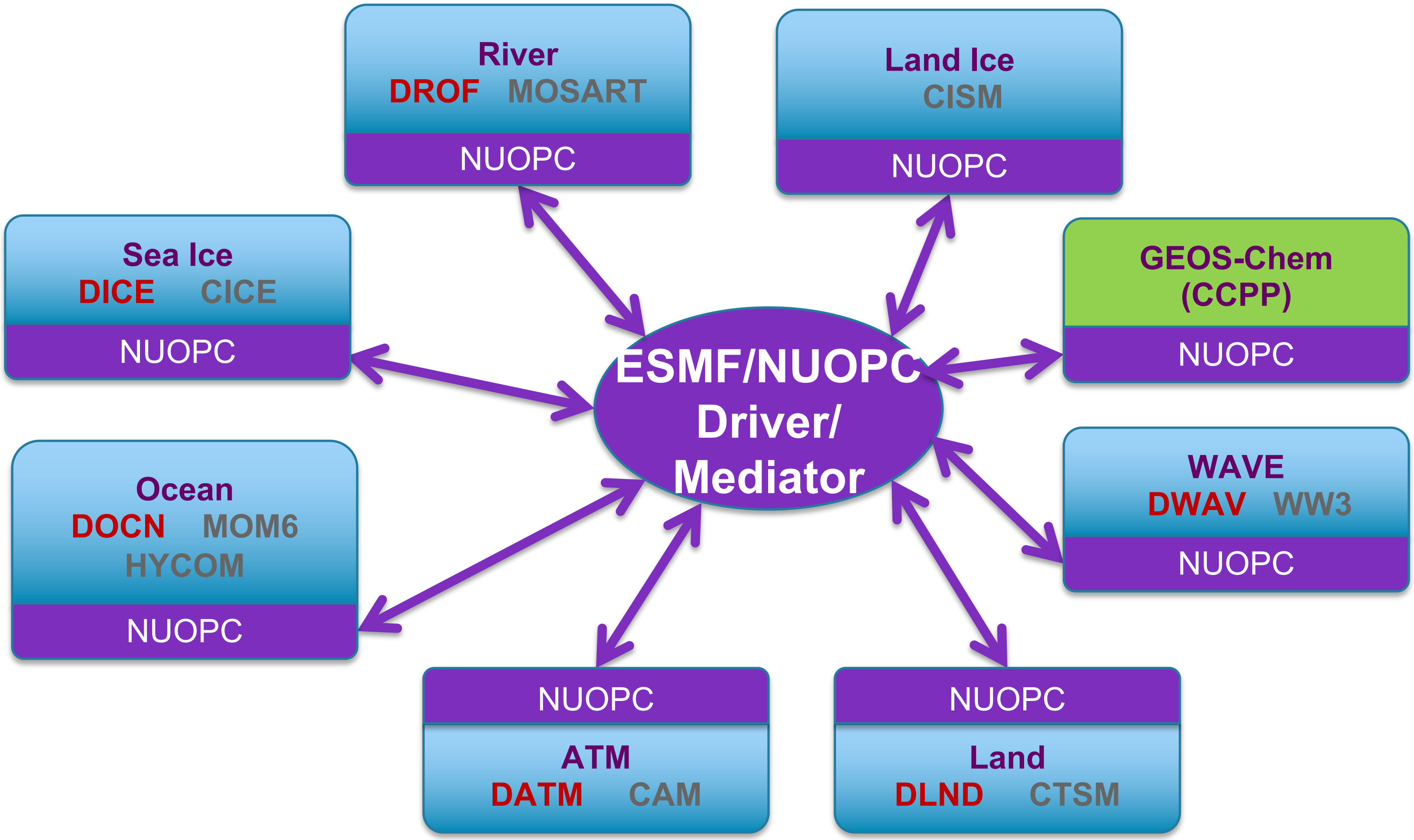
How does a Host Model use the CCPP?

Method 1: As a CCPP Suite called by a host model.



How does a Host Model use the CCPP?

Method 2: As a NUOPC component in a coupled system



Outline

- What is the CCPP?
- What does the CCPP Framework do?
- How does a host model use the CCPP?
- How do I convert a chemistry scheme to be CCPP compliant?

How do I convert a chemistry scheme to CCPP?

1. Modify the scheme’s arguments to be portable:
   - Modify the scheme’s inputs and outputs to expose basic physical quantities

     — or —
   - Write metadata for the scheme’s DDT arguments so that the CCPP Framework can find the portable quantities inside.
2. Add a header to the scheme
3. Run the metadata template generator
4. Fill in the template fields (standard_name, units, dimension standard names)

1: Modify the scheme's inputs and outputs to or document DDTs

- For GEOS-Chem, this would mean unwrapping the DDTs such as ChmState and MetState to expose the underlying fields (e.g., nAdvect, WetDepNitrogen).
- The alternative approach is to write a metadata file for each DDT that documents a standard name and other properties for each field that is used.

```
[ccpp-arg-table]
  name = host_data
  type = module
[ State_Chm ]
  standard_name = geos_chem_chem_state
  ddt_type = ChmState
  dimensions = ()
```

```
[ccpp-arg-table]
  name = ChmState
  type = DDT
[ WetDepNitrogen ]
  standard_name = wet_dep_rate_for_N
  units = kg s-1
  type = real | kind = kind_phys
  dimensions = (horizontal_dimension,
               vertical_layer_dimension)
```

2: Add a header to the scheme

```
!> \section arg_table_DO_CONVECTION Argument Table
!! \htmlinclude DO_CONVECTION.html
SUBROUTINE DO_CONVECTION(am_I_Root, Input_Opt,           &
         State_Chm, State_Diag, State_Grid, State_Met, RC)
  LOGICAL,          INTENT(IN)    :: am_I_Root
  TYPE(OptInput),   INTENT(IN)    :: Input_Opt
  TYPE(MetState),   INTENT(IN)    :: State_Met
  TYPE(GrdState),   INTENT(IN)    :: State_Grid
  TYPE(ChmState),   INTENT(INOUT) :: State_Chm
  TYPE(DgnState),   INTENT(INOUT) :: State_Diag
  INTEGER,          INTENT(OUT)   :: RC

…
```

3: Run the metadata template generator

```
<ccpp_framework>scripts/ccpp_fortran_to_metadata.py convection_mod.F
```

This will produce convection_mod.meta

4: Fill in the template fields (convection_mod.meta)

```
[ccpp-arg-table]
  name   = DO_CONVECTION
  type   = scheme
[ am_I_Root ]
  standard_name = enter_standard_name_1
  units = enter units
  type = logical
  dimensions = ()
  intent = in
[ Input_Opt ]
  standard_name = enter_standard_name_2
  units = enter units
  ddt_type = OptInput
  dimensions = ()
  intent = in
[ State_Chm ]
  standard_name = enter_standard_name_5
  units = enter units
  ddt_type = ChmState
```

How is the CCPP-framework governed?

- The CCPP Framework is jointly governed by NOAA and NCAR via an agreement signed by NOAA and NCAR (upper) management.
- The framework code is on a public GitHub repository (https://github.com/NCAR/ccpp-framework). The governance document is on that site's wiki.
- Short-term planning is done via GitHub issues. You can join conversations, open new issues, or even submit a code change or improvement for consideration (via a Pull Request).
- All the long-term planning documents are on the shared Google drive, Community Physics Framework (https://drive.google.com/drive/folders/0ANYmHUUzoxgkUk9PVA).

Questions?

Thanks!

https://github.com/NCAR/ccpp-framework