

Driver Requirements (From CPD document)

Classification used in tables below

D = dycore and model application development

O = operations

P = parameterization development

U = model user

“Type” updated to priority (1-3)

1 = First prototype

2 = First release (end of year?)

3 = Desirable (for future, or not precluded)

ID	Class	Priority	Item	Reason	Source
D1	D P	1	Driver allows parameterizations to be agnostic of dycore.	Scale awareness in parameterizations must support unstructured grids (pass grid cell area as an array, not single variable).	GMTB
D2	D	2	Provides easily configurable entry point for passing information to/from physics parameterizations.	Transfer of data through single driver entry point; facilitates grid interpolation and conversion, array reordering, variable conversion, destaggering.	GMTB
D3	P	1	Expandable to include new variables, of any dimensionality, in case parameterizations newly added to CCPP need information that cannot be obtained from existing Driver variables.		GMTB
D4	D P U	3	Ability to select different parameterizations of the same category via external option selection.	Provides flexibility and ease-of-use; allows direct comparison between schemes, possibly within an existing suite.	GMTB

D5	D P U	1	Parameterizations can be used as suites or be selected individually.	Suites are useful in both an operational and research environment; the ability to choose individual schemes is important for testing and development.	GMTB
D6	D P U	2	Order and frequency of calls to individual parameterizations is configurable.		GMTB
D7	D P	1	Capability to share same instance of physical constants with all model components (dycore, Driver, parameterizations).		GMTB
D8	D O P U	2	Availability of documentation including references, functional descriptions of code, guidance for how to call parameterizations as suites or individually in any order, and guidance on how to connect new parameterizations or dycores.		GMTB
D9	D O P	1-3	Driver is developed using modern and robust coding standards balancing portability, computational performance, usability, maintainability, and flexibility	For NCAR and community purposes: 1 usability and 1 computational performance should be prioritized over 2 flexibility (either 2 or 3). Portability here = across 'models' (not a high priority)	GMTB and various, including modified Kalnay rules
D10	P	1-2	Ability to drive parameterizations or suites in "offline mode".	Offline mode allows for sensitivity and process-based studies; CPD takes in 'data' and runs	GMTB

				parameterizations. I.e. Single column.	
D11	D O P	1	Ability to pass arbitrary “chunks” of dycore variables to parameterizations.	Increases computational performance. (Related to D9 performance)	GMTB, mod. Kalnay 6,7
D12	P U	1	Ability to provide variables computed by, or for, use within any parameterization for diagnostic purposes to the model I/O component.	Important for testing, development, and evaluation.	GMTB
D13	D P	1	Ability to provide variables computed by, or for, use within any parameterization to external models.	Facilitates consistency with other Earth System models (e.g. ability to share roughness length between parameterizations and LSM). Note that coupling to external models will be done at the dycore level, not by the CPD. CPD will provide variables from physics.	GMTB
D14	D P	2	The Driver will not modify answers produced by the parameterizations.	Provides a mechanism to prove inadvertent errors were not introduced by the driver. [Kind of assume this is the case?] [Transformations (e.g. height) will be explicit]	GMTB
D15	P U	2	Allows run-time specification of parameters (possibly greater than 1D).	Allows rapid tuning and sensitivity experimentation. (Specify namelist variables)	GMTB
D16	D P	2	More than one code variable with the same physical meaning but different names cannot exist.	Minimizes ambiguity.	GMTB

D17	D O P	3	Code management is designed so scientists can use and propose contributions to Driver.	Meets the NCEP goals for community modeling and enhances R2O.	GMTB
D18	D P U	1	Parameterizations can specify to the driver what fields it requires, which it generates (or modifies), and which fields it 'owns' (if any).	Some parameterizations provide data for other parameterizations. but require that other parameterizations do not modify the field.	NCAR
D19	D P	1	Parameterizations and physics driver must be able to communicate information to allow the host model to write state to restart, restart from previously written files, and to reset their internal state during a run from restart files.	The restart requirement is essential both to long runs and for data assimilation.	NCAR
D20			Does not exist		
D21		2	Host must communicate index ordering (which index is horizontal dimension, which index is vertical dimension, etc.) to parameterizations.	Parameterizations must know the ordering of input and output data so that it can do data rearrangement if necessary.	NCAR
D22	D P	2-3	The physics driver must be able to be multiply-instantiated (i.e., one run supports multiple, independent physics packages).	This allows a model to call, for example, a chemistry package on a different time scale and possibly different grid than the mainline physics package.	NCAR
D23	D P	2	The physics driver must be able to receive error codes from any parameterization and pass them to the host model for output and model termination.	If we are discouraging parameterizations from doing I/O and from stopping the model, we must pass this information to the host model	NCAR

D24 (a)	D P	2	The physics driver system must produce Fortran code as part of the preprocess step.	Scientists and performance engineers must be able to inspect the code that is part of the model run.	NCAR
D25	D P	2	The physics driver must be able to communicate arrays of variables (e.g., an array of tracers) whose extent is only known at compile time.	Many chemistry packages operate on arrays of species which may only be known at compile time. The metadata for each species must be passed through the driver from this host model to the parameterizations.	NCAR
D26	D	2	The physics suite definition must be able to define both process split and time split sequences as well as shadow (diagnostic) parameterizations.	Diagnostic parameterizations sample the model state but do not update the state or contribute to tendencies.	NCAR
D27	D P	2	The physics driver system must be able to handle fields with multiple time levels.		NCAR
D28	D P	2	The physics driver system must be configurable in an offline mode where a single parameterization or suite is driven from captured data.	Offline studies of a physics parameterization are essential during the development cycle. Merge with D10	NCAR
D29	D P	1	The physics driver system must be able to capture data during runtime for offline studies	This functionality supports D28.	NCAR
D30	D P	1	Pressure field handed to physics schemes must indicate whether they are wet or dry and, if wet, which water phases are included.	Physics parameterizations need to either make calculations based on the nature of the pressure field or to convert the pressure field (e.g., to dry) before doing calculations.	NCAR

				See physics dynamics coupling	
D31	D U	3	Run-time selection of physics suites and schemes	Regression testing, multi-physics ensembles Needs to also NOT compile everything.	NCAR
D32	D P	3	The physics driver system must operate on systems which do not support dynamic library loading.	Some leadership-class HPC systems do not support dynamic libraries.	NCAR
D33	D P	3	The physics driver system must support processing (e.g., averaging, max/min selection) of diagnostic fields which may be output multiple times during a run.	Parameterizations are often called multiple times during a suite time step. Since the host model is not active during this time, the driver (and / or parameterizations) must be able to handle multiple diagnostic outputs. Less important (can work around if hard)	NCAR
D34	D P	2-3	The driver must support passing of derived types between host model and physics schemes	When porting new physics that originally used DDTs in their argument list, it may be helpful to call the schemes exactly as they were called. Also, schemes may call libraries that store, e.g., state in a derived type.	NCAR
D35	D P	1	Driver supports multiple kind-types for integers and reals	Mixed-precision physics (may need more analysis as we proceed)	NCAR
D36	D P	2	Driver supports calling of generic physical parameterizations (including land-surface, simple ocean, combined schemes, purely diagnostic schemes)	Capability to call land-surface model as a physics parameterization is required for stand-alone WRF and MPAS. Parameterizations should not read files, but analytic forcing is fine.	NCAR
D37	D	2	Host model and CPD	At run-time (initialization),	NCAR

	P		may interrogate each other during a “handshaking” phase	the host model may need information about physics choices in order to properly allocate memory; the CPD may need information on which fields are available in the host model	
D38a	D P	2	The host model is responsible for allocating all data that flows between the host model and physics schemes	What is the source of this requirement? [See below: think a parameterization can define something]. Inputs and outputs touch the host model	NCAR
D38b	D P	2	The CPD must facilitate passing of data between physics schemes, which is never seen by the host model	[Semantic: if the CPD sees it, it is the host model?]	NCAR
D39	D P	3	The CPD should provide “hooks” to facilitate debugging and general scheme development	For example: can we call a developer-defined function before and after each call to a physics scheme to enable checking of field values? [Assume this would be a parameterization]	NCAR
D40	D P	3	Internally, the CPD should support the association of arbitrary (and extensible) metadata with fields	This would facilitate, e.g., range checks on fields before and after calls to schemes (through hooks), tracking of data flow in the CPD, and it may be used to implement matching of units, long_name, etc. of fields	NCAR
D41	P	3	Ability to deal with ‘stencils’ or subcolumns	May need some adjacent information available.	

Physics (CCPP) Requirements

CCPP = Common Community Physics Packages

Intent is NCAR will support it's own 'subset' of these for the community

ID	Class	Type	Item	Reason	Source
C1	P	1	Parameterizations in the CCPP are required to conform to the standard variables used in the Driver.	In order to avoid being intrusive to the parameterizations, if necessary suites will have an associated pre/post interface to convert from variables used in the driver to variables used in the parameterization.	GMTB
C2	D U	2	Multiple parameterizations of each category coexist in the CCPP.	A single package can support all NCEP needs (including research and development).	GMTB
C3	P U	N/A	Objective and transparent criteria are used to guide number and choice of parameterizations included in CCPP.	A Physics Review Committee reviews test results and ensures quality control of parameterizations and has authority over portfolio of supported parameterizations. Maintenance is kept to a manageable level while focusing on operational and research applications.	GMTB
C4	P U	3	Standard and documented testing procedures and metrics applied by all physics	The Physics Review Committee defines minimum testing procedures and metrics.	EMC

			developers.	This may include specific codes/tools to be employed in the test harness.	
C5	P U	2	Standard and documented observation and model databases for testing.	Both observation and model-generated datasets need to be selected and available for testing. This ensures that the Review Committee has material that is easy to judge. Tools to subset or process data may be part of this, as necessary.	EMC
C6	D P U	3	Parameterizations expose all parameters that are necessary for tuning to a particular model or application.	Tunable aspects of parameterizations will be configurable by run-time settings, e.g. Fortran namelists, allowing a single software instance of a parameterization to satisfy all foreseeable models and applications.	GMTB
C7	D P	1	Capability to share same instance of physical constants with all model components (dycore, Driver, parameterizations).		GMTB
C8	P	1	Code management is designed so community scientists can use and propose contributions to the CCpp.	Meets the NCEP goals for community modeling and enhances R2O.	GMTB
C9	D P U	3	Availability of documentation including references, functional descriptions of code, information on inputs/outputs to parameterizations, and guidance on how to add new parameterizations.		GMTB

C10	D P U	3	Employs modern and robust coding standards supporting portability, computational performance, usability, maintainability, and flexibility and follow coding guidelines listed in the Coding Standards .		GMTB and various, including modified Kalnay rules
C11		1	Parameterizations may read non-decomposed data (such as look-up tables), and other arrays that do not have scope outside of the physics scheme.		NCAR
C12		3	Parameterizations must pass a return code to the driver to indicate success or failure.	This allows status communication without the parameterization trying to write log messages.	NCAR
C13	P	2	The system that translates the parameterization metadata cap into a driver-callable interface must produce Fortran code as part of the preprocess step.	Scientists and performance engineers must be able to inspect the code that is part of the model run.	NCAR
C14	P	2	A physics parameterization must be able to conditionally compute a diagnostic field depending on whether or not the field will be output.	Some diagnostic calculations are expensive.	NCAR
C15	P	1	Physics schemes must not implement parallelism (OpenMP, MPI) internally	Threading in individual schemes would preclude any sane threading mechanism in the CPD. Task parallelism in a scheme may interfere with MPI at the model level.	NCAR

C16	P	1	Every physics scheme entry point (interface routine) is documented with specially formatted metadata.	The metadata is used to build “cap” routines to be called by the framework.	NCAR
C17	P	2-3	Physics schemes are 1-d / column-independent	We have no idea how to handle schemes that influence neighboring columns that may then need to be communicated	NCAR
C18	P	1	Schemes must follow standardized naming for entry points	Calls to schemes may be automatically constructed, and we’d need to be able to identify, e.g., the init, compute, and finalize entry points	NCAR
C19	P	1	Parameterizations can specify to the driver what fields it requires, which it generates (or modifies), and which fields it ‘owns’ (if any).	Some parameterizations provide data for other parameterizations. but require that other parameterizations do not modify the field.	NCAR
C20	P	1	Driver must have knowledge of its internal vertical level and indexing (e.g., index 1 equals model top). It must either use the model vertical level or convert to a ‘standard’ vertical level. In either case, this level information must be communicated to the parameterizations in case they use a different vertical scheme.	Models have different vertical schemes as do physics parameterizations. Parameterizations must know the vertical scheme of its interface fields so that they can convert if they use a different scheme.	NCAR
C21	P	1	A parameterization needs the ability to specify to the host model which species will be transported.	A chemistry package may have species which are not to be advected.	NCAR

C22	P	3	Parameterization metadata should allow a field specifying allowed field ranges.	As an example, a temperature field could generate an error if outside the range of 100K -- 345K. Could be used with D39.	NCAR
-----	---	---	---	--	------