REQUIREMENTS

An initial proposal on requirements for interoperability of physical parameterizations and associated driver was drafted by GMTB and sent to EMC and the NGGPS Program Office in November 2015. This proposal incorporated some materials produced previously by the Earth System Prediction Capability Physics Interoperability ESPC PI) team. Requirements evolved over the following year with input provided by EMC, the NGGPS Program Office, the NGGPS Physics Team, the EMC Strategic Implementation Plan (SIP) Physics Working Group, and the ESPC PI team, with the most substantial revision being done in February 2016 with input from EMC. These requirements are consistent with the *Project 3: Collaborative framework for developing physical parameterizations* listed in the document *EMC Strategic Implementation Plan (SIP) for Evolution of NGGPS to a National Unified Modeling System*.

These requirements were formally accepted by the NGGPS Program Office in October 2017.

Classification used in tables below
D = dycore and host application development
O = operations
P = parameterization development
U = model user

5.1 IPDe requirements

ID	Class	Type	Item	Reason
D1	D P	Requirement	The driver shall allow parameterizations to be agnostic of host application.	Well-established convention facilitates data mapping.
D2	D	Requirement	The driver shall provide an easily configurable entry point for passing information to/from physics parameterizations.	Enhances portability and simplifies the interface for community contributions.
D3	P	Requirement	The driver shall be expandable to include new variables.	Newly added parameterizations may need information not already provided by host application.

D4	D P U	Require ment	The driver shall provide the ability to select different parameterizations of the same category via an external option selection.	Provides flexibility and ease-of-use; allows direct comparison between schemes, possibly within an existing suite.
D5	D P U	Require ment	The driver shall allow parameterizations to 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.
D6	D P U	Require ment	The driver shall allow the order and frequency of calls to individual parameterizations to be configurable.	Allows for sensitivity testing of different physics configurations.
D7	D P	Require ment	The driver shall provide the capability to share the same instance of physical constants with all model components (host application and parameterizations).	Maintains consistency among model components.
D8	D O P U	Require ment	The driver shall include 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 host applications.	Community code should be well-documented for users and developers.
D9	D O P	Require ment	The driver shall be developed using modern and robust coding standards balancing portability, computational performance, usability, maintainability, and	Following Kalnay rules (Kalnay et al., 1989, BAMS and Doyle, J. D, M. Iredell, P. Tripp, J. Dudhia, T. Henderson, J. Michalakes, J. A. Ridout, J. Rosinski, S. Rugg, R. Adams Selin, T. R. Whitcomb, K. Lutz, and D. McCarren, 2015. Revisiting Kalnay rules for physics interoperability: 25 years later. AMS Eugenia Kalnay Simposium, Jan 5-8,

			flexibility, and follow coding guidelines listed here.	Phoenix, AZ)
D10	P	Requirement	The driver shall provide the ability to drive parameterizations or suites in “offline mode”.	Offline mode allows for sensitivity and process-based studies; removes response from other components to focus on impact from parameterizations.
D11	D O P	Requirement	The driver shall provide the ability to pass arbitrary “chunks” of input variables to parameterizations.	Follows modified Kalnay rules 6, 7. Increases computational performance.
D12	P U	Requirement	The driver shall provide the ability to deliver variables computed by, or for, use within any parameterization for diagnostic purposes to the model I/O component .	Important for testing, development, and evaluation.
D13	D P	Requirement	The driver shall provide the ability to deliver 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 host application level.
D14	D P	Requirement	The driver shall not modify answers produced by the parameterizations.	Eliminates inadvertent errors.
D15	P U	Requirement	The driver shall allow run-time specification of parameters (possibly greater than 1D).	Allows rapid tuning and sensitivity experimentation.
D16	D P	Requirement	The driver shall ensure that more than one code variable with the same physical meaning but different names cannot exist.	Minimizes ambiguity.

D17	D O P	Require ment	The driver code management and support shall be designed so community scientists can use and propose contributions.	Meets the NCEP goals for community modeling and enhances R2O.

5.2 CCPP physical parameterization requirements

The primary purpose for each of the physical parameterizations within a model is to advance the solution of specified tendencies or state variable. The CCPP relies on a separation of concerns between physics and dynamics, and it is recommended that parameterizations do not deviate from this primary purpose.

Note is that the operational FY17 GFS suite has been grandfathered and is not required to meet these requirements.

ID	**Class**	**Type**	**Item**	**Reason**
C1	P	Require ment	Physics schemes shall conform to standard variables.	Common variables facilitate scheme portability.
C2	D U	Require ment	The CCPP shall allow multiple parameterizations of each category to coexist in the CCPP.	The CCPP can support all NCEP needs (including research and development).
C3	P U	Require ment	Transparent criteria shall be used to guide number and choice of parameterizations included in CCPP.	A Change Review Board 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.
C4	P U	Require ment	The CCPP schemes shall have standard and documented testing procedures and metrics applied by all physics developers.	The Change Review Board defines minimum testing procedures and metrics. This may include specific codes/tools to be employed in the test harness.

C5	P U	Require ment	The CCPP schemes shall have 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 Change Review Board has material that is easy to judge. Tools to subset or process data may be part of this, as necessary.
C6	D P U	Require ment	The CCPP schemes shall permit parameterizations to expose all tunable parameters	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.
C7	D P	Require ment	The CCPP schemes shall permit a capability to share same instance of physical constants with all model components (host application and parameterizations).	Maintains consistency among model components.
C8	P	Require ment	The CCPP schemes code management shall be designed so community scientists can use and propose contributions.	Meets the NCEP goals for community modeling and enhances R2O.
C9	D P U	Require ment	The CCPP schemes shall have documentation including references, functional descriptions of code, information on inputs/outputs to parameterizations, and guidance on how to add new parameterizations.	Community code should be well-documented for users and developers.
C10	D P U	Require ment	The CCPP schemes shall employ modern and robust coding standards supporting portability, computational performance, usability, maintainability, and flexibility and follow	Follows modified Kalnay rules.

			coding guidelines listed in the Coding Standards.	

5.3 Host application cap requirements

Note that the host application cap (FV3GFS cap) used in the first release of CCPP already meets the requirement below

ID	Class	Type	Item	Reason
H1	D	Requirement	The host application cap shall be written in Fortran.	The IPDe assumes various C to Fortran constructs, and assumes Fortran module constructs.
H2	D	Requirement	The host application cap (or the upstream calling application) shall manage all variables that are arguments to individual parameterizations. Manage includes, but is not limited to, allocation, distributed communication, initialization, I/O, correct numbers of scalars, and metadata.	For consistency,it is the task of the upstream systems to manage the variables in the subroutine argument lists.
H3	D	Requirement	The host application cap (or the upstream calling application) shall perform all required I/O, except for a few notable cases (individual parameterizations may use look-up table initializations and table entries).	The fundamental purpose of a physical parameterizations is to compute some specific physical process. The more the parameterizations stay aligned with this standard, the more portable the schemes are. Allowing complicated I/O systems to be introduced into parameterizations reduces the chance at simple portability of those schemes with a new Host Application.
H4	D	Requirement	The host application cap (or the upstream calling application) shall handle all required distributed memory processing for decomposed arrays, if	The physical parameterizations do not carry information that allows them to determine neighboring grid columns. The parameterizations are all assumed to be 1d columns of independent data, though for performance purposes, blocks of

			any is required. Upon entry into each parameterization, each input field in the argument list is assumed to be the correct value to use. One exception is that the parameterizations may broadcast look-up table information from the master task to the rest of the communicator. This MPI communicator must be an input argument from the host model to the physics parameterization.	those 1d columns may be bundled into arrays.
H5	D	Require ment	The host application cap (or the upstream calling application) shall handle all processing that requires that the parameterizations be computed on a grid or resolution different than the upstream calling application.	For ease of portability, the parameterizations only know the arrays provided in the argument lists, the provided array sizes, and the computational extent for each of the arrays. The parameterizations are not aware if the incoming arrays are indeed the original size of the grid that the Host Application is running.
H6	D	Require ment	The host application cap (or the upstream calling application) shall handle all processing that requires that the parameterizations be run concurrently.	The physical parameterizations have no information about the sequential nature of their own processing, other than the list of arguments defined as either input or output. Because all information for a parameterization comes through the argument list, the parameterization is well suited to being insulated from external processing techniques. The upstream calling application has the necessary software infrastructure tools to set up concurrent parameterization processing.
H7	D	Require ment	The host application cap (or the upstream calling application) can have OpenMP parallelism and/or allow physics schemes to use OpenMP internally. The	The cap for the physical parameterizations is automatically manufactured. For timing performance and portability, all OpenMP threading for a particular scheme is controlled by each scheme's cap.

			number of OpenMP thread a scheme is allowed to use internally must be provided as an input argument to the scheme.	
H8	D	Require ment	The host application cap shall use Fortran array syntax that is valid for arguments that have an explicit interface.	Taking advantage of argument mismatch for type, kind, and rank is only available with explicit interfaces. Given that the purpose of the effort is to include additional schemes, allowing the compilers to find argument mismatches is a benefit.