

Extensions to the Basic Model Interface to Support Serialization of a Model's State Variables for Load Balancing and Checkpointing

**Scott D. Peckham^{1,4}, Jessica L. Garrett^{1,3}, Jonathan M. Frame^{1,5}, Keith S. Jennings^{1,3},
Luciana L. Kindl da Cunha^{1,6}, Shengting Cui^{1,2}, Robert L. Bartel¹, Nels J. Frazier^{1,2},
Donald W. Johnson¹, Fred L. Ogden¹, Trey C. Flowers¹**

*¹NOAA National Water Center, ²ERT Inc., ³Lynker Technologies, ⁴University of Colorado,
⁵University of Alabama, ⁶West Consultants, Inc.*



NOAA's NextGen National Water Model & BMI

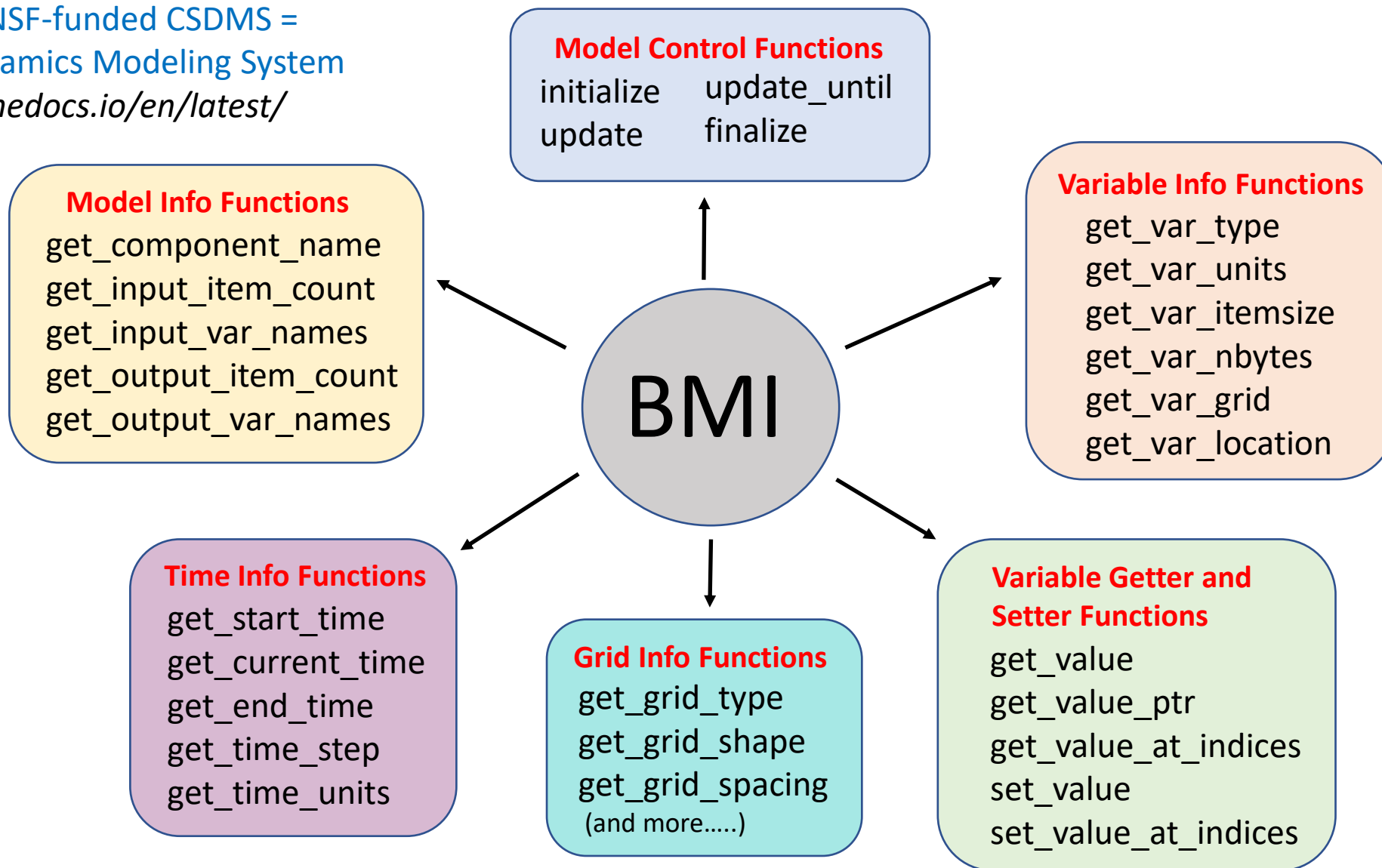
In coordination with federal water prediction partners, NOAA's Office of Water Prediction (OWP) leads development of a **model coupling framework** called the Next Generation Water Resources Modeling Framework (Nextgen). As the next version of the National Water Model, this framework uses a non-invasive, **community-standard API** for computational models called the **Basic Model Interface (BMI)**. Nextgen uses an **Adapter-Mediator pattern** to access model functions using calls to BMI functions.

BMI function calls provide fine-grained control (initialize, update, finalize), variable getters and setters, and functions to retrieve information about a model's input and output variables (e.g. grid, type, and units). This information allows the framework to automatically call **mediators**, when needed, to facilitate passing values of variables between models (e.g. for **regridding**, **time interpolation** and **unit conversion**). BMI supports models written in **C**, **C++**, **Fortran** and **Python**. The Nextgen project has implemented BMI for several hydrologic models and has run proof-of-concept tests.

BMI provides something like a **universal remote control** for interacting with a set of **heterogeneous** models, written by different authors in different languages, etc.

The Basic Model Interface (BMI) v2.0

BMI was developed by NSF-funded CSDMS =
Community Surface Dynamics Modeling System
See: <https://bmi.readthedocs.io/en/latest/>



Extending BMI with Variable Roles

In BMI version 2.0, the focus was on **coupling** models, i.e. passing the values of variables between models and automatically calling mediators for regridding, unit conversion, and temporal interpolation. BMI **variable information functions** (i.e. `get_var_*`) therefore only needed to provide metadata for a model's **input and output variables**.

We've extended BMI to provide metadata for **any** model variable, including its **role**. This supports new capabilities such as model **serialization** and **calibration**.

New Variable Role Names

- **all**
- `array_length`
- `constant`
- `diagnostic`
- `directory`
- `filename`
- `file_offset`
- `info_string`
- **`input_from_bmi`**
- `input_from_file`
- `not_set`
- `option`
- **`output_to_bmi`**
- `output_to_file`
- **`parameter_adjustable`**
- `parameter_fixed`
- `state`
- `time_info`

Extending BMI with New Functions

BMI Function	Description
get_bmi_version	Get the version of the Basic Model Interface.
get_var_count	Get the number of variables that have the given role .
get_var_names	Get the names of all variables that have the given role .

Variable Information Functions

BMI Function	Description
get_var_type	Get the data type of a variable.
get_var_units	Get the units of a variable.
get_var_itemsize	Get size of one element of a variable (in bytes).
get_var_nbytes	Get the total size of a variable (in bytes).
get_var_grid	Get the integer grid identifier for a variable.
get_var_location	Get the grid element type of a variable (e.g. “node”, “face”)
get_var_role	Get the role of a variable.
get_var_length	Get total number of elements of a variable.
get_var_index	Get the index of a variable (in struct array).

A new role-based **get_var_count()** generalizes the BMI v. 2.0 functions: **get_input_item_count()** and **get_output_item_count()**.

Similarly, a new **get_var_names()** generalizes the BMI v. 2.0 functions: **get_input_var_names()** and **get_output_var_names()**.

Variable Information Functions now return info for **any** model variable, not just input and output variables.

Three new functions (left, in **red**) have been added to get the **role**, array **length** and (var_info) **index**.

Framework Utility for Serialization & Deserialization

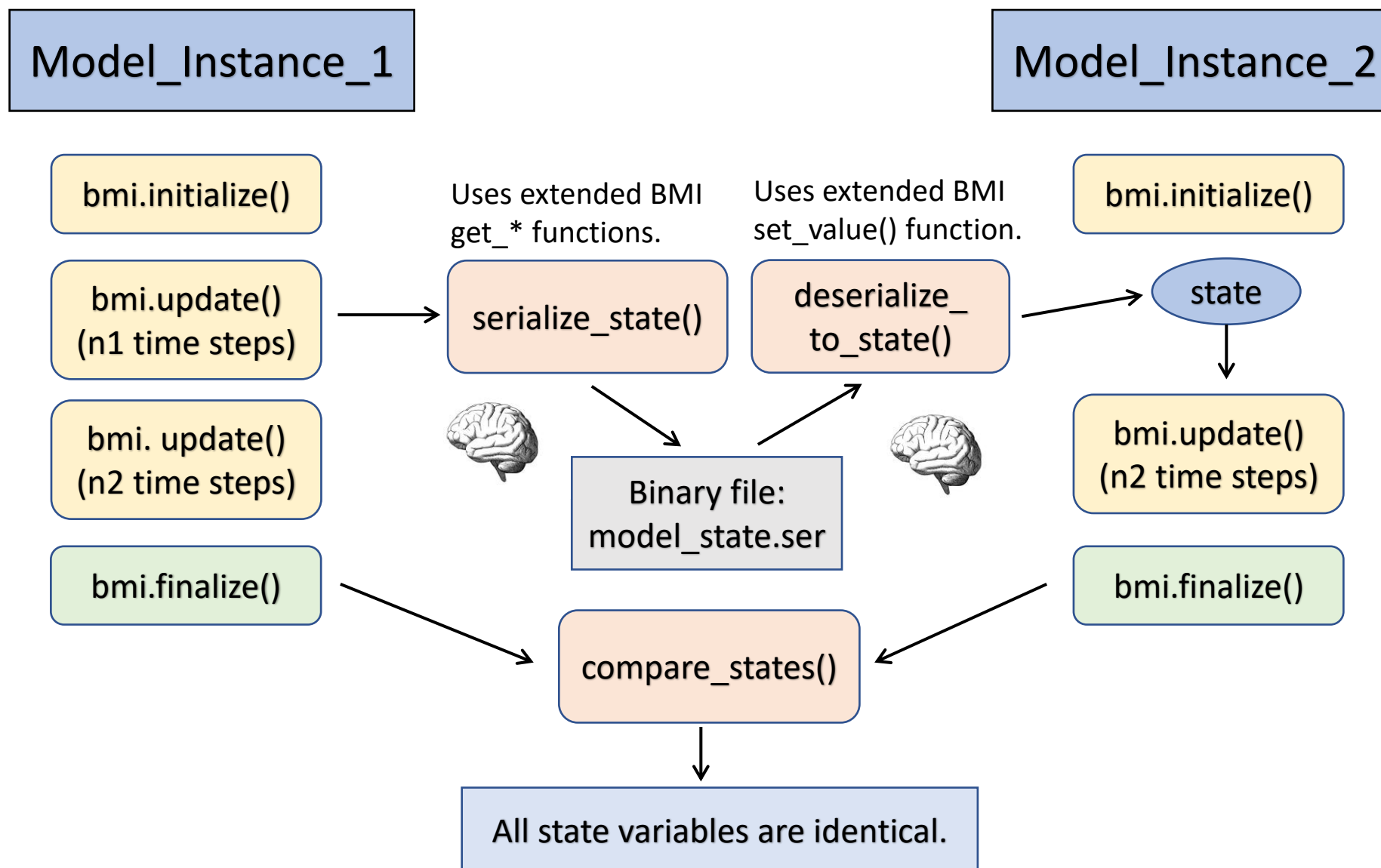
BMI follows a **separation of concerns** philosophy that provides model developers with maximum added value for a minimum amount of extra work. The modeling framework (e.g. CSDMS or NextGen) uses information provided by BMI functions in order to **automatically** perform tasks such as **coupling** (i.e. mediation required to exchange values), **calibration** & **serialization**. The framework calls general-purpose **framework utilities** to perform these tasks without user intervention.

We have written a general framework utility for **efficient, binary serialization and deserialization** of model state variables that provides the functions shown here.

It gets required information from BMI functions and uses the **msgpack-c** library.

Utility Function	Description
get_file_size	Get size of file with binary, serialized model state.
get_state_var_count	Get the total number of model "state" variables.
get_state_var_names	Get the names of all model "state" variables.
get_state_var_types	Get the types of all model "state" variables.
get_state_var_lengths	Get the array lengths of all model "state" variables.
get_state_var_ptrs	Get pointers to all model "state" variables.
serialize	Serialize all model "state" variables to a binary file.
deserialize_to_state	Deserialize saved model state and write to new instance.
compare_states	Compare all "state" variables for 2 instances of a model.

Serialization and Transference of Model State



Simplified BMI Implementation for C Models

```
typedef struct Variable{
    unsigned int index;
    char name[BMI_MAX_VAR_NAME];
    char type[BMI_MAX_TYPE_NAME];
    unsigned int length; // (nbytes / itemsize)
    char role[BMI_MAX_ROLE_NAME];
    char units[BMI_MAX_UNITS_NAME];
    char location[BMI_MAX_LOCATION_NAME]; // e.g. "node", "face"
    int grid; // e.g. 0
} Variable;
```

Define a type “Variable” as a **struct** with fields for all of the variable attributes in the C header file.

Then define “var_info” as an array of type “Variable” as shown below.

Advantages

Easier to implement the BMI functions.

Helps to prevent “misalignment errors”.

Most BMI get_var_* functions can then be written to get variable info from here and can be reused with little or no change for all C models

```
Variable var_info[] = {
{ 0, "input_dir", "char", 80, "directory", "none", "none", -1 },
{ 1, "input_file", "char", 80, "filename", "none", "none", -1 },
{ 2, "pi", "double", 1, "constant", "none", "node", -1 },
{ 3, "x", "double", 10, "input_from_bmi", "m", "node", 0 },
{ 4, "y", "double", 10, "output_to_bmi", "m", "node", 0 },
{ 5, "m", "double", 1, "parameter_adjustable", "none", "none", -1 },
{ 6, "b", "double", 1, "parameter_adjustable", "m", "none", -1 },
{ 7, "verbose", "int", 1, "option", "none", "node", -1 },
{ 8, "n_steps", "int", 1, "time_info", "none", "none", -1 },
{ 9, "time_step", "int", 1, "time_info", "h", "none", -1 },
{ 10, "model_name", "char", 80, "info_string", "none", "none", -1 },
{ 11, "error_code", "int", 1, "diagnostic", "none", "none", -1 } }
```


Summary

To support the requirements of NOAA's NextGen Modeling Framework, we have extended the Basic Model Interface (BMI) v. 2.0 by:

- (1) Introducing a variable's **role** as a new attribute (with 17 role names),
- (2) Implementing **reusable** Variable Information Functions for **all** variables.
- (3) Adding a few new BMI functions.
- (4) Providing a simplified "var_info" mechanism for BMI implementation.

These enhancements extend the capabilities of BMI beyond **model coupling** to support **calibration**, **serialization** and other future, role-based capabilities.

Serialization & deserialization is provided to models with a **new framework utility** that supports HPC **load balancing** and **checkpointing**.

You can email me at: scott.peckham@noaa.gov for more information.



OWP | OFFICE OF
WATER
PREDICTION



2021 AGU Fall Meeting

Thank You!



Scott D. Peckham, NOAA Affiliate
University of Colorado, Boulder



scott.peckham@noaa.gov



<https://water.noaa.gov>