INL/EXT-16-41982

# NEAMS-IPL MOOSE Midyear Framework Activities

Cody Permann
Brian Alger
John Peterson
Andrew Slaughter
David Andrš
Richard Martineau

May 2017


The INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance

INL/EXT-16-41982

# NEAMS-IPL MOOSE
# Midyear Framework Activities

**Cody Permann**
**Brian Alger**
**John Peterson**
**Andrew Slaughter**
**David Andrš**
**Richard Martineau**

**May 2017**

**Idaho National Laboratory**
**Modeling and Simulation Department**
**Idaho Falls, Idaho 83415**

**http://www.inl.gov**

**Prepared for the**
**U.S. Department of Energy**
**Office of Nuclear Energy**
**Under DOE Idaho Operations Office**
**Contract DE-AC07-05ID14517**

# ABSTRACT

The MOOSE Framework is a modular pluggable framework for building complex simulations. The ability to add new objects with custom syntax is a core capability that makes MOOSE a powerful platform for coupling multiple applications together within a single environment. The creation of a new, more standardized JSON syntax output improves the external interfaces for generating graphical components or for validating input file syntax automatically without the need for explicit developer interaction. The design of this interface and the requirements it satisfies are covered in this short report.

# ACRONYMS

| | |
|---|---|
| GUI | Graphical User Interface |
| JSON | JavaScript Object Notation |
| MOOSE | Multiphysics Object Oriented Simulation Environment |
| NEAMS | Nuclear Engineering Advanced Modeling and Simulation |
| YAML | Yet Another Markup Langague |

# 1 Workbench Integration

The Multiphysics Object Oriented Simulation Environment (MOOSE) is designed to be a modular pluggable framework for building complex simulations. The core of this design relies on a commonly used object-oriented paradigm called the factory pattern. MOOSE defines a set of basic interfaces which may be extended by developers to provide new capabilities and enable new simulations. Each user-developed object inherits from a common base class and must supply a corresponding template specialization of a free function which describes the names, types and several other attributes of every parameter which is used by the object at construction time. These syntax registration functions (validParams) collectively make up the available syntax for an application and can be used for dynamically generating GUI components or validating input. The data from these methods form the basis for integration within the NEAMS Workbench suite.

Each objects validParams function allows for fine-grained control over the attributes of every parameter an object expects to receive. These attributes include whether or not parameters are required, have default values, or have restricted valid ranges for numeric parameter types, among several other properties. These functions provide a rigorous interface which is used by MOOSE to validate the parameters passed through input files or the programmatic creation of objects within a simulation prior to the creation of each registered MOOSE object. This enables the framework to perform the tedious parameter validation checks in a single location without having the need for involving the developers of physics modules to replicate these checks in each object that is built individually.

As part of the 2017 NEAMS work package, a new JavaScript Object Notation (JSON) syntax tree formatter was created to simplify the integration of MOOSE-based applications into the NEAMS Workbench. JSON is a standardized data-interchange format, which is easy for both machines and humans to read and write. Modules for handling the input and output of JSON formatting files are widely available in many different programming languages. Figure 1 shows a small portion of the MOOSE syntax tree as built by the Parser and Action system. Tree formatters are given a reference to this tree to produce data in various formats. Prior to performing the work under this scope, MOOSE supported two syntax output formats: GetPot, a block structured format which mirrors the input file syntax, and YAML a machine-friendly data serialization language. The GetPot format is not well-suited for machine parsing and the YAML format is not as commonly used and is more difficult for humans to read. Hence, the JSON format was developed to better support external applications and to simplify internal maintenance. Additionally, the amount of meta-data output in the JSON format was increased substantially to remove the number of the assumptions made in recreating valid input syntax or validation rules.

As part of the JSON format development, several issues identified by the Workbench teams independent input file parser were identified and addressed. Several improvements to the meta-data in all of the formats were enhanced to remove ambiguities, unspecified behaviors and assumptions. Several input files in multiple applications were updated as a result of the Workbench teams findings. Additionally, some application behaviors have been modified to conform more to the valid syntax they produce (e.g. not allowing private or hidden parameters to be set from the input file).

Currently, the Workbench team uses a parser which has been published as a separate module but is not available publicly. The MOOSE maintainers have expressed interest in utilizing this module but its not practical to use a module which cannot be obtained publicly as open-source software. While we do have the ability to download and test against this parser to prevent regressions that may cause validation issues but otherwise allow simulations to run properly, the onus would fall onto the MOOSE maintainers to fix errors uncovered by the module with limited availability. The MOOSE maintainers propose having the Workbench developers consider open-sourcing the

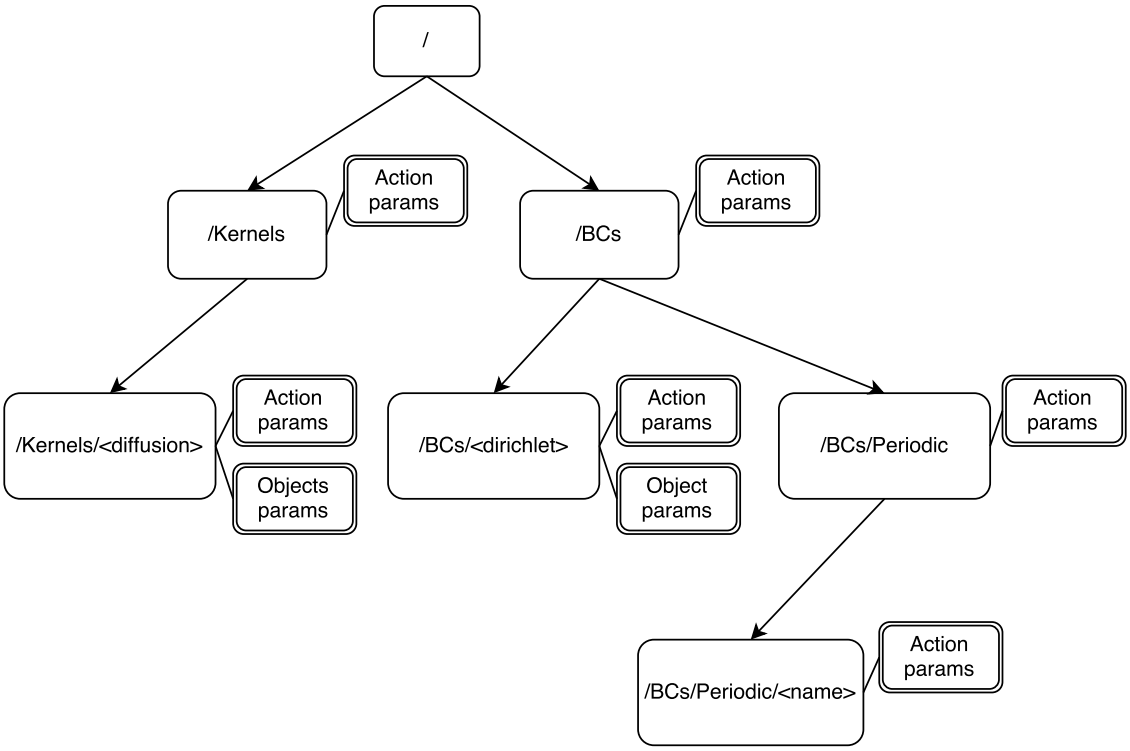


Figure 1: MOOSE syntax tree representing the valid syntax paths with InputParameter objects suitable for creating arbitrary meta-data dumps.

workbench software to enable a common parsing platform for modern tool development.

# 2 Conclusion

The new JSON syntax output greatly simplifies the automatic generation of GUI components and input file validation. In addition, undocumented assumptions and special case syntax handling has been removed from all GUI side development in an effort to assure that the information required for the validation of inputs is in the syntax information dump. The Workbench integration and MOOSE specific GUI generation is currently being handled with a single meta-data output format. This simplifies maintainability while creating a common platform capable of serving the needs of multiple disparate projects. The MOOSE team plans to continue supporting and expanding the capabilities of the Workbench validator as the framework development is advanced.