HELICS Domain API v0.1

Trevor Hardy *PNNL*

I. Introduction

The HELICS Domain API task is intended to define and implement a standardized interface for federates covering similar domains. The standardized interface allows simulation tools to be interchanged without re-writing the HELICS interface definitions and provides a guide when defining HELICS interfaces outside the domain API. The consequence is less arbitrary variability in interface definitions for a more consistent modeling experience.

A consequence of the domain API definition is the creation of a "federate" abstraction layer. This layer is the interface between a particular simulation tool (and its corresponding model) and the rest of the federation. For some simulation tools, this layer already exists as their current integration with HELICS involved the creation of wrapper code where API calls were made to the simulation tool to extract information for publication to the federation and commands to adjust the operation of the simulator or state of the model prior to executing the simulation tool to resolve the updated model. That is, the wrapper code treats the simulation tool as a solver and it pulls and pushes information to and from the solver to connect it to the rest of the federation.

This wrapper can effectively become a federate layer providing the capability of being able to interact appropriately with a given simulation tool in response to a standardized interface with the rest of the federation. This document is primarily concerned with the latter portion of this functionality by defining the HELICS interface the federate layer must implement.

This is currently a very early draft of the API definition and discussion on its contents is welcome. Even in this state there are several unanswered questions in my mind:

1) Must a simulation tool support all of interface laid out here to conform to the API? Said differently, do we plan on having two parts of the API, a core portion we expect all tools to implement and an optional portion that any tool can implement but isn't required to (and if it is implemented it must conform to this standard)?
2) For things like nodal voltages and branch currents (where there are many values most of which are not needed by any other federate most of the time), do we require the creation of all the handles to publish these out with the expectation that HELICS will cull all the unused ones? Or, conversely, do we require only those that are needed in the federation to be published? I lean towards the later.
3) The Domain API must define the form of the handle (pub, sub, input) and I have created these all as publications and subscriptions. I'm not 100% sure that is the right choice, though, and I'm not sure how we know what the right choice is.
4) More of a TODO: we need a common method of reporting errors, warnings or other similar methods if a federate is asked to do something that the underlying tool is not able to do. Maybe just a dedicated logging level?

A. Document Naming Conventions

`<GUID>` - Indicates "<GUID>" is variable whose value is defined based on the particular co-simulation (typically by values from the model that that simulation tool is actually solving).

`String` - Indicates a literal string.

II. Transmission Federates

Transmission federates should be expected to interface with distribution federates, controller federates, and possibly market federates. Distribution federates should be connected at the point of common coupling, which should be agreed upon by both transmission and distribution systems such that subtransmission and intermediate voltage sections are included in one side of the T&D model. Controller federates will control the dispatch of generation, storage, and potentially send a flexible load signal to nodes which are connected to loads at the distribution system level. A market federate would differ from a direct control federate in that it would calculate and disseminate clearing prices.

A. Output Handles (Publications)

1) PCC Voltage: Typically there may be only a single point of common coupling between a distribution system and the transmission system but there have been a few use cases where multiple coupling points have been implemented. Thus, this specification attempts to address the general case.

Distribution system simulators typically solve multi-phase unbalanced power flows while transmission system simulators assumed balanced flows. Thus, the publication of these values will be a single positive sequence value.

- **Handle type:** value
- **Pub. key:** pcc_`<GUID>`.v
- **Units:** kV
- **HELICS data type:** complex

III. Distribution Federates

Distribution Federates are primarily expected to interface with two other federate types: transmission system federates and controllers. The former will connect to the distribution system at one or more points of common coupling (PCC)

and the information exchange will be physical values used to solve powerflows in both systems. In defining the data exchange model, controllers present a unique problem in that any given asset modeled in the distribution system with a controllable component can have an external controller. Even assets not explicitly modeled as controllable (*e.g.* static loads) could effectively be made as such by adjusting their parameter values dynamically throughout the co-simulation. Thus, the range of values from the distribution simulator to which a given controller needs access is large. This API will define values that are likely to be needed and amend this list as use cases continue to expand the reach of data required.

Open question: Are these line-to-line or line-to-neutral voltages?

A. Output Handles (Publications)

1) PCC Load Power: Typically there may be only a single point of common coupling between a distribution system and the transmission system but there have been a few use cases where multiple coupling points have been implemented. Thus, this specification attempts to address the general case.

Distribution system simulators typically solve multi-phase unbalanced power flows while transmission system simulators assumed balanced flows. Thus, the publication of these values will be a single positive sequence value.

- **Handle type:** value
- **Pub. key:** pcc_<GUID>.pq
- **Units:** MVA
- **HELICS data type:** complex

2) Node voltage: All nodal voltages solved by the powerflow shall be available to external federates. Generally, these will be assumed to be presenting sensor measurements and thus will have a message handle. If a physical connection is being assumed between federates an additional value handle can be created as well. All phases are sent as a single complex vector with any unused phases marked by 0.

- **Handle type:** message or value
- **Pub. key:** node_<GUID>_msg.v or node_<GUID>.v
- **Units:** n/a or kV
- **HELICS data type:** string or complex vector
- **String format:** %f,%f,%f

3) Branch currents: All branch currents solved by the powerflow shall be available to external federates. Generally, these will be assumed to be presenting sensor measurements and thus will have a message handle. If a physical connection is being assumed between federates an additional value handle can be created as well. All phases are sent as a single complex vector with any unused phases marked by 99999+j99999.

- **Handle type:** message or value
- **Pub. key:** node_<GUID>_msg.i or node_<GUID>.i
- **Units:** n/a or kV
- **HELICS data type:** string or complex vector
- **String format:** %f,%f,,%f

4) ZIP Load: Loads can be modeled in a variety of manners and this handle is used to observe the state of those that are modeled using ZIP. All phases are represented as a complex value. Any portion of the load that is not modeled shall have its value set to null in the output string.

- **Handle type:** message
- **Pub. key:** zip_load_<GUID>.state
- **Units:** kVA
- **Data type:** string
- **String format:** %f,%f,%f;%f,%f,%f;%f,%f,%f where %f = <p>±j<q> for all phases of constant impedance, current, and power (in that order)

5) Tap-changing Voltage Regulator: Tap-changing voltage regulators are common voltage regulation equipment. The only information they need to provide for control purposes is the current tap position. All phases are sent in a single string with any unused phases marked by 0.

- **Handle type:** message
- **Pub. key:** regulator_<GUID>.position
- **Units:** n/a
- **Data type:** string
- **String format:** %i,%i,%i

6) Switched Capacitor: Switched capacitors are common voltage regulation equipment. The only information they need to provide for control purposes is their current state (open or closed). All phases are sent in a single string with any unused phases marked by null.

- **Handle type:** message
- **Pub. key:** capacitor_<GUID>.state
- **Units:** n/a
- **Data type:** string
- **String format:** %s,%s,%s where %s = OPEN or CLOSED or null

7) Switches: Switches are used for connecting and disconnecting portions of a distribution circuit for a variety of purposes. The only information they need to provide for control purposes is their current state (open or closed). All phases are sent in a single string with any unused phases marked by null.

- **Handle type:** message
- **Pub. key:** switch_<GUID>.state
- **Units:** n/a
- **HELICS data type:** string
- **String format:** %s,%s,%s where %s = OPEN or CLOSED or null

B. Input Handles (Subscriptions)

Controllers that need to manage equipment in the distribution system will need to send these control signals in a standardized manner. This section will define the required format of the publications from these controllers to interface correctly with the distribution federate. (This section is necessary because we probably won't have a controller domain API and thus we actually need to define the format of those signals to produce the standardization we need. Right? Or am I confused?)

1) Tap-changing Voltage Regulator: Tap-changing voltage regulators are common voltage regulation equipment. They receive command signals to move up or down a single position.

All phases are sent in a single string with any un-commanded phases marked by `null`.

- **Handle type:** message
- **Pub. key:** regulator_<GUID>.command
- **Units:** n/a
- **HELICS data type:** string
- **String format:** %s,%s,%s where %s = OPEN or CLOSE or null

2) Switched Capacitors: Switched capacitors are common voltage regulation equipment. They receive command signals to open or close. All phases are sent in a single string with any un-commanded phases marked by `null`.

- **Handle type:** message
- **Pub. key:** capacitor_<GUID>.command
- **Units:** n/a
- **HELICS data type:** string
- **String format:** %s,%s,%s where %s = OPEN or CLOSE or null

3) Switches: Switches are used for connecting and disconnecting portions of a distribution circuit for a variety of purposes. They receive command signals to open or close. All phases are sent in a single string with any un-commanded phases marked by `null`.

- **Handle type:** message
- **Pub. key:** switch_<GUID>.command
- **Units:** n/a
- **HELICS data typee:** string
- **String format:** %s,%s,%s where %s = OPEN or CLOSE or null

4) ZIP Load: Loads can be modeled in a variety of manners and this handle is used to adjust those that are modeled using ZIP. All phases can be adjusted independently with each phase being represented as a complex value. Any portion of the load that is not to be adjusted from its current value should have its value set to `null` in the command string.

- **Handle type:** message
- **Pub. key:** zip_load_<GUID>.command
- **Units:** kVA
- **HELICS data type:** string
- **String format:** %f,%f,%f;%f,%f,%f;%f,%f,%f where %f = <p>±j<q> for all phases of constant impedance, current, and power (in that order)

IV. Natural Gas Network Federates

There is substantial interest in co-simulations that couple simulators focused on natural gas network operations with power system simulators. Much of the focus is likely to center on co-simulating gas networks with electric power systems at the transmission level; co-simulation with distribution system federates is likely to be less common but may also be useful in certain applications. Although the most common coupling point between gas and power federates is likely to be gas-fired power plants (GFPPs), other potential coupling points include electric-driven compressors, underground storage or liquefied natural gas facilities, and electrolyzers.

A. Input Handles (Subscriptions)

1) Requested thermal power at GFPPs: Represents the amount of energy that GFPPs would consume when operating at their active power set points. Can be computed based on power plant dispatch and heat rate; values for thermal power can be converted to gas flow if other variables such as gas composition and pressure are fixed. As GFPPs are typically the first to be curtailed in cases of gas constraints (assuming they do not have firm contracts), the requested thermal power can be treated as a set point which may be violated if necessary.

- **Handle type:** value
- **Pub. key:** node_<GUID>.requested
- **Units:** MW (thermal)
- **HELICS data type:** double

2) Max compressor station load: Maximum electricity consumption at compressor stations. In many simulations electricity consumption at compressor stations is likely to be treated as a uncurtailable load, but using this subscription would allow the power federate to treat compressor stations as a controllable demand response resource that then constrains the gas simulation.

- **Handle type:** value
- **Pub. key:** compressor_<GUID>.power_max
- **Units:** MW
- **HELICS data type:** double

3) Electrolyzer injection profiles: Injection of gas into the gas network based on operation of electrolyzer objects in a power federate. Since high renewable systems may have variable electrolyzers profiles, this publication would allow for feedback from the power system into gas injection. This value would also need to be accompanied with information on gas composition and energy content.

- **Handle type:** value
- **Pub. key:** node_<GUID>.flow
- **Units:** sm^3/s
- **HELICS data type:** double

B. Output Handles (Publications)

1) Available thermal power: The complement of the requested thermal power subscription value, this publication indicates the amount of thermal power that is actually available to the GFPP given gas network constraints. The value is used by power federates to curtail gas-constrained power plants and re-dispatch as necessary. Available thermal power is typically calculated by the gas federate based on gas flow and composition/energy content.

- **Handle type:** value
- **Pub. key:** node_<GUID>.avail
- **Units:** MW (thermal)
- **HELICS data type:** double

2) Compressor station load: Represents the actual electric power consumption at compressor stations in the gas network. Passed to the power federate as a load.

- **Handle type:** value
- **Pub. key:** compressor_<GUID>.load
- **Units:** MW
- **HELICS data type:** double

3) Load at injection nodes: Electricity requirements at any injection node objects (e.g. underground storages, liquefied natural gas terminals).

- **Handle type:** value
- **Pub. key:** node_<GUID>.load
- **Units:** MW
- **HELICS data type:** double