**Microgrid Monitoring Protocol Specification**

Version: 1.0.0 – Active Development

Overview: This document provides the details of the monitoring protocol developed as part of Dr. Park’s Microgrid research in April 2018. The purpose of the protocol is to accurately communicate grid topology and live measurement data between JADE agents. This communication should enable live monitoring of Microgrid systems. There are fundamentally two types of data in this protocol: grid topology and live measurement. These two types of data are treated differently in the protocol implementation.

Grid topology data is treated as a graph data structure. Each controller agent is responsible for sending its “subgraph” to the receiving agent. The receiving agent the combines all subgraphs it has received into a final graph. Grid topology will rarely change in production systems and will change fairly slowly in development. As a result, grid topology is considered “semi-permanent” when it is sent. Sender agents are expected to send grid topology on startup. This grid topology data is expected to carry an expiration date. This puts the responsibility for determining how often topology data should be updated in the hands of sender implementation agents. Sender agents should send updates of their grid topology to the receiving agents just before the previous set of topology expires.

Measurement data is assumed to constantly change. It is also assumed to exist at some point on the grid topology graph. Each measurement consists of a measurement type, grid location, and a measurement.

Another goal of the communication protocol is to avoid re-inventing the wheel. JADE provides a transport layer to send Java objects as messages between JADE agents. Thus, we use Jade’s Agent Identifier (AID) identification system for agents, all communication is implemented via JADE INFORM messages containing externalized Java objects. As a result, the remainder of this document will describe Java classes. All data will be wrapped in “Message” objects in order to separate message processing logic from data model. The data model should function independently of the messaging layer used to move it.

All data objects implement the Externalizable interface from the Java serialization library. Each class definition is responsible for representing its own state in an output stream. As a result, all objects are compatible with ObjectOutputStream serialization. However, this protocol does not use Java serialization. Java serialization includes more redundant class definition data than actual data we want to transmit. Instead, objects call writeExternal() directly with a reference to an object output stream. As a result, only field data is outputted. Jade inform messages contain a byte[] that is the result of this process. Standard ObjectOutputStream serialization will/should work with all objects. However, this is not recommended because the stream adds class descriptors that contain redundant data. This overhead would choke message processing.

A reference implementation is provided. The java classes in the packages data and message in the reference implementation serve as the official protocol specification. The remainder of this document serves as an overview. Inheritance relationships have been omitted for clarity in the descriptions in the following document. However, inheritance from basic data types (such as “Graph”) is implemented in the Java classes.

Package: edu.ucdenver.park.microgrid.message

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name | Purpose | Properties | Example |
| MicrogridDatumMessage | The purpose of this message is to communicate one measurement at one point in time at one point in the grid to the receiving agent. This message is assumed to expire after it is sent. | Datum: A subclass implementation of the Datum class.  The datum will contain a grid location, measurement type, and measurement value (float or boolean). | Type: Voltage  Node: Generator A  Value: 15.0 |
| MicrogridGraphMessage | The purpose of this message is to communicate the subgraph that a sender agent knows about to the receiving agent. This message expires in order to keep the graph data up to date at the receiver agent. | Subgraph: An instance of MicrogridGraph containing  Expiration: timestamp (in the future) | Two generators linked to a battery then linked to the main grid  Timestamp: 1522804060 |

Package: edu.ucdenver.park.microgrid.data

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name | Purpose | Properties | Example |
| FloatMicrogridDatum | Represent one floating point measurement at one point in time at one point in the grid | Type  Timestamp  Node  Value | Type: Voltage  Timestamp: 1522804060  Node: Generator A  Value: 15.0 |
| BooleanMicrogridDatum | Represent one digital measurement at one point in time at one point in the grid | Type  Timestamp  Node  Value | Type: Fault  Timestamp: 1522804060  Node: Generator A  Value: True |
| MicrogridGraph | Represent an electrical grid as a graph  Despite containing the labels “to” and “from” on edges, all Java code ignores edge directions. The frontend updates edge directions based on amerage data. | Nodes  Edges | Nodes: Generator A, Battery A  Edges: Link between Generator A and Battery A |

|  |  |  |
| --- | --- | --- |
| Enum Name | Purpose | Values |
| MicrogridEdgeType | Represent the types of edges that can exist between nodes in the power grid. | BUS  COMMUNICATION\_LINK |
| MicrogridNodeType | Represent the types of nodes that can exist in the power grid. | CONTROLLER  BATTERY  GENERATOR  LOAD |
| MicrogridFloatMeasurementType | Represent the types of measurements that FloatMicrogridDatum can represent | VOLTS  AMPS  WATTS  FAULT |
| MicrogridBooleanMeasurementType | Represent the types of measurements that BooleanMicrogridDatum can represent | FAULT  WARNING |