Gridworks AtomicTNode

gridworks

Aug 03, 2023

CODE SUPPORT

1 Installation **3**

1.1 Hello AtomicTNode . . . 3
1.2 GridWorks Atn lexicon . . . 4
1.3 Type API Specs . . . 4
1.4 TwoChannelActorBase . . . 76
1.5 GridWorks DataClasses . . . 76
1.6 GridWorks Enums . . . 79
1.7 SDK for gridworks-atn Types . . . 86
1.8 Contributor Guide . . . 159
1.9 GridWorks Energy Consulting Code of Conduct . . . 161
1.10 License . . . 163

Python Module Index **165**

Index **167**

This is the GridWorks Python SDK for building Atomic Transactive Nodes, or AtomicTNodes. AtomicTNodes are the most fun and interesting GridWorks actors to design and build. They are what make electrical devices *transactive*. More specifically, each AtomicTNode is dedicated to the job of operating its very own Transactive Device. This is a juggling act. The AtomicTNode bids into electricity markets 24-7 so that its Transactive Device can thriftly taking advantage of the lowest prices. Simultaneously, the AtomicTNode has to respect the primary use of the device.

The GridWorks Millinocket demo is a simulation of hundreds of transactive thermal storage heating systems bidding into a local PNode market of ISO NE, the grid operator running wholesale electricity markets for New England. (The simulated AtomicTNodes in this demo uses proprietary code built on top of this SDK). The simulated results are impressive: significantly cutting the operating costs of home heating while simultaneously reducing the curtailment (i.e. turning off and wasting) of local wind farms.

Imagine I tell you that the transactive heating systems in the demo are real.

Do you believe me?

If you are the grid operator, are you prepared to settle financial transactions with these heaters? In order for an aggregation of AtomicTNode (aka, an AggregatedTNode) to participate in electricity markets, the grid operator needs a reason to believe that the AtomicTNode is:

- WHERE it claims to be on the electric grid;
- consuming WHAT it claims to be consuming (in terms of kWh of electricity); and
- that WHEN it claims to consume is accurate.

This calls a mechanism that enables distributed, decentralized, secure, and trustless transactions.

Enter the Algorand blockchain.

The first thing this SDK does is provide blockchain-related mechanisms for establishing these transactions. Go through the Validation and Atn Contract sequences in this documetation in order to learn how this works.

To explore the rest of GridWorks, visit the GridWorks docs.

CHAPTER

ONE

INSTALLATION

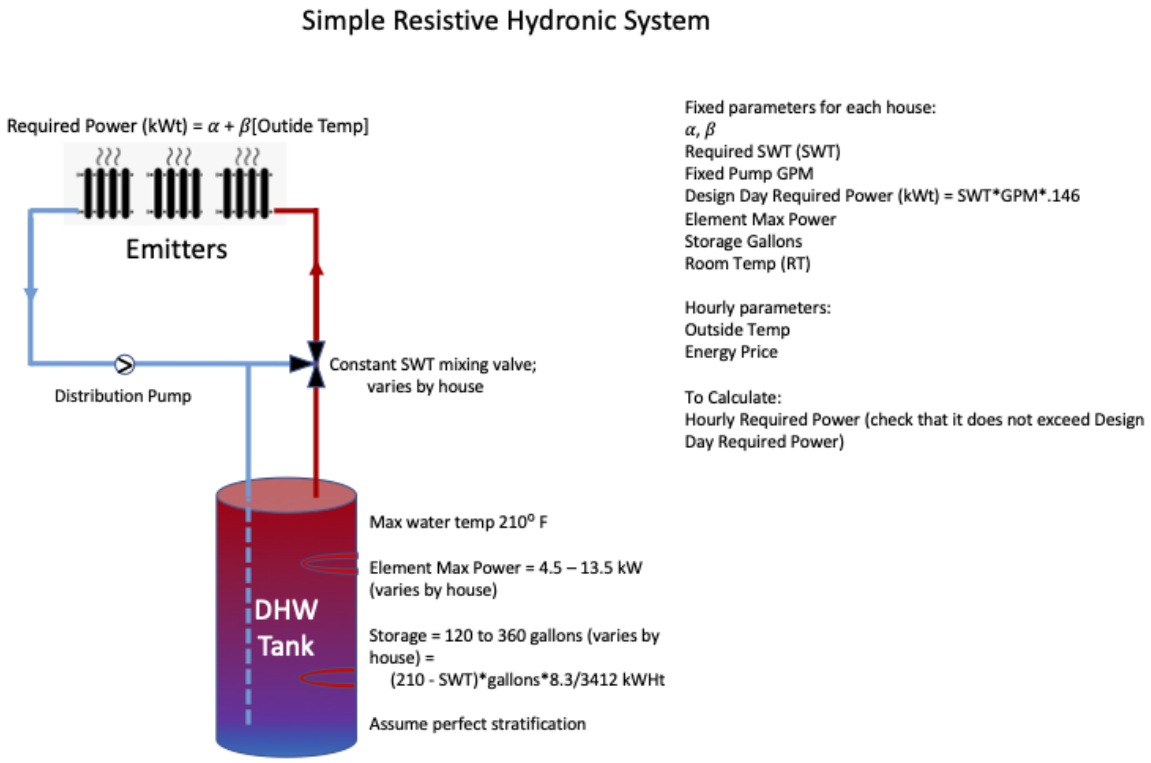
Note: gridworks-atn requires python 3.10 or higher.

```
(venv)$ pip install gridworks-atn
```

1.1 Hello AtomicTNode

AtomicTNodes intro - what they are, what they do.

Walk through the tutorial using an open-source demo strategy for what we call a simple resistive hydronic heating system. This code emakes some simplifying assumptions that one would want to adjust for better performance in the field, but captures many of the core functional concepts of an AtomicTNode operating for a thermal storage asset that is indeed an appropriate part of the mix for a high renewables future.



1.2 GridWorks Atn lexicon

See GridWorks Lexicon.

1.3 Type API Specs

1.3.1 AcceptedBid

```
{
  "gwapi": "001",
  "type_name": "accepted.bid",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Bid acceptance sent from MarketMaker to a market partipant. This is a␣
↪legally binding contract for the bidder to consume or produce the quantity in its Bid␣
↪consistent with the actual price.",
  "url": "https://gridworks.readthedocs.io/en/latest/market-bid.html",
  "formats": {
```

(continues on next page)

(continued from previous page)


```
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    },
    "MarketSlotNameLrdFormat": {
      "type": "string",
      "description": "",
      "example": ""
    }
  },
  "properties": {
    "MarketSlotName": {
      "type": "string",
      "format": "MarketSlotNameLrdFormat",
      "title": "",
      "required": true
    },
    "BidderAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "PqPairs": {
      "type": "price.quantity.unitless.000",
      "title": "",
      "required": true
    },
    "ReceivedTimeUnixNs": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "accepted.bid.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.2 AtnBid


```
{
  "gwapi": "001",
  "type_name": "atn.bid",
  "version": "001",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "AtomicTNode bid sent to a MarketMaker",
  "url": "https://gridworks.readthedocs.io/en/latest/market-bid.html",
  "formats": {
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    },
    "MarketSlotNameLrdFormat": {
      "type": "string",
      "description": "",
      "example": ""
    },
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "enums": {
    "MarketPriceUnit000": {
      "type": "string",
      "name": "market.price.unit.000",
      "description": "Price unit assigned to MarketMaker MarketType",
      "oneOf": [
        {
          "const": "00000000",
          "title": "USDPerMWh",
```


(continues on next page)

(continued from previous page)

```
          "description": ""
        }
      ]
    },
    "MarketTypeName000": {
      "type": "string",
      "name": "market.type.name.000",
      "description": "Categorizes different markets run by MarketMaker",
      "oneOf": [
        {
          "const": "00000000",
          "title": "unknown",
          "description": "Default unknown"
        },
        {
          "const": "d20b81e4",
          "title": "rt5gate5",
          "description": "Real-time energy, 5 minute MarketSlots, gate closing 5 minutes␣
↪prior to start"
        },
        {
          "const": "b36cbfb4",
          "title": "rt60gate5",
          "description": "Real-time energy, 60 minute MarketSlots, gate closing 5␣
↪minutes prior to start"
        },
        {
          "const": "94a3fe9b",
          "title": "da60",
          "description": "Day-ahead energy, 60 minute MarketSlots"
        },
        {
          "const": "5f335bdb",
          "title": "rt60gate30",
          "description": "Real-time energy, 60 minute MarketSlots, gate closing 30␣
↪minutes prior to start"
        },
        {
          "const": "01a84101",
          "title": "rt15gate5",
          "description": "Real-time energy, 15 minute MarketSlots, gate closing 5␣
↪minutes prior to start"
        },
        {
          "const": "e997ccfb",
          "title": "rt30gate5",
          "description": "Real-time energy, 30 minute MarketSlots, gate closing 5␣
↪minutes prior to start"
        },
        {
          "const": "618f9c0a",
          "title": "rt60gate30b",
```

(continues on next page)

(continued from previous page)

```
          "description": "Real-time energy, 30 minute MarketSlots, gate closing 5␣
↪minutes prior to start, QuantityUnit AvgkW"
        }
      ]
    },
    "MarketQuantityUnit000": {
      "type": "string",
      "name": "market.quantity.unit.000",
      "description": "Quantity unit assigned to MarketMaker MarketType",
      "oneOf": [
        {
          "const": "00000000",
          "title": "AvgMW",
          "description": ""
        },
        {
          "const": "c272f3b3",
          "title": "AvgkW",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "BidderAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "BidderGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "MarketSlotName": {
      "type": "string",
      "format": "MarketSlotNameLrdFormat",
      "title": "",
      "required": true
    },
    "PqPairs": {
      "type": "price.quantity.unitless.000",
      "title": "Price Quantity Pairs",
      "description": "The list of Price Quantity Pairs making up the bid. The units are␣
↪provided by the AtnBid.PriceUnit and AtnBid.QuantityUnit.",
      "required": true
    },
    "InjectionIsPositive": {
      "type": "boolean",
      "title": "",
```

(continues on next page)

(continued from previous page)


```
      "required": true
    },
    "PriceUnit": {
      "type": "string",
      "format": "MarketPriceUnit000",
      "title": "",
      "required": true
    },
    "QuantityUnit": {
      "type": "string",
      "format": "MarketQuantityUnit000",
      "title": "",
      "required": true
    },
    "SignedMarketFeeTxn": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "atn.bid.001",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "001",
      "required": true
    }
  },
  "axioms": {
    "Axiom1": {
      "title": "PqPairs PriceMax matches MarketType",
      "description": "There is a GridWorks global list of MarketTypes (a GridWorks type),
↪ identified by their MarketTypeNames (a GridWorks enum). The MarketType has a PriceMax,
↪ which must be the first price of the first PriceQuantity pair in PqPairs."
    },
    "Axiom2": {
      "title": "",
      "description": ""
    }
  }
}
```

1.3.3 AtnOutsideTempRegrCoeffs

```
{
  "gwapi": "001",
  "type_name": "atn.outside.temp.regr.coeffs",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": ". Coefficients for a linear regression of avg power leaving a building␣
↪as a function of weather: PowerOut = Alpha + Beta * OutsideTempF These are an example␣
↪of Slowly Varying State variables maintained for a thermal storage heating Terminal␣
↪Asset by its AtomicTNode and Scada.",
  "url": "https://gridworks-atn.readthedocs.io/en/latest/data-categories.html#slowly-
↪varying-state-variables",
  "properties": {
    "Alpha": {
      "type": "integer",
      "title": "Alpha (units: W)",
      "required": true
    },
    "Beta": {
      "type": "number",
      "title": "Beta (units: W / deg F)  ",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "atn.outside.temp.regr.coeffs.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.4 AtnParams

```
{
  "gwapi": "001",
  "type_name": "atn.params",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Generic AtnParams. This is a partial type, which is expected to be␣
↪satisfied by all types starting with atn.params (like atn.params.
↪heatpumpwithbooststore). It is used to describe the generic info required for␣
↪configuring an AtomicTNode.",
  "formats": {
    "LeftRightDot": {
```

(continues on next page)

(continued from previous page)

```
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "properties": {
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "HomeCity": {
      "type": "string",
      "title": "",
      "required": true
    },
    "TimezoneString": {
      "type": "string",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "atn.params.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.5 AtnParamsBrickstorageheater

```
{
  "gwapi": "001",
  "type_name": "atn.params.brickstorageheater",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Params for a BrickStorageHeater Atn Strategy. Parameters for a model␣
↪of a ceramic brick thermal storage room unit, using a two-dimensional Dijstra graph␣
↪parameterized by 'fullness of the thermal battery' and time.",
  "url": "https://gridworks-atn.readthedocs.io/en/latest/brick-storage-heater.html",
  "formats": {
    "LeftRightDot": {
```

(continues on next page)

(continued from previous page)

```
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "enums": {
    "DistributionTariff000": {
      "type": "string",
      "name": "distribution.tariff.000",
      "description": "Name of distribution tariff of local network company/utility",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
        },
        {
          "const": "2127aba6",
          "title": "VersantA1StorageHeatTariff",
          "url": "https://github.com/thegridelectric/gridworks-ps/blob/dev/input_data/
↪electricity_prices/isone/distp__w.isone.stetson__2022__gw.me.versant.a1.res.ets.csv",
          "description": "Versant is a utility serving customers in Maine, and in␣
↪particular serves much of the area behind the Keene Rd Constraint in the [GridWorks␣
↪Millinocket Demo](https://gridworks.readthedocs.io/en/latest/millinocket-demo.html
↪#background). Alternately known as the 'Home Eco Rate With Bonus Meter, Time-of-Use.'␣
↪Look for rate A1 in Versant [rate schedules](https://www.versantpower.com/residential/
↪rates/rates-schedules/); details are also available [here](https://drive.google.com/
↪drive/u/0/folders/1mhIeNj2JWVyIJrQnSHmBDOkBpNnRRVKB). More: Service under this rate␣
↪will be available to residential customers with thermal energy storage devices,␣
↪electric battery storage devices, and/or vehicle chargers who agree to install a␣
↪second metered point of delivery. The customer will be subject to inspections to␣
↪ensure that the thermal storage device, electric battery storage device, and electric␣
↪vehicle charger(s) are sized appropriately for residential use. If the thermal storage␣
↪device, electric battery storage device, and electric vehicle charger(s) do not pass␣
↪Company inspection, then the service will be denied. Service will be single-phase,␣
↪alternating current, 60 hertz, at one standard secondary distribution voltage.␣
↪Customers taking service under this rate schedule are responsible for paying both␣
↪Distribution Service and Stranded Cost."
        },
        {
          "const": "ea5c675a",
          "title": "VersantATariff",
          "description": "Versant is a utility serving customers in Maine, and in␣
↪particular serves much of the area behind the Keene Rd Constraint in the [GridWorks␣
↪Millinocket Demo](https://gridworks.readthedocs.io/en/latest/millinocket-demo.html
↪#background). The A Tariff is their standard residential tariff. Look for rate A in␣
↪Versant [rate schedules](https://www.versantpower.com/residential/rates/rates-
↪schedules/)"
        },
        {
          "const": "54aec3a7",
```

(continues on next page)

(continued from previous page)

```
          "title": "VersantA20HeatTariff",
          "description": "Versant is a utility serving customers in Maine, and in␣
↪particular serves much of the area behind the Keene Rd Constraint in the [GridWorks␣
↪Millinocket Demo](https://gridworks.readthedocs.io/en/latest/millinocket-demo.html
↪#background).  This is an alternative tariff available for electric heat."
        }
      ]
    },
    "RecognizedCurrencyUnit000": {
      "type": "string",
      "name": "recognized.currency.unit.000",
      "description": "Unit of currency",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
        },
        {
          "const": "e57c5143",
          "title": "USD",
          "description": "US Dollar"
        },
        {
          "const": "f7b38fc5",
          "title": "GBP",
          "description": "Pounds sterling"
        }
      ]
    },
    "RecognizedTemperatureUnit000": {
      "type": "string",
      "name": "recognized.temperature.unit.000",
      "description": "Unit of temperature",
      "oneOf": [
        {
          "const": "00000000",
          "title": "C",
          "description": "Celcius"
        },
        {
          "const": "6f16ee63",
          "title": "F",
          "description": "Fahrenheit"
        }
      ]
    },
    "EnergySupplyType000": {
      "type": "string",
      "name": "energy.supply.type.000",
      "description": "",
      "oneOf": [
```

(continues on next page)

(continued from previous page)

```
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
        },
        {
          "const": "cb18f937",
          "title": "StandardOffer",
          "description": ""
        },
        {
          "const": "e9dc99a6",
          "title": "RealtimeLocalLmp",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "GNode Alias",
      "required": true
    },
    "HomeCity": {
      "type": "string",
      "title": "Home City",
      "required": true
    },
    "TimezoneString": {
      "type": "string",
      "title": "Timezone String",
      "required": true
    },
    "StorageSteps": {
      "type": "integer",
      "title": "Storage Steps",
      "description": "The number of nodes in each TimeSlice for the Dijkstra graph.",
      "required": true
    },
    "FloSlices": {
      "type": "integer",
      "title": "",
      "description": "The number of nodes for each storage level in the Dijstra graph.",
      "required": true
    },
    "SliceDurationMinutes": {
      "type": "integer",
      "title": "",
      "description": "The length of",
      "required": true
```

(continues on next page)

(continued from previous page)


```
    },
    "CurrencyUnit": {
      "type": "string",
      "format": "RecognizedCurrencyUnit000",
      "title": "",
      "required": true
    },
    "Tariff": {
      "type": "string",
      "format": "DistributionTariff000",
      "title": "",
      "required": true
    },
    "EnergyType": {
      "type": "string",
      "format": "EnergySupplyType000",
      "title": "",
      "required": true
    },
    "StandardOfferPriceDollarsPerMwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "DistributionTariffDollarsPerMwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "MaxBrickTempC": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "RatedMaxPowerKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "C": {
      "type": "number",
      "title": "",
      "required": true
    },
    "ROff": {
      "type": "number",
      "title": "",
      "required": true
    },
    "ROn": {
      "type": "number",
      "title": "",
```


(continues on next page)

(continued from previous page)

```
      "required": true
    },
    "RoomTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AnnualHvacKwhTh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "BetaOt": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TempUnit": {
      "type": "string",
      "format": "RecognizedTemperatureUnit000",
      "title": "",
      "required": true
    },
    "AmbientPowerInKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "HouseWorstCaseTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "atn.params.brickstorageheater.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.6 AtnParamsReport


```
{
  "gwapi": "001",
  "type_name": "atn.params.report",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "AtomicTNode reporting its AtnParams. Parameters like the size of the␣
↪thermal store.",
  "formats": {
    "ReasonableUnixTimeS": {
      "type": "string",
      "description": "Integer reflecting unix time seconds between 1970 and 3000",
      "example": ""
    },
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "properties": {
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "GNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "AtnParamsTypeName": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "TimeUnixS": {
      "type": "integer",
      "format": "ReasonableUnixTimeS",
      "title": "",
      "required": true
    },
    "IrlTimeUnixS": {
```


(continues on next page)

(continued from previous page)

```
      "type": "integer",
      "format": "ReasonableUnixTimeS",
      "title": "",
      "required": false
    },
    "Params": {
      "type": "atn.params.000",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "atn.params.report.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.7 AtnParamsSimpleresistivehydronic

```
{
  "gwapi": "001",
  "type_name": "atn.params.simpleresistivehydronic",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "enums": {
    "DistributionTariff000": {
      "type": "string",
      "name": "distribution.tariff.000",
      "description": "Name of distribution tariff of local network company/utility",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
```

(continues on next page)

(continued from previous page)

```
        },
        {
          "const": "2127aba6",
          "title": "VersantA1StorageHeatTariff",
          "url": "https://github.com/thegridelectric/gridworks-ps/blob/dev/input_data/
↪electricity_prices/isone/distp__w.isone.stetson__2022__gw.me.versant.a1.res.ets.csv",
          "description": "Versant is a utility serving customers in Maine, and in␣
↪particular serves much of the area behind the Keene Rd Constraint in the [GridWorks␣
↪Millinocket Demo](https://gridworks.readthedocs.io/en/latest/millinocket-demo.html
↪#background). Alternately known as the 'Home Eco Rate With Bonus Meter, Time-of-Use.'␣
↪Look for rate A1 in Versant [rate schedules](https://www.versantpower.com/residential/
↪rates/rates-schedules/); details are also available [here](https://drive.google.com/
↪drive/u/0/folders/1mhIeNj2JWVyIJrQnSHmBDOkBpNnRRVKB). More: Service under this rate␣
↪will be available to residential customers with thermal energy storage devices,␣
↪electric battery storage devices, and/or vehicle chargers who agree to install a␣
↪second metered point of delivery. The customer will be subject to inspections to␣
↪ensure that the thermal storage device, electric battery storage device, and electric␣
↪vehicle charger(s) are sized appropriately for residential use. If the thermal storage␣
↪device, electric battery storage device, and electric vehicle charger(s) do not pass␣
↪Company inspection, then the service will be denied. Service will be single-phase,␣
↪alternating current, 60 hertz, at one standard secondary distribution voltage.␣
↪Customers taking service under this rate schedule are responsible for paying both␣
↪Distribution Service and Stranded Cost. See attached csv for instantiation of this␣
↪rate as an 8760."
        },
        {
          "const": "ea5c675a",
          "title": "VersantATariff",
          "description": "Versant is a utility serving customers in Maine, and in␣
↪particular serves much of the area behind the Keene Rd Constraint in the [GridWorks␣
↪Millinocket Demo](https://gridworks.readthedocs.io/en/latest/millinocket-demo.html
↪#background). The A Tariff is their standard residential tariff. Look for rate A in␣
↪Versant [rate schedules](https://www.versantpower.com/residential/rates/rates-
↪schedules/)"
        },
        {
          "const": "54aec3a7",
          "title": "VersantA20HeatTariff",
          "description": "Versant is a utility serving customers in Maine, and in␣
↪particular serves much of the area behind the Keene Rd Constraint in the [GridWorks␣
↪Millinocket Demo](https://gridworks.readthedocs.io/en/latest/millinocket-demo.html
↪#background). This is an alternative tariff available for electric heat."
        }
      ]
    },
    "RecognizedCurrencyUnit000": {
      "type": "string",
      "name": "recognized.currency.unit.000",
      "description": "Unit of currency",
      "oneOf": [
        {
          "const": "00000000",
```

(continues on next page)

(continued from previous page)

```
          "title": "Unknown",
          "description": ""
        },
        {
          "const": "e57c5143",
          "title": "USD",
          "description": "US Dollar"
        },
        {
          "const": "f7b38fc5",
          "title": "GBP",
          "description": "Pounds sterling"
        }
      ]
    },
    "EnergySupplyType000": {
      "type": "string",
      "name": "energy.supply.type.000",
      "description": "",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
        },
        {
          "const": "cb18f937",
          "title": "StandardOffer",
          "description": ""
        },
        {
          "const": "e9dc99a6",
          "title": "RealtimeLocalLmp",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "HomeCity": {
      "type": "string",
      "title": "",
      "required": true
    },
    "TimezoneString": {
      "type": "string",
```

(continues on next page)

(continued from previous page)


```
      "title": "",
      "required": true
    },
    "StorageSteps": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "FloSlices": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "SliceDurationMinutes": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "CurrencyUnit": {
      "type": "string",
      "format": "RecognizedCurrencyUnit000",
      "title": "",
      "required": true
    },
    "Tariff": {
      "type": "string",
      "format": "DistributionTariff000",
      "title": "",
      "required": true
    },
    "EnergyType": {
      "type": "string",
      "format": "EnergySupplyType000",
      "title": "",
      "required": true
    },
    "StandardOfferPriceDollarsPerMwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "FlatDistributionTariffDollarsPerMwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StoreSizeGallons": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "MaxStoreTempF": {
```


(continues on next page)

(continued from previous page)

```
      "type": "integer",
      "title": "",
      "required": true
    },
    "ElementMaxPowerKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RequiredSourceWaterTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "CirculatorPumpGpm": {
      "type": "number",
      "title": "",
      "required": true
    },
    "ReturnWaterDeltaTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AnnualHvacKwhTh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AmbientPowerInKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "HouseWorstCaseTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "RoomTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StorePassiveLossRatio": {
      "type": "number",
      "title": "",
      "required": true
    },
    "AmbientTempStoreF": {
      "type": "integer",
      "title": "",
```

(continues on next page)

(continued from previous page)

```
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "atn.params.simpleresistivehydronic.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.8 BasegnodeScadaCreate

```
{
  "gwapi": "001",
  "type_name": "basegnode.scada.create",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Scada BaseGNode Creation. This is a payload designed to be sent from a␣
↪TaOwner to the GNodeFactory. The TaOwner creates a private Algorand key and puts it on␣
↪the Scada Device that will sense and control their TerminalAsset. The public address␣
↪is associated with the Scada GNode by the GNodeFactory.",
  "formats": {
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    },
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
```

(continues on next page)

(continued from previous page)

```
    }
  },
  "properties": {
    "TaAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "TerminalAsset Alias",
      "description": "GNodeAlias of the TerminalAsset that will be controlled by the new␣
↪SCADA GNode. The SCADA GNodeAlias will have '.scada' appended to this.",
      "required": true
    },
    "ScadaAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "Algorand address for the SCADA",
      "description": "The TaOwner makes the corresponding private key, puts it on the␣
↪SCADA device, and then sends this address to the GNodeFactory.",
      "required": true
    },
    "TaDaemonAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "Algorand address of the associated TaDaemon",
      "description": "The TaDaemonAddr will have the TaDeed, and can be used to verify␣
↪the public address of the TaOwner",
      "required": true
    },
    "GNodeRegistryAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "GNodeRegistry Algorand address",
      "description": "The GNodeRegistry that contains Make/Model information about the␣
↪SCADA and TerminalAsset",
      "required": true
    },
    "SignedProof": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "Recent transaction signed by the TaOwner",
      "description": "These will be replaced by composite transactions in next gen code.
↪",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "basegnode.scada.create.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
```

(continues on next page)

(continued from previous page)

```
      "required": true
    }
  },
  "axioms": {
    "Axiom1": {
      "title": "TaOwner is SignedProof signer",
      "description": "The TaDaemonAddr provides the public address for the TaOwner. This␣
↪TaOwnerAddr must match the signature on the SignedProof."
    },
    "Axiom2": {
      "title": "TaAlias matches TaDeed",
      "description": "The TaDaemonAddr owns a TaDeed for the TaAlias."
    }
  }
}
```

1.3.9 DiscoverycertAlgoCreate

```
{
  "gwapi": "001",
  "type_name": "discoverycert.algo.create",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    },
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    }
  },
  "enums": {
    "CoreGNodeRole000": {
      "type": "string",
      "name": "core.g.node.role.000",
      "description": "CoreGNodeRole assigned by GNodeFactory",
      "url": "https://gridworks.readthedocs.io/en/latest/core-g-node-role.html",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Other",
          "description": ""
        },
        {
```

(continues on next page)

(continued from previous page)


```
          "const": "0f8872f7",
          "title": "TerminalAsset",
          "description": ""
        },
        {
          "const": "d9823442",
          "title": "AtomicTNode",
          "description": ""
        },
        {
          "const": "86f21dd2",
          "title": "MarketMaker",
          "description": ""
        },
        {
          "const": "9521af06",
          "title": "AtomicMeteringNode",
          "description": ""
        },
        {
          "const": "4502e355",
          "title": "ConductorTopologyNode",
          "description": ""
        },
        {
          "const": "d67e564e",
          "title": "InterconnectionComponent",
          "description": ""
        },
        {
          "const": "7a8e4046",
          "title": "Scada",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "Role": {
      "type": "string",
      "format": "CoreGNodeRole000",
      "title": "",
      "required": true
    },
    "OldChildAliasList": {
      "type": "string",
```


(continues on next page)

(continued from previous page)

```
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "DiscovererAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": true
    },
    "SupportingMaterialHash": {
      "type": "string",
      "title": "",
      "required": true
    },
    "MicroLat": {
      "type": "integer",
      "title": "",
      "required": false
    },
    "MicroLon": {
      "type": "integer",
      "title": "",
      "required": false
    },
    "TypeName": {
      "type": "string",
      "value": "discoverycert.algo.create.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.10 DispatchContractConfirmed

```
{
  "gwapi": "001",
  "type_name": "dispatch.contract.confirmed",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Message sent from AtomicTNode back to SCADA via Rabbit . Paired with␣
↪join.dispatch.contract. Sent from AtomicTNode back to SCADA once the AtomicTNode has␣
↪successfully finished bootstrapping the Dispatch Contract and opted in. Once it has␣
↪done this, the Dispatch Contract is ready to collect audit information about␣
```

(continues on next page)

(continued from previous page)

```
↪heartbeats, dispatch, energy and power. https://gridworks.readthedocs.io/en/latest/
↪dispatch-contract.html",
  "formats": {
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "AtnParamsTypeName": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "SignedProof": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "",
      "required": true
    },
```

(continues on next page)

(continued from previous page)

```
    "Params": {
      "type": "atn.params.000",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "dispatch.contract.confirmed.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  },
  "axioms": {
    "Axiom1": {
      "title": "AtnParamsTypeName matches AtnParams",
      "description": "AtnParams must have"
    }
  }
}
```

1.3.11 FloParams

```
{
  "gwapi": "001",
  "type_name": "flo.params",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Base class for Forward Looking Optimizer params. Derived classes are␣
↪expected to have TypeNames enforced as literals that start with flo.params. E.g. flo.
↪params.brickstorageheater. This container is used for sending messages that include␣
↪flo.params (i.e, flo.params.report",
  "url": "https://gridworks-atn.readthedocs.io/en/latest/flo.html#flo-params",
  "formats": {
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
```

(continues on next page)

(continued from previous page)

```
  "properties": {
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "FloParamsUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "HomeCity": {
      "type": "string",
      "title": "",
      "required": true
    },
    "TimezoneString": {
      "type": "string",
      "title": "",
      "required": true
    },
    "StartYearUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartMonthUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartDayUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartHourUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartMinuteUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "flo.params.000",
      "title": "The type name"
```

(continues on next page)

(continued from previous page)

```
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.12 FloParamsBrickstorageheater

```
{
  "gwapi": "001",
  "type_name": "flo.params.brickstorageheater",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": ". FloParams for the BrickStorageHeater AtomicTNode strategy.",
  "url": "https://gridworks-atn.readthedocs.io/en/latest/brick-storage-heater.html",
  "formats": {
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "enums": {
    "RecognizedCurrencyUnit000": {
      "type": "string",
      "name": "recognized.currency.unit.000",
      "description": "Unit of currency",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
        },
        {
          "const": "e57c5143",
          "title": "USD",
          "description": "US Dollar"
        },
        {
          "const": "f7b38fc5",
```

(continues on next page)

(continued from previous page)

```
          "title": "GBP",
          "description": "Pounds sterling"
        }
      ]
    },
    "RecognizedTemperatureUnit000": {
      "type": "string",
      "name": "recognized.temperature.unit.000",
      "description": "Unit of temperature",
      "oneOf": [
        {
          "const": "00000000",
          "title": "C",
          "description": "Celcius"
        },
        {
          "const": "6f16ee63",
          "title": "F",
          "description": "Fahrenheit"
        }
      ]
    }
  },
  "properties": {
    "MaxBrickTempC": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "RatedMaxPowerKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "ROff": {
      "type": "number",
      "title": "",
      "required": true
    },
    "ROn": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RoomTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "CurrencyUnit": {
      "type": "string",
      "format": "RecognizedCurrencyUnit000",
```

(continues on next page)

(continued from previous page)


```
      "title": "",
      "required": true
    },
    "TempUnit": {
      "type": "string",
      "format": "RecognizedTemperatureUnit000",
      "title": "",
      "required": true
    },
    "TimezoneString": {
      "type": "string",
      "title": "",
      "required": true
    },
    "HomeCity": {
      "type": "string",
      "title": "",
      "required": true
    },
    "IsRegulating": {
      "type": "boolean",
      "title": "",
      "required": true
    },
    "StorageSteps": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "SliceDurationMinutes": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "PowerRequiredByHouseFromSystemAvgKwList": {
      "type": "number",
      "title": "",
      "required": true
    },
    "C": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RealtimeElectricityPrice": {
      "type": "number",
      "title": "",
      "required": true
    },
    "OutsideTempF": {
      "type": "number",
      "title": "",
```


(continues on next page)

(continued from previous page)

```
      "required": true
    },
    "DistributionPrice": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RtElecPriceUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "RegulationPrice": {
      "type": "number",
      "title": "",
      "required": true
    },
    "WeatherUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "DistPriceUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "RegPriceUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": false
    },
    "StartYearUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartMonthUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartDayUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartHourUtc": {
```

(continues on next page)

(continued from previous page)


```
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartMinuteUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartingStoreIdx": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AmbientPowerInKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "HouseWorstCaseTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "FloParamsUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "flo.params.brickstorageheater.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.13 FloParamsReport


```
{
  "gwapi": "001",
  "type_name": "flo.params.report",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Flo Params Report. Type used for a message provided by an AtomicTNode␣
↪or SCADA actor re the flo parameters just used to run a FLO.",
  "url": "https://gridworks-atn.readthedocs.io/en/latest/flo.html#flo-params",
  "formats": {
    "ReasonableUnixTimeS": {
      "type": "string",
      "description": "Integer reflecting unix time seconds between 1970 and 3000",
      "example": ""
    },
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "properties": {
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "GNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "FloParamsTypeName": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "FloParamsTypeVersion": {
      "type": "string",
      "title": "",
      "required": true
    },
    "ReportGeneratedTimeUnixS": {
```


(continues on next page)

(continued from previous page)

```
      "type": "integer",
      "format": "ReasonableUnixTimeS",
      "title": "",
      "required": true
    },
    "IrlTimeUnixS": {
      "type": "integer",
      "format": "ReasonableUnixTimeS",
      "title": "",
      "required": false
    },
    "Params": {
      "type": "flo.params.000",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "flo.params.report.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.14 FloParamsSimpleresistivehydronic

```
{
  "gwapi": "001",
  "type_name": "flo.params.simpleresistivehydronic",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
```

(continues on next page)

(continued from previous page)

```
  },
  "enums": {
    "DistributionTariff000": {
      "type": "string",
      "name": "distribution.tariff.000",
      "description": "Name of distribution tariff of local network company/utility",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
        },
        {
          "const": "2127aba6",
          "title": "VersantA1StorageHeatTariff",
          "url": "https://github.com/thegridelectric/gridworks-ps/blob/dev/input_data/
↪electricity_prices/isone/distp__w.isone.stetson__2022__gw.me.versant.a1.res.ets.csv",
          "description": "Versant is a utility serving customers in Maine, and in␣
↪particular serves much of the area behind the Keene Rd Constraint in the [GridWorks␣
↪Millinocket Demo](https://gridworks.readthedocs.io/en/latest/millinocket-demo.html
↪#background). Alternately known as the 'Home Eco Rate With Bonus Meter, Time-of-Use.'␣
↪Look for rate A1 in Versant [rate schedules](https://www.versantpower.com/residential/
↪rates/rates-schedules/); details are also available [here](https://drive.google.com/
↪drive/u/0/folders/1mhIeNj2JWVyIJrQnSHmBDOkBpNnRRVKB). More: Service under this rate␣
↪will be available to residential customers with thermal energy storage devices,␣
↪electric battery storage devices, and/or vehicle chargers who agree to install a␣
↪second metered point of delivery. The customer will be subject to inspections to␣
↪ensure that the thermal storage device, electric battery storage device, and electric␣
↪vehicle charger(s) are sized appropriately for residential use. If the thermal storage␣
↪device, electric battery storage device, and electric vehicle charger(s) do not pass␣
↪Company inspection, then the service will be denied. Service will be single-phase,␣
↪alternating current, 60 hertz, at one standard secondary distribution voltage.␣
↪Customers taking service under this rate schedule are responsible for paying both␣
↪Distribution Service and Stranded Cost. See attached csv for instantiation of this␣
↪rate as an 8760."
        },
        {
          "const": "ea5c675a",
          "title": "VersantATariff",
          "description": "Versant is a utility serving customers in Maine, and in␣
↪particular serves much of the area behind the Keene Rd Constraint in the [GridWorks␣
↪Millinocket Demo](https://gridworks.readthedocs.io/en/latest/millinocket-demo.html
↪#background). The A Tariff is their standard residential tariff. Look for rate A in␣
↪Versant [rate schedules](https://www.versantpower.com/residential/rates/rates-
↪schedules/)"
        },
        {
          "const": "54aec3a7",
          "title": "VersantA20HeatTariff",
          "description": "Versant is a utility serving customers in Maine, and in␣
↪particular serves much of the area behind the Keene Rd Constraint in the [GridWorks␣
↪Millinocket Demo](https://gridworks.readthedocs.io/en/latest/millinocket-demo.html
```

(continues on next page)

(continued from previous page)

```
↪#background). This is an alternative tariff available for electric heat."
        }
      ]
    },
    "RecognizedCurrencyUnit000": {
      "type": "string",
      "name": "recognized.currency.unit.000",
      "description": "Unit of currency",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
        },
        {
          "const": "e57c5143",
          "title": "USD",
          "description": "US Dollar"
        },
        {
          "const": "f7b38fc5",
          "title": "GBP",
          "description": "Pounds sterling"
        }
      ]
    },
    "EnergySupplyType000": {
      "type": "string",
      "name": "energy.supply.type.000",
      "description": "",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
        },
        {
          "const": "cb18f937",
          "title": "StandardOffer",
          "description": ""
        },
        {
          "const": "e9dc99a6",
          "title": "RealtimeLocalLmp",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "GNodeAlias": {
      "type": "string",
```

(continues on next page)

(continued from previous page)

```
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "FloParamsUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "HomeCity": {
      "type": "string",
      "title": "",
      "required": true
    },
    "TimezoneString": {
      "type": "string",
      "title": "",
      "required": true
    },
    "StartYearUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartMonthUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartDayUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartHourUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartMinuteUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StorageSteps": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StoreSizeGallons": {
      "type": "integer",
```

(continues on next page)

(continued from previous page)

```
      "title": "",
      "required": true
    },
    "MaxStoreTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "RatedPowerKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RequiredSourceWaterTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "CirculatorPumpGpm": {
      "type": "number",
      "title": "",
      "required": true
    },
    "ReturnWaterDeltaTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "RoomTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AmbientPowerInKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "HouseWorstCaseTempF": {
      "type": "number",
      "title": "",
      "required": true
    },
    "StorePassiveLossRatio": {
      "type": "number",
      "title": "",
      "required": true
    },
    "PowerLostFromHouseKwList": {
      "type": "number",
      "title": "",
      "required": true
```

(continues on next page)

(continued from previous page)

```
    },
    "AmbientTempStoreF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "SliceDurationMinutes": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "RealtimeElectricityPrice": {
      "type": "number",
      "title": "",
      "required": true
    },
    "DistributionPrice": {
      "type": "number",
      "title": "",
      "required": true
    },
    "OutsideTempF": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RtElecPriceUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "DistPriceUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "WeatherUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "CurrencyUnit": {
      "type": "string",
      "format": "RecognizedCurrencyUnit000",
      "title": "",
      "required": true
    },
    "Tariff": {
      "type": "string",
```

(continues on next page)

(continued from previous page)


```
      "format": "DistributionTariff000",
      "title": "",
      "required": true
    },
    "EnergyType": {
      "type": "string",
      "format": "EnergySupplyType000",
      "title": "",
      "required": true
    },
    "StandardOfferPriceDollarsPerMwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "FlatDistributionTariffDollarsPerMwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartingStoreIdx": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "flo.params.simpleresistivehydronic.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```


1.3.15 InitialTadeedAlgoCreate


```
{
  "gwapi": "001",
  "type_name": "initial.tadeed.algo.create",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "TaValidator sends to GNodeFactory to complete creation of an initial␣
↪TaDeed. If this message is valid, the GNodeFactory co-signs and submits the TaDeed␣
↪creation. In addition, the GnodeFactory creates a TerminalAsset with GNodeStatus␣
↪pending. For more information: [TaDeed](https://gridworks.readthedocs.io/en/latest/ta-
```


(continues on next page)

(continued from previous page)

```
↪deed.html) [TaValidator](https://gridworks.readthedocs.io/en/latest/ta-validator.html)
↪",
  "formats": {
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "ValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "Address of the TaValidator",
      "description": "The Algorand address of the TaValidator who is going to validate␣
↪the location, device type, and power metering of the TerminalAsset.",
      "required": true
    },
    "HalfSignedDeedCreationMtx": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "Algo mulit-transaction for TaDeed creation",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "initial.tadeed.algo.create.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  },
  "axioms": {
    "Axiom1": {
      "title": "Is correct Multisig",
      "description": "Decoded HalfSignedDeedCreationMtx must have type␣
```

(continues on next page)

(continued from previous page)

```
↪MultisigTransaction from the 2-sig MultiAccount [GnfAdminAddr, ValidatorAddr].",
      "url": "https://gridworks.readthedocs.io/en/latest/g-node-factory.html#gnfadminaddr
↪"
    },
    "Axiom2": {
      "title": "Creates Initial ASA TaDeed",
      "description": "The transaction must create an Algorand Standard Asset - Total is␣
↪1 - UnitName is TADEED - Manager is GnfAdminAddr - AssetName has the following␣
↪characteristics: - length <= 32 characters - LeftRightDot format - final word is '.ta'
↪",
      "url": "https://gridworks.readthedocs.io/en/latest/ta-deed.html#asa-tadeed-specs"
    },
    "Axiom3": {
      "title": "Mtx signed by TaValidator",
      "description": ""
    }
  }
}
```

1.3.16 InitialTadeedAlgoOptin

```
{
  "gwapi": "001",
  "type_name": "initial.tadeed.algo.optin",
  "version": "002",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Received by TaDaemon so that it can opt into intial TaDeed. The␣
↪TaDaemon must opt into the TaDeed before receiving it. This message prompts that␣
↪action.",
  "formats": {
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    },
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
```

(continues on next page)

(continued from previous page)

```
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "TerminalAssetAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "The GNodeAlias of the TerminalAsset",
      "required": true
    },
    "TaOwnerAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "The Algorand address of the owner for the TerminalAsset",
      "required": true
    },
    "ValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "Address of the TaValidator",
      "description": "The Algorand address of the TaValidator who has validated the␣
↪location, device type, and power metering of the TerminalAsset.",
      "required": true
    },
    "SignedInitialDaemonFundingTxn": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "",
      "description": "Funding transaction for the TaDaemon account, signed by the␣
↪TaOwner.",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "initial.tadeed.algo.optin.002",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "002",
      "required": true
    }
  },
  "axioms": {
    "Axiom1": {
      "title": "Is correct Multisig",
      "description": "Decoded SignedInitialDaemonFundingTxn must be a SignedTransaction␣
↪signed by TaOwnerAddr."
    },
    "Axiom2": {
```

(continues on next page)

(continued from previous page)

```
      "title": "TaDeed consistency",
      "description": "There is an ASA TaDeed created by and owned by the 2-sig␣
↪MultiAccount [GnfAdminAddr, ValidatorAddr], where the TaDeed's AssetName is equal to␣
↪the payload's TerminalAssetAlias."
    }
  }
}
```

1.3.17 InitialTadeedAlgoTransfer

```
{
  "gwapi": "001",
  "type_name": "initial.tadeed.algo.transfer",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "TaValidator sends to GNodeFactory after validating Transactive Device.␣
↪Once the TaValidator has done the initial on-site inspection of the Transactive Device␣
↪including its location and the type and quality of its power and energy metering, the␣
↪TaValidator lets the GNodeFactory know by sending this message. Note the message also␣
↪includes the lat/lon of the Transactive Device. On receiving and validating this␣
↪message, the GNodeFactory will co-sign the transfer and send the TaDeed to the␣
↪TaDaemon address. In addition, the GNodeFactory creates and sends a TaTradingRights␣
↪certificate to the TaDaemon address. Only once the GNodeFactory has verified that the␣
↪TaDaemon address owns the TaDeed and TaTradingRights will it change the GNodeStatus of␣
↪the associated TerminalAsset from Pending to Active. [GNodeStatus](https://gridworks.
↪readthedocs.io/en/latest/g-node-status.html) [TaDeed](https://gridworks.readthedocs.io/
↪en/latest/ta-deed.html) [TaTradingRights](https://gridworks.readthedocs.io/en/latest/
↪ta-trading-rights.html) [TaValidator](https://gridworks.readthedocs.io/en/latest/ta-
↪validator.html) [TerminalAsset](https://gridworks.readthedocs.io/en/latest/terminal-
↪asset.html) [Transactive Device](https://gridworks.readthedocs.io/en/latest/
↪transactive-device.html)",
  "formats": {
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
```

(continues on next page)

(continued from previous page)

```
  "properties": {
    "MicroLat": {
      "type": "integer",
      "title": "",
      "description": "The Latitude of the Transactive Device, times 10^6",
      "required": true
    },
    "MicroLon": {
      "type": "integer",
      "title": "",
      "description": "The Longitude of the Transactive Device, times 10^6",
      "required": true
    },
    "ValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "description": "The Algoand address for the TaValidator who validated the location,
↪ metering and type of the Transactive Device.",
      "required": true
    },
    "TaDaemonAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "description": "The Algorand address for the TaDaemon which will own the TaDeed␣
↪and initially the TaTradingRights), as well as holding funds on behalf of the TaOwner.
↪",
      "required": true
    },
    "TaOwnerAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "description": "The Algorand address of the entity owning the Transactive Device,␣
↪and thus also the TerminalAsset",
      "required": true
    },
    "FirstDeedTransferMtx": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "",
      "description": "The half-signed multi transaction for transferring the TaDeed to␣
↪the TaDaemon.",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "initial.tadeed.algo.transfer.000",
      "title": "The type name"
    },
    "Version": {
```

(continues on next page)

(continued from previous page)

```
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  },
  "axioms": {
    "Axiom1": {
      "title": "Is correct Multisig",
      "description": "Decoded FirstDeedTransferMtx must have type MultisigTransaction␣
↪from the 2-sig MultiAccount [GnfAdminAddr, ValidatorAddr].",
      "url": "https://gridworks.readthedocs.io/en/latest/g-node-factory.html#gnfadminaddr
↪"
    },
    "Axiom2": {
      "title": "TaDaemon funded by TaOwner",
      "description": "The TaDaemonAddr was created with funding from the TaOwnerAddr,␣
↪and has sufficient funding according to the GNodeFactory."
    }
  }
}
```

1.3.18 JoinDispatchContract

```
{
  "gwapi": "001",
  "type_name": "join.dispatch.contract",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Sent from a Scada to its paired AtomicTNode on RabbitMQ. This is sent␣
↪as an invitation to join the DispatchContract. Upon receipt of this, the AtomicTNode␣
↪can check that the DispatchContract has finished the first part of its bootstrapping.␣
↪This means it is well-funded, and also has the Scada Cert Id and the Scada Addr␣
↪publicly available. The AtomicTNode can check these against the signature provided by␣
↪the SCADA in its invitation. An AtomicTNode actor accepts the invitation by finishing␣
↪the Dispatch Contract bootstrap (which it can only do if its Algorand Account holds␣
↪the associated TaTradingRights certificate) and then responding to the SCADA via␣
↪RabbitMQ with a dispatch.contract.confirmed payload. https://gridworks.readthedocs.io/
↪en/latest/dispatch-contract.html",
  "formats": {
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
```

(continues on next page)

(continued from previous page)

```
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "DispatchContractAppId": {
      "type": "integer",
      "minimum": 0,
      "title": "",
      "required": true
    },
    "SignedProof": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "join.dispatch.contract.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  },
  "axioms": {
```

(continues on next page)

(continued from previous page)

```
    "Axiom0": {
      "title": "ScadaCert matches FromGNodeAlias",
      "description": "The name in the ScadaCert should be the GNodeAlias of the␣
↪TerminalAsset corresponding to the sending SCADA. Therefore, FromGNodeAlias should be␣
↪equal to the name of the ScadaCert ASA with `.scada` appended."
    }
  }
}
```

1.3.19 LatestPrice

```
{
  "gwapi": "001",
  "type_name": "latest.price",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Latest Price for a MarketType, sent by a MarketMaker. The price of the␣
↪current MarketSlot",
  "url": "https://gridworks.readthedocs.io/en/latest/market-slot.html",
  "formats": {
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "IsoFormat": {
      "type": "string",
      "description": "",
      "example": ""
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    },
    "MarketSlotNameLrdFormat": {
      "type": "string",
      "description": "",
      "example": ""
    }
  },
  "enums": {
    "MarketPriceUnit000": {
      "type": "string",
      "name": "market.price.unit.000",
      "description": "Price unit assigned to MarketMaker MarketType",
      "oneOf": [
        {
          "const": "00000000",
```

(continues on next page)

(continued from previous page)

```
          "title": "USDPerMWh",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "PriceTimes1000": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "PriceUnit": {
      "type": "string",
      "format": "MarketPriceUnit000",
      "title": "",
      "required": true
    },
    "MarketSlotName": {
      "type": "string",
      "format": "MarketSlotNameLrdFormat",
      "title": "",
      "required": true
    },
    "IrlTimeUtc": {
      "type": "string",
      "format": "IsoFormat",
      "title": "",
      "required": false
    },
    "MessageId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": false
    },
    "TypeName": {
      "type": "string",
      "value": "latest.price.000",
      "title": "The type name"
```

(continues on next page)

(continued from previous page)

```
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.20 MarketSlot

```
{
  "gwapi": "001",
  "type_name": "market.slot",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "MarketSlot",
  "url": "https://gridworks.readthedocs.io/en/latest/market-slot.html",
  "formats": {
    "ReasonableUnixTimeS": {
      "type": "string",
      "description": "Integer reflecting unix time seconds between 1970 and 3000",
      "example": ""
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "properties": {
    "Type": {
      "type": "market.type.gt.000",
      "title": "",
      "required": true
    },
    "MarketMakerAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "StartUnixS": {
      "type": "integer",
      "format": "ReasonableUnixTimeS",
      "title": "",
      "required": true
    },
```

(continues on next page)

(continued from previous page)

```
    "TypeName": {
      "type": "string",
      "value": "market.slot.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.21 MarketTypeGt

```
{
  "gwapi": "001",
  "type_name": "market.type.gt",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Used by MarketMakers to simultaneously run several different types of␣
↪Markets. A [MarketMaker](https://gridworks.readthedocs.io/en/latest/market-maker.html)␣
↪GNode can run several types of Markets. For example, it can run an hourly real-time␣
↪market and also an ancillary services market for Regulation. This is captured by the␣
↪concept of MarketType.",
  "url": "https://gridworks.readthedocs.io/en/latest/market-type.html",
  "enums": {
    "MarketPriceUnit000": {
      "type": "string",
      "name": "market.price.unit.000",
      "description": "Price unit assigned to MarketMaker MarketType",
      "oneOf": [
        {
          "const": "00000000",
          "title": "USDPerMWh",
          "description": ""
        }
      ]
    },
    "RecognizedCurrencyUnit000": {
      "type": "string",
      "name": "recognized.currency.unit.000",
      "description": "Unit of currency",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
        },
```

(continues on next page)

(continued from previous page)

```
        {
          "const": "e57c5143",
          "title": "USD",
          "description": "US Dollar"
        },
        {
          "const": "f7b38fc5",
          "title": "GBP",
          "description": "Pounds sterling"
        }
      ]
    },
    "MarketTypeName000": {
      "type": "string",
      "name": "market.type.name.000",
      "description": "Categorizes different markets run by MarketMaker",
      "oneOf": [
        {
          "const": "00000000",
          "title": "unknown",
          "description": "Default unknown"
        },
        {
          "const": "d20b81e4",
          "title": "rt5gate5",
          "description": "Real-time energy, 5 minute MarketSlots, gate closing 5 minutes␣
↪prior to start"
        },
        {
          "const": "b36cbfb4",
          "title": "rt60gate5",
          "description": "Real-time energy, 60 minute MarketSlots, gate closing 5␣
↪minutes prior to start"
        },
        {
          "const": "94a3fe9b",
          "title": "da60",
          "description": "Day-ahead energy, 60 minute MarketSlots"
        },
        {
          "const": "5f335bdb",
          "title": "rt60gate30",
          "description": "Real-time energy, 60 minute MarketSlots, gate closing 30␣
↪minutes prior to start"
        },
        {
          "const": "01a84101",
          "title": "rt15gate5",
          "description": "Real-time energy, 15 minute MarketSlots, gate closing 5␣
↪minutes prior to start"
        },
        {
```

(continues on next page)

(continued from previous page)

```
          "const": "e997ccfb",
          "title": "rt30gate5",
          "description": "Real-time energy, 30 minute MarketSlots, gate closing 5␣
↪minutes prior to start"
        },
        {
          "const": "618f9c0a",
          "title": "rt60gate30b",
          "description": "Real-time energy, 30 minute MarketSlots, gate closing 5␣
↪minutes prior to start, QuantityUnit AvgkW"
        }
      ]
    },
    "MarketQuantityUnit000": {
      "type": "string",
      "name": "market.quantity.unit.000",
      "description": "Quantity unit assigned to MarketMaker MarketType",
      "oneOf": [
        {
          "const": "00000000",
          "title": "AvgMW",
          "description": ""
        },
        {
          "const": "c272f3b3",
          "title": "AvgkW",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "Name": {
      "type": "string",
      "format": "MarketTypeName000",
      "title": "Name of the MarketType",
      "required": true
    },
    "DurationMinutes": {
      "type": "integer",
      "title": "Duration of MarketSlots, in minutes",
      "required": true
    },
    "GateClosingSeconds": {
      "type": "integer",
      "title": "Seconds before the start of a MarketSlot after which bids are not␣
↪accepted",
      "required": true
    },
    "PriceUnit": {
      "type": "string",
      "format": "MarketPriceUnit000",
```

(continues on next page)

(continued from previous page)


```
      "title": "Price Unit for market (e.g. USD Per MWh)",
      "required": true
    },
    "QuantityUnit": {
      "type": "string",
      "format": "MarketQuantityUnit000",
      "title": "Quantity Unit for market (e.g. AvgMW)",
      "required": true
    },
    "CurrencyUnit": {
      "type": "string",
      "format": "RecognizedCurrencyUnit000",
      "title": "Currency Unit for market (e.g. USD)",
      "required": true
    },
    "PriceMax": {
      "type": "integer",
      "title": "PMax, required for defining bids",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "market.type.gt.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```


1.3.22 NewTadeedAlgoOptin


```
{
  "gwapi": "001",
  "type_name": "new.tadeed.algo.optin",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
```


(continues on next page)

(continued from previous page)

```
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "NewTaDeedIdx": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "OldTaDeedIdx": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TaDaemonAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": true
    },
    "ValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": true
    },
    "SignedTaDeedCreationTxn": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "new.tadeed.algo.optin.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
```

(continues on next page)

(continued from previous page)

```
}
```

1.3.23 NewTadeedSend

```
{
  "gwapi": "001",
  "type_name": "new.tadeed.send",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "NewTaDeedIdx": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "OldTaDeedIdx": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TaDaemonAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": true
    },
    "ValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
```

(continues on next page)

(continued from previous page)

```
      "required": true
    },
    "SignedTadeedOptinTxn": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "new.tadeed.send.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.24 OldTadeedAlgoReturn

```
{
  "gwapi": "001",
  "type_name": "old.tadeed.algo.return",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
```

(continues on next page)

(continued from previous page)

```
    "OldTaDeedIdx": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TaDaemonAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": true
    },
    "ValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": true
    },
    "SignedNewDeedTransferTxn": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "old.tadeed.algo.return.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.25 PriceQuantity

```
{
  "gwapi": "001",
  "type_name": "price.quantity",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "enums": {
    "MarketPriceUnit000": {
      "type": "string",
      "name": "market.price.unit.000",
      "description": "Price unit assigned to MarketMaker MarketType",
```

(continues on next page)

(continued from previous page)

```
      "oneOf": [
        {
          "const": "00000000",
          "title": "USDPerMWh",
          "description": ""
        }
      ]
    },
    "MarketQuantityUnit000": {
      "type": "string",
      "name": "market.quantity.unit.000",
      "description": "Quantity unit assigned to MarketMaker MarketType",
      "oneOf": [
        {
          "const": "00000000",
          "title": "AvgMW",
          "description": ""
        },
        {
          "const": "c272f3b3",
          "title": "AvgkW",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "PriceTimes1000": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "QuantityTimes1000": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "PriceUnit": {
      "type": "string",
      "format": "MarketPriceUnit000",
      "title": "",
      "required": true
    },
    "QuantityUnit": {
      "type": "string",
      "format": "MarketQuantityUnit000",
      "title": "",
      "required": true
    },
    "InjectionIsPositive": {
      "type": "boolean",
      "title": "",
```

(continues on next page)

(continued from previous page)


```
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "price.quantity.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```


1.3.26 PriceQuantityUnitless


```
{
  "gwapi": "001",
  "type_name": "price.quantity.unitless",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "properties": {
    "PriceTimes1000": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "QuantityTimes1000": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "price.quantity.unitless.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.27 ScadaCertTransfer


```
{
  "gwapi": "001",
  "type_name": "scada.cert.transfer",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Scada Certificate Transfer. This is a payload designed to be sent from␣
↪a SCADA device to the GNodeFactory after the SCADA has opted into its certificate.",
  "formats": {
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i(
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xseU
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "TaAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "TerminalAsset Alias",
      "description": "GNodeAlias of the TerminalAsset for which the SCADA certificate is␣
↪issued. The ScadaCert can be found from this.",
      "required": true
    },
    "SignedProof": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "Signed Proof from the SCADA Actor",
      "description": "The Scada GNode has a ScadaAlgoAddr in the GNodeFactory database,␣
↪and the identity of the SCADA actor can be verified by this.",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "scada.cert.transfer.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
```


(continues on next page)

(continued from previous page)

```
      "title": "The type version",
      "default": "000",
      "required": true
    }
  },
  "axioms": {
    "Axiom1": {
      "title": "Scada is SignedProof signer",
      "description": "Axiom 1: Scada is SignedProof signer. There is a ScadaCert created␣
↪by the Gnf with this ta_alias, and the txn is the OptIn."
    }
  }
}
```

1.3.28 SimplesimDriverData

```
{
  "gwapi": "001",
  "type_name": "simplesim.driver.data",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Generic driver report for a simple Scada Simulation",
  "formats": {
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "simplesim.driver.data.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.29 SimplesimDriverDataBsh

```
{
  "gwapi": "001",
  "type_name": "simplesim.driver.data.bsh",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "PowerWatts": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StoreKwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "MaxStoreKwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "simplesim.driver.data.bsh.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.30 SimplesimDriverReport


```
{
  "gwapi": "001",
  "type_name": "simplesim.driver.report",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "DriverDataTypeName": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "DriverData": {
      "type": "simplesim.driver.data.000",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "simplesim.driver.report.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
```


(continues on next page)

(continued from previous page)

```
      "default": "000",
      "required": true
    }
  }
}
```

1.3.31 SimplesimSnapshotBrickstorageheater

```
{
  "gwapi": "001",
  "type_name": "simplesim.snapshot.brickstorageheater",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "PowerWatts": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StoreKwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
```

(continues on next page)

(continued from previous page)

```
    "MaxStoreKwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AboutTerminalAssetAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "simplesim.snapshot.brickstorageheater.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.32 SlaEnter

```
{
  "gwapi": "001",
  "type_name": "sla.enter",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "properties": {
    "TerminalAssetAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
```

(continues on next page)

(continued from previous page)

```
      "value": "sla.enter.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.33 TadeedSpecsHack

```
{
  "gwapi": "001",
  "type_name": "tadeed.specs.hack",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    }
  },
  "properties": {
    "TerminalAssetAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "MicroLat": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "MicroLon": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "DaemonPort": {
      "type": "integer",
      "title": "",
      "required": true
    },
```

(continues on next page)

(continued from previous page)

```
    "TypeName": {
      "type": "string",
      "value": "tadeed.specs.hack.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.3.34 TavalidatorcertAlgoCreate

```
{
  "gwapi": "001",
  "type_name": "tavalidatorcert.algo.create",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Used for Step 1 of TaValidator certification. Meant to be sent from a␣
↪pending TaValidator to the GNodeFactory (Gnf), to initiate the process of certifying␣
↪the pending TaValidator.",
  "url": "https://gridworks.readthedocs.io/en/latest/ta-validator.html",
  "formats": {
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "ValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "The address of the pending TaValidator",
      "required": true
```

(continues on next page)

(continued from previous page)

```
    },
    "HalfSignedCertCreationMtx": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "Algo multi-transaction for certificate creation, with 1 of 2 signatures",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "tavalidatorcert.algo.create.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  },
  "axioms": {
    "Axiom1": {
      "title": "Is correct Multisig",
      "description": "Decoded HalfSignedCertCreationMtx must have type␣
↪MultisigTransaction from the 2-sig MultiAccount [GnfAdminAddr, ValidatorAddr], signed␣
↪by ValidatorAddr.",
      "url": "https://gridworks.readthedocs.io/en/latest/g-node-factory.html#gnfadminaddr
↪"
    },
    "Axiom2": {
      "title": "Is AssetConfigTxn",
      "description": "The transaction must have type AssetConfigTxn."
    },
    "Axiom3": {
      "title": "Is ValidatorCert",
      "description": "For the asset getting created: Total is 1, Decimals is 0, UnitName␣
↪is VLDTR, Manager is GnfAdminAddr, AssetName is not blank.",
      "url": "https://gridworks.readthedocs.io/en/latest/ta-validator.html#tavalidator-
↪certificate"
    },
    "Axiom5": {
      "title": "Uniqueness",
      "description": "There must not already be a TaValidatorCert belonging to the 2-sig␣
↪[GnfAdminAddr, ValidatorAddr] address."
    }
  },
  "example": {
    "ValidatorAddr": "7QQT4GN3ZPAQEFCNWF5BMF7NULVK3CWICZVT4GM3BQRISD52YEDLWJ4MII",
    "HalfSignedCertCreationMtx":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪lXB23O98qsyuf7f9jqDu+WT2U/
↪KB53CPR+XSUWGh5nonEUdp63TDIEo3RocgKhdgGjdHhuiaRhcGFyhKJhbq9Nb2xseSBNZXRlcm1haWShbcQgi1hzb1WaDzF+215cR8
↪",
```

(continues on next page)

(continued from previous page)


```
    "TypeName": "tavalidatorcert.algo.create",
    "Version": "000"
  }
}
```


1.3.35 TavalidatorcertAlgoTransfer


```
{
  "gwapi": "001",
  "type_name": "tavalidatorcert.algo.transfer",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Used for Step 2 of TaValidator certification. Meant to be sent from a␣
↪pending TaValidator to the GNodeFactory (Gnf), so the Gnf will transfer its␣
↪ValidatorCert to the pending TaValidator's Algorand address.",
  "url": "https://gridworks.readthedocs.io/en/latest/ta-validator.html",
  "formats": {
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    },
    "AlgoMsgPackEncoded": {
      "type": "string",
      "description": "Error is not thrown with algosdk.encoding.future_msg_
↪decode(candidate)",
      "example":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "ValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "The address of the pending TaValidator",
      "required": true
    },
    "HalfSignedCertTransferMtx": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "Algo multi-transaction for certificate transfer, with 1 of 2 signatures",
      "required": true
    },
    "TypeName": {
      "type": "string",
```


(continues on next page)

(continued from previous page)

```
      "value": "tavalidatorcert.algo.transfer.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  },
  "axioms": {
    "Axiom1": {
      "title": "Is correct Multisig",
      "description": "Decoded HalfSignedCertTransferMtx must have type␣
↪MultisigTransaction from the 2-sig MultiAccount [GnfAdminAddr, ValidatorAddr], signed␣
↪by the ValidatorAddr",
      "url": "https://gridworks.readthedocs.io/en/latest/g-node-factory.html#gnfadminaddr
↪"
    },
    "Axiom2": {
      "title": "Transfers correct certificate",
      "description": "- The transaction must be the transfer of an Algorand Standard␣
↪Asset -The sender must be the 2-sig Multi [GnfAdminAddr, TaValidatorAddr], which also␣
↪created and owns the ASA - It must be getting sent to the ValidatorAddr -The ASA must␣
↪have: - Total = 1 - UnitName=VLDITR - GnfAdminAddr as manage - AssetName not blank -␣
↪The transfer amount must be 1",
      "url": "https://gridworks.readthedocs.io/en/latest/ta-validator.html#tavalidator-
↪certificate"
    },
    "Axiom3": {
      "title": "TaValidator has opted in",
      "description": "ValidatorAddr must be opted into the transferring ASA."
    },
    "Axiom4": {
      "title": "TaValidator has sufficient Algos",
      "description": "ValidatorAddr must have enough Algos to meet the GNodeFactory␣
↪criterion."
    }
  }
}
```

1.3.36 TerminalassetCertifyHack

```
{
  "gwapi": "001",
  "type_name": "terminalasset.certify.hack",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "formats": {
```

(continues on next page)

(continued from previous page)

```
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
      "example": "dw1.isone.me.freedom.apple"
    },
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    }
  },
  "properties": {
    "TerminalAssetAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "TaDaemonApiPort": {
      "type": "string",
      "title": "",
      "required": true
    },
    "TaDaemonApiFqdn": {
      "type": "string",
      "title": "",
      "required": true
    },
    "TaDaemonAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "terminalasset.certify.hack.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.4 TwoChannelActorBase

This is similar to the ActorBase available in the gridworks package (included as a sub-package here).

The difference is that ActorBase publishes and consumes from the same channel, while the TwoChannelActorBase publishes and consumes on different channels - which is the method that RabbitMq recommends. Having two channels helps prevent slow-downs and gridlock, which is especially important for AtomicTNodes as they send and receive a lot of messages.

At some point both will be combined. However, there is an intermittent bug in the TwoChannelActorBase, which is that if the publishing channel stops working correctly it does not get restarted. The currently implemented workaround for this is to have Supervisor GNodes use their consuming channel for both publishing and consuming. That channel is correctly restarted when it stops working. In addition, the Supervisor will detect if any of its Subordinates (e.g. an AtomicTNode) have a faulty publishing channel and will kill and restart them.

This class should **not** be initialized directly.

1.5 GridWorks DataClasses

class gridworks.data_classes.base_g_node.**BaseGNode**(*g_node_id*, **args*, ***kwargs*)

DataClass corresponding to base.g.node.gt type.

The global authority for state of BaseGNodes is managed by the GNodeFactory, and shared with AtomicTNodes from its World Registry. Do not update or create these objects except when starting up and using local data from a file (provided by the World to its docker instance) or in a direct message (Rest POST or rabbit broadcast) from the World.

[more info](https://gridworks.readthedocs.io/en/latest/g-node.html#basegnodes)

children()

Returns the list of BaseGnodes identifying this node as parent

Return type
List[BaseGNode]

is_copper()

Returns true if role is not other

Return type
bool

parent()

Raises: DcError if "natural" parent (as suggested by alias) is not Active, and either

- prev_alias is None, OR
- the parent as suggested by prev_alias is not Active and/or

does not exist.

Returns

BaseGNode. Parent BaseGNode

- If the parent as suggested by the alias exists as an

Active BaseGNode, returns that ("natural" parent)

- Else, if the parent as suggested by the prev_alais exists

as an active BaseGNode, returns that.

None.

- If alias is one word long (i.e. root of world)

Return type
BaseGNode | None

`classmethod parent_from_alias(alias)`

Returns

- BaseGNode. If the parent as suggested by the alias exists as an

Active BaseGNode, returns that. - None. If alias is one word long (i.e. root of world)

Parameters
alias (*str*) –

Return type
BaseGNode | None

`class gridworks.data_classes.g_node.GNode(g_node_id, *args, **kwargs)`

Parameters
g_node_id (*str*) –

Return type
GNode

`children()`

Returns the list of BaseGnodes identifying this node as parent.

Return type
List[GNode]

`parent()`

Raises: DcError if "natural" parent (as suggested by alias) is not Active, and either

- prev_alias is None, OR
- the parent as suggested by prev_alias is not Active and/or

does not exist.

Returns

BaseGNode. Parent BaseGNode

- If the parent as suggested by the alias exists as an

Active GNode, returns that ("natural" parent)

- Else, if the parent as suggested by the prev_alais exists

as an active GNode, returns that.

None.

- If alias is one word long (i.e. root of world)

Return type
GNode | None

classmethod parent_from_alias(*alias*)

Returns

- GNode. If the parent as suggested by the alias exists as an

Active BaseGNode, returns that. - None. If alias is one word long (i.e. root of world)

Parameters
alias (*str*) –

Return type
GNode | None

class gridworks.data_classes.g_node_instance.**GNodeInstance**(*g_node_instance_id*, **args*, ***kwargs*)

Parameters
g_node_instance_id (*str*) –

Return type
GNodeInstance

children()

Returns the list of BaseGnodes identifying this node as parent

Return type
List[GNodeInstance]

parent()

Raises: DcError if "natural" parent (as suggested by alias) is not Active, and either

- prev_alias is None, OR
- the parent as suggested by prev_alias is not Active and/or

does not exist.

Returns

BaseGNode. Parent BaseGNode

- If the parent as suggested by the alias exists as an

Active GNode, returns that ("natural" parent)

- Else, if the parent as suggested by the prev_alais exists

as an active GNode, returns that.

None.

- If alias is one word long (i.e. root of world)

Return type
GNodeInstance | None

classmethod parent_from_alias(*alias*)

Returns

- GNode. If the parent as suggested by the alias exists as an

Active BaseGNode, returns that. - None. If alias is one word long (i.e. root of world)

Parameters
alias (*str*) –

Return type
GNodeInstance | None

1.6 GridWorks Enums

GwSchema Enums used in gridworks

class gridworks.enums.**AlgoCertType**(*value*)

Used to distinguish ASA vs SmartSignature certificates

Choices and descriptions:

- ASA: Certificate based on Algorand Standard Asset
- SmartSig: Certificate based on Algorand Smart Signature

classmethod default()

Returns default value ASA

Return type
AlgoCertType

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**CoreGNodeRole**(*value*)

CoreGNodeRole assigned by GNodeFactory. [More Info](https://gridworks.readthedocs.io/en/latest/core-g-node-role.html).

Choices and descriptions:

- Other:
- TerminalAsset:
- AtomicTNode:
- MarketMaker:
- AtomicMeteringNode:
- ConductorTopologyNode:
- InterconnectionComponent:
- Scada:

classmethod default()

Returns default value Other

Return type
CoreGNodeRole

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**GNodeRole**(*value*)

Categorizes GNodes by their function within GridWorks. [More Info](https://gridworks.readthedocs.io/en/latest/g-node-role.html).

Choices and descriptions:

- GNode: Default value
- TerminalAsset: An avatar for a real-word Transactive Device. [More Info](https://gridworks.readthedocs.io/en/latest/transactive-device.html).
- AtomicTNode: Transacts in markets on behalf of, and controlling the power use of, a TerminalAsset. [More Info](https://gridworks.readthedocs.io/en/latest/atomic-t-node.html).
- MarketMaker: Runs energy markets at its Node in the GNodeTree. [More Info](https://gridworks.readthedocs.io/en/latest/market-maker.html).
- AtomicMeteringNode: Role of a GNode that will become an AtomicTNode, prior to it owning TaTradingRights
- ConductorTopologyNode: An avatar for a real-world electric grid node - e.g. a substation or transformer
- InterconnectionComponent: An avatar for a cable or wire on the electric grid
- World: Adminstrative GNode responsible for managing and authorizing instances. [More Info](https://gridworks.readthedocs.io/en/latest/world-role.html).
- TimeCoordinator: Responsible for managing time in simulations
- Supervisor: Responsible for GNode actors running in a container. [More Info](https://gridworks.readthedocs.io/en/latest/supervisor.html).
- Scada: GNode associated to the device and code that directly monitors and actuates a Transactive Device
- PriceService: Provides price forecasts for markets run by MarketMakers
- WeatherService: Provides weather forecasts
- AggregatedTNode: An aggregation of AtomicTNodes
- Persister: Responsible for acking events with delivery guarantees

classmethod default()

Returns default value GNode

Return type
GNodeRole

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**GNodeStatus**(*value*)

Enum for managing GNode lifecycle. [More Info](https://gridworks.readthedocs.io/en/latest/g-node-status.html).

Choices and descriptions:

- Unknown: Default value
- Pending: The GNode exists but cannot be used yet.
- Active: The GNode can be used.

- PermanentlyDeactivated: The GNode can no longer be used, now or in the future.
- Suspended: The GNode cannot be used, but may become active in the future.

classmethod default()

Returns default value Unknown

Return type
GNodeStatus

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**GniStatus**(*value*)

Enum for managing GNodeInstance lifecycle. [More Info](https://gridworks.readthedocs.io/en/latest/g-node-instance.html).

Choices and descriptions:

- Unknown: Default Value
- Pending: Has been created by the World, but has not yet finished provisioning
- Active: Active in its GridWorks world. If the GNodeInstance has an actor, that actor is communicating
- Done: No longer represents the GNode.

classmethod default()

Returns default value Unknown

Return type
GniStatus

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**MarketPriceUnit**(*value*)

Price unit assigned to MarketMaker MarketType

Choices and descriptions:

- USDPerMWh:

classmethod default()

Returns default value USDPerMWh

Return type
MarketPriceUnit

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**MarketQuantityUnit**(*value*)

Quantity unit assigned to MarketMaker MarketType

Choices and descriptions:

- AvgMW:
- AvgkW:

classmethod default()

Returns default value AvgMW

Return type
MarketQuantityUnit

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**MarketTypeName**(*value*)

Categorizes different markets run by MarketMaker

Choices and descriptions:

- unknown: Default unknown
- rt5gate5: Real-time energy, 5 minute MarketSlots, gate closing 5 minutes prior to start
- rt60gate5: Real-time energy, 60 minute MarketSlots, gate closing 5 minutes prior to start
- da60: Day-ahead energy, 60 minute MarketSlots
- rt60gate30: Real-time energy, 60 minute MarketSlots, gate closing 30 minutes prior to start
- rt15gate5: Real-time energy, 15 minute MarketSlots, gate closing 5 minutes prior to start
- rt30gate5: Real-time energy, 30 minute MarketSlots, gate closing 5 minutes prior to start
- rt60gate30b: Real-time energy, 30 minute MarketSlots, gate closing 5 minutes prior to start, QuantityUnit AvgkW

classmethod default()

Returns default value unknown

Return type
MarketTypeName

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**MessageCategory**(*value*)

Categorizes how GridWorks messages are sent and decoded/encoded

Choices and descriptions:

- Unknown: Default value
- RabbitJsonDirect: Serialized Json message sent on the world rabbit broker from one GNode actor to another
- RabbitJsonBroadcast: Serailized Json message broadcast on the world rabbit broker by a GNode actor

- RabbitGwSerial: GwSerial protocol message sent on the world rabbit broker
- MqttJsonBroadcast: Serialized Json message following MQTT topic format, sent on the world rabbit broker
- RestApiPost: REST API post
- RestApiPostResponse: REST API post response
- RestApiGet: REST API GET

classmethod default()

Returns default value Unknown

Return type
MessageCategory

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**MessageCategorySymbol**(*value*)

Shorthand symbols for MessageCategory000 Enum, used in meta-data like routing keys

Choices and descriptions:

- unknown: Default value
- rj: Serialized Json message sent on the world rabbit broker from one GNode actor to another
- rjb: Serailized Json message broadcast on the world rabbit broker by a GNode actor
- s: GwSerial protocol message sent on the world rabbit broker
- gw: Serialized Json message following MQTT topic format, sent on the world rabbit broker
- post: REST API post
- postack: REST API post response
- get: REST API GET

classmethod default()

Returns default value unknown

Return type
MessageCategorySymbol

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**RecognizedCurrencyUnit**(*value*)

Unit of currency

Choices and descriptions:

- Unknown:
- USD: US Dollar
- GBP: Pounds sterling

classmethod default()

Returns default value UNKNOWN

Return type
RecognizedCurrencyUnit

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**StrategyName**(*value*)

Used to assign code to run a particular GNodeInstance

Choices and descriptions:

- NoActor: Assigned to GNodes that do not have actors
- WorldA: Authority on GNodeInstances, and their private keys. Maintains a FastAPI used for relational information about backoffice information held locally and/or in the GNodeRegistry/GNodeFactory. [More Info](https://gridworks.readthedocs.io/en/latest/world-role.html)
- SupervisorA: A simple supervisor that monitors its supervised AtomicTNode GNode strategies via a heartbeat health check. [More Info](https://gridworks.readthedocs.io/en/latest/supervisor.html)
- AtnHeatPumpWithBoostStore: AtomicTNode for a heat pump thermal storage heating system with a boost element and a thermal

heated by the boost element. [More on AtomicTNodes](https://gridworks.readthedocs.io/en/latest/atomic-t-node.html)

- TcGlobalA: Used to manage the global time of the Gridworks system when run with

in a fully simulated universe. [More on TimeCoordinators](https://gridworks.readthedocs.io/en/latest/time-coordinator.html)

- MarketMakerA: Runs a two-sided market associated to its GNode as part of the copper GNode sub-tree. [More on MarketMakers](https://gridworks.readthedocs.io/en/latest/market-maker.html)
- AtnBrickStorageHeater: Publicly available Dijkstra-based AtomicTNode strategy for a brick storage heater. These heaters are rooom units that store heat in a brick core, are heated with resistive elements, and typically have a fan to blow air over the brick core. They are sometimes called Electric Thermal Storage (ETS) heaters, and in the UK are often called Economy 7 heaters or Night Storage Heaters. A strategy very similar to this was used by VCharge to manage an ETS fleet of several thousand heaters in Pennsylvania. This strategy is meant to serve as a template for other private strategies, and also allows for an end-to-end simulation of a realistic aggregated transactive load capable of generating a highly elastic bid curve [More Info](https://gridworks-atn.readthedocs.io/en/latest/brick-storage-heater.html)

classmethod default()

Returns default value NoActor

Return type
StrategyName

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**SupervisorContainerStatus**(*value*)

Manages lifecycle of the docker containers where GridWorks actors run

Choices and descriptions:

- Unknown: Default value
- Authorized: World has created the information for starting the container
- Launching: World has launched the container
- Provisioning: Container has started, but is going through its provisioning process
- Running: GNode actors in the container are active
- Stopped: Stopped
- Deleted: Deleted

classmethod default()

Returns default value Unknown

Return type
SupervisorContainerStatus

classmethod values()

Returns enum choices

Return type
List[str]

class gridworks.enums.**UniverseType**(*value*)

Allows for multiple GridWorks, in particular for development and shared simulations. [More Info](https://gridworks.readthedocs.io/en/latest/universe.html).

Choices and descriptions:

- Dev: Simulation running on a single computer.
- Hybrid: Anything goes.
- Production: Money at stake.

classmethod default()

Returns default value Dev

Return type
UniverseType

classmethod values()

Returns enum choices

Return type
List[str]

1.7 SDK for gridworks-atn Types

The Python classes enumerated below provide an interpretation of gridworks-atn type instances (serialized JSON) as Python objects. Types are the building blocks for all GridWorks APIs. You can read more about how they work here, and examine their API specifications here. The Python classes below also come with methods for translating back and forth between type instances and Python objects.

List of all the types

1.7.1 AcceptedBid

Python pydantic class corresponding to json type `accepted.bid`.

class gwatn.types.**AcceptedBid**(*, *MarketSlotName*, *BidderAlias*, *PqPairs*, *ReceivedTimeUnixNs*, *TypeName='accepted.bid'*, *Version='000'*)

Bid acceptance sent from MarketMaker to a market partipant.

This is a legally binding contract for the bidder to consume or produce the quantity in its Bid consistent with the actual price. [More info](https://gridworks.readthedocs.io/en/latest/market-bid.html).

Parameters

- **MarketSlotName** (*str*) –
- **BidderAlias** (*str*) –
- **PqPairs** (*List[PriceQuantityUnitless]*) –
- **ReceivedTimeUnixNs** (*int*) –
- **TypeName** (*Literal['accepted.bid']*) –
- **Version** (*str*) –

MarketSlotName:

- Description:
- Format: MarketSlotNameLrdFormat

BidderAlias:

- Description:
- Format: LeftRightDot

PqPairs:

- Description:

ReceivedTimeUnixNs:

- Description:

class gwatn.types.accepted_bid.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises

ValueError – if not LeftRightDot format

Parameters

v (*str*) –

class gwatn.types.accepted_bid.**check_is_market_slot_name_lrd_format**(*v*)

MaketSlotNameLrdFormat: the format of a MarketSlotName.

- The first word must be a MarketTypeName
- The last word (unix time of market slot start) must

be a 10-digit integer divisible by 300 (i.e. all MarketSlots start at the top of 5 minutes) - More strictly, the last word must be the start of a MarketSlot for that MarketType (i.e. divisible by 3600 for hourly markets) - The middle words have LeftRightDot format (GNodeAlias of the MarketMaker)

Example: rt60gate5.d1.isone.ver.keene.1673539200

Parameters
v (*str*) –

class gwatn.types.**AcceptedBid_Maker**(*market_slot_name*, *bidder_alias*, *pq_pairs*, *received_time_unix_ns*)

Parameters

- **market_slot_name** (*str*) –
- **bidder_alias** (*str*) –
- **pq_pairs** (*List[PriceQuantityUnitless]*) –
- **received_time_unix_ns** (*int*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (AcceptedBid) –

Return type
str

classmethod **type_to_tuple**(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
AcceptedBid

1.7.2 AtnBid

Python pydantic class corresponding to json type `atn.bid`.

class gwatn.types.**AtnBid**(*, *BidderAlias*, *BidderGNodeInstanceId*, *MarketSlotName*, *PqPairs*, *InjectionIsPositive=False*, *PriceUnit=MarketPriceUnit.USDPerMWh*, *QuantityUnit=MarketQuantityUnit.AvgMW*, *SignedMarketFeeTxn*, *TypeName='atn.bid'*, *Version='001'*)

AtomicTNode bid sent to a MarketMaker [More info](https://gridworks.readthedocs.io/en/latest/market-bid.html).

Parameters

- **BidderAlias** (*str*) –
- **BidderGNodeInstanceId** (*str*) –

- **MarketSlotName** (*str*) –
- **PqPairs** (*List[PriceQuantityUnitless]*) –
- **InjectionIsPositive** (*bool*) –
- **PriceUnit** (MarketPriceUnit) –
- **QuantityUnit** (MarketQuantityUnit) –
- **SignedMarketFeeTxn** (*str*) –
- **TypeName** (*Literal['atn.bid']*) –
- **Version** (*str*) –

classmethod check_axiom_1(*v*)

Axiom 1: PqPairs PriceMax matches MarketType. There is a GridWorks global list of MarketTypes (a GridWorks type), identified by their MarketTypeNames (a GridWorks enum). The MarketType has a PriceMax, which must be the first price of the first PriceQuantity pair in PqPairs.

Parameters
v (*dict*) –

Return type
dict

classmethod check_axiom_2(*v*)

Axiom 2: .

Parameters
v (*dict*) –

Return type
dict

BidderAlias:

- Description:
- Format: LeftRightDot

BidderGNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

MarketSlotName:

- Description:
- Format: MarketSlotNameLrdFormat

PqPairs:

- Description: Price Quantity Pairs. The list of Price Quantity Pairs making up the bid. The units are provided by the AtnBid.PriceUnit and AtnBid.QuantityUnit.

InjectionIsPositive:

- Description:

PriceUnit:

- Description:

QuantityUnit:

- Description:

SignedMarketFeeTxn:

- Description:
- Format: AlgoMsgPackEncoded

class gwatn.types.atn_bid.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.atn_bid.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.atn_bid.**check_is_market_slot_name_lrd_format**(*v*)

MaketSlotNameLrdFormat: the format of a MarketSlotName.

- The first word must be a MarketTypeName
- The last word (unix time of market slot start) must

be a 10-digit integer divisible by 300 (i.e. all MarketSlots start at the top of 5 minutes) - More strictly, the last word must be the start of a MarketSlot for that MarketType (i.e. divisible by 3600 for hourly markets) - The middle words have LeftRightDot format (GNodeAlias of the MarketMaker)

Example: rt60gate5.d1.isone.ver.keene.1673539200

Parameters
v (*str*) –

class gwatn.types.atn_bid.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.atn_bid.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**AtnBid_Maker**(*bidder_alias*, *bidder_g_node_instance_id*, *market_slot_name*, *pq_pairs*, *injection_is_positive*, *price_unit*, *quantity_unit*, *signed_market_fee_txn*)

Parameters

- **bidder_alias** (*str*) –
- **bidder_g_node_instance_id** (*str*) –
- **market_slot_name** (*str*) –
- **pq_pairs** (*List[PriceQuantityUnitless]*) –
- **injection_is_positive** (*bool*) –
- **price_unit** (MarketPriceUnit) –
- **quantity_unit** (MarketQuantityUnit) –
- **signed_market_fee_txn** (*str*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (AtnBid) –

Return type
str

classmethod **type_to_tuple**(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
AtnBid

1.7.3 AtnOutsideTempRegrCoeffs

Python pydantic class corresponding to json type `atn.outside.temp.regr.coeffs`.

class gwatn.types.**AtnOutsideTempRegrCoeffs**(***, *Alpha=200*, *Beta=-1.5*, *TypeName='atn.outside.temp.regr.coeffs'*, *Version='000'*)

.

Coefficients for a linear regression of avg power leaving a building as a function of weather:

PowerOut = Alpha + Beta * OutsideTempF

These are an example of Slowly Varying State variables maintained for a thermal storage heating Terminal Asset by its AtomicTNode and Scada. [More info](https://gridworks-atn.readthedocs.io/en/latest/data-categories.html#slowly-varying-state-variables).

Parameters

- **Alpha** (*int*) –
- **Beta** (*float*) –
- **TypeName** (*Literal['atn.outside.temp.regr.coeffs']*) –

- **Version** (*str*) –

Alpha:

- Description: Alpha (units: W)

Beta:

- Description: Beta (units: W / deg F)

class gwatn.types.**AtnOutsideTempRegrCoeffs_Maker**(*alpha*, *beta*)

Parameters

- **alpha** (*int*) –
- **beta** (*float*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (AtnOutsideTempRegrCoeffs) –

Return type
str

classmethod **type_to_tuple**(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
AtnOutsideTempRegrCoeffs

1.7.4 AtnParams

Python pydantic class corresponding to json type `atn.params`.

class gwatn.types.**AtnParams**(*, *GNodeAlias*, *HomeCity='MILLINOCKET_ME'*, *TimezoneString='US/Eastern'*, *TypeName='atn.params'*, *Version='000'*, ***extra_data*)

Generic AtnParams.

This is a partial type, which is expected to be satisfied by all types starting with atn.params (like atn.params.heatpumpwithbooststore). It is used to describe the generic info required for configuring an AtomicTNode.

Parameters

- **GNodeAlias** (*str*) –
- **HomeCity** (*str*) –
- **TimezoneString** (*str*) –
- **TypeName** (*str*) –
- **Version** (*str*) –
- **extra_data** (*Any*) –

GNodeAlias:

- Description:
- Format: LeftRightDot

HomeCity:

- Description:

TimezoneString:

- Description:

class gwatn.types.atn_params.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**AtnParams_Maker**(*g_node_alias*, *home_city*, *timezone_string*)

Parameters

- **g_node_alias** (*str*) –
- **home_city** (*str*) –
- **timezone_string** (*str*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (AtnParams) –

Return type
str

classmethod **type_to_tuple**(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
AtnParams

1.7.5 AtnParamsBrickstorageheater

Python pydantic class corresponding to json type `atn.params.brickstorageheater`.

class gwatn.types.**AtnParamsBrickstorageheater**(*, *GNodeAlias*, *HomeCity='MILLINOCKET_ME'*, *TimezoneString='US/Eastern'*, *StorageSteps=100*, *FloSlices=48*, *SliceDurationMinutes=60*, *CurrencyUnit=RecognizedCurrencyUnit.USD*, *Tariff=DistributionTariff.VersantA1StorageHeatTariff*, *EnergyType=EnergySupplyType.RealtimeLocalLmp*, *StandardOfferPriceDollarsPerMwh=110*, *DistributionTariffDollarsPerMwh=113*, *MaxBrickTempC=190*, *RatedMaxPowerKw=13.5*, *C=200*, *ROff=0.08*, *ROn=0.15*, *RoomTempF=70*, *AnnualHvacKwhTh=28125*, *BetaOt=158*, *TempUnit=RecognizedTemperatureUnit.F*, *AmbientPowerInKw=1.25*, *HouseWorstCaseTempF=-7*, *TypeName='atn.params.brickstorageheater'*, *Version='000'*)

Params for a BrickStorageHeater Atn Strategy.

Parameters for a model of a ceramic brick thermal storage room unit, using a two-dimensional Dijstra graph parameterized by 'fullness of the thermal battery' and time. [More info](https://gridworks-atn.readthedocs.io/en/latest/brick-storage-heater.html).

Parameters

- **GNodeAlias** (*str*) –
- **HomeCity** (*str*) –
- **TimezoneString** (*str*) –
- **StorageSteps** (*int*) –
- **FloSlices** (*int*) –
- **SliceDurationMinutes** (*int*) –
- **CurrencyUnit** (RecognizedCurrencyUnit) –
- **Tariff** (*DistributionTariff*) –
- **EnergyType** (*EnergySupplyType*) –
- **StandardOfferPriceDollarsPerMwh** (*int*) –
- **DistributionTariffDollarsPerMwh** (*int*) –
- **MaxBrickTempC** (*int*) –
- **RatedMaxPowerKw** (*float*) –
- **C** (*float*) –
- **ROff** (*float*) –
- **ROn** (*float*) –
- **RoomTempF** (*int*) –
- **AnnualHvacKwhTh** (*int*) –
- **BetaOt** (*int*) –
- **TempUnit** (*RecognizedTemperatureUnit*) –
- **AmbientPowerInKw** (*float*) –
- **HouseWorstCaseTempF** (*int*) –

- **TypeName** (*Literal['atn.params.brickstorageheater']*) –
- **Version** (*str*) –

GNodeAlias:

- Description: GNode Alias
- Format: LeftRightDot

HomeCity:

- Description: Home City

TimezoneString:

- Description: Timezone String

StorageSteps:

- Description: Storage Steps. The number of nodes in each TimeSlice for the Dijkstra graph.

FloSlices:

- Description: . The number of nodes for each storage level in the Dijstra graph.

SliceDurationMinutes:

- Description: . The length of

CurrencyUnit:

- Description:

Tariff:

- Description:

EnergyType:

- Description:

StandardOfferPriceDollarsPerMwh:

- Description:

DistributionTariffDollarsPerMwh:

- Description:

MaxBrickTempC:

- Description:

RatedMaxPowerKw:

- Description:

C:

- Description:

ROff:

- Description:

ROn:

- Description:

RoomTempF:

- Description:

AnnualHvacKwhTh:

- Description:

BetaOt:

- Description:

TempUnit:

- Description:

AmbientPowerInKw:

- Description:

HouseWorstCaseTempF:

- Description:

class gwatn.types.atn_params_brickstorageheater.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**AtnParamsBrickstorageheater_Maker**(*g_node_alias*, *home_city*, *timezone_string*, *storage_steps*, *flo_slices*, *slice_duration_minutes*, *currency_unit*, *tariff*, *energy_type*, *standard_offer_price_dollars_per_mwh*, *distribution_tariff_dollars_per_mwh*, *max_brick_temp_c*, *rated_max_power_kw*, *c*, *r_off*, *r_on*, *room_temp_f*, *annual_hvac_kwh_th*, *beta_ot*, *temp_unit*, *ambient_power_in_kw*, *house_worst_case_temp_f*)

Parameters

- **g_node_alias** (*str*) –
- **home_city** (*str*) –
- **timezone_string** (*str*) –
- **storage_steps** (*int*) –
- **flo_slices** (*int*) –
- **slice_duration_minutes** (*int*) –
- **currency_unit** (RecognizedCurrencyUnit) –
- **tariff** (*DistributionTariff*) –
- **energy_type** (*EnergySupplyType*) –
- **standard_offer_price_dollars_per_mwh** (*int*) –
- **distribution_tariff_dollars_per_mwh** (*int*) –
- **max_brick_temp_c** (*int*) –

- **rated_max_power_kw** (*float*) –
- **c** (*float*) –
- **r_off** (*float*) –
- **r_on** (*float*) –
- **room_temp_f** (*int*) –
- **annual_hvac_kwh_th** (*int*) –
- **beta_ot** (*int*) –
- **temp_unit** (*RecognizedTemperatureUnit*) –
- **ambient_power_in_kw** (*float*) –
- **house_worst_case_temp_f** (*int*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (AtnParamsBrickstorageheater) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
AtnParamsBrickstorageheater

1.7.6 AtnParamsReport

Python pydantic class corresponding to json type `atn.params.report`.

class gwatn.types.**AtnParamsReport**(*, *GNodeAlias*, *GNodeInstanceId*, *AtnParamsTypeName*, *TimeUnixS*, *IrlTimeUnixS=None*, *Params*, *TypeName='atn.params.report'*, *Version='000'*)

AtomicTNode reporting its AtnParams.

Parameters like the size of the thermal store.

Parameters

- **GNodeAlias** (*str*) –
- **GNodeInstanceId** (*str*) –
- **AtnParamsTypeName** (*str*) –
- **TimeUnixS** (*int*) –
- **IrlTimeUnixS** (*int | None*) –
- **Params** (AtnParams) –
- **TypeName** (*Literal['atn.params.report']*) –

- **Version** (*str*) –

GNodeAlias:

- Description:
- Format: LeftRightDot

GNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

AtnParamsTypeName:

- Description:
- Format: LeftRightDot

TimeUnixS:

- Description:
- Format: ReasonableUnixTimeS

IrlTimeUnixS:

- Description:
- Format: ReasonableUnixTimeS

Params:

- Description:

class gwatn.types.atn_params_report.**check_is_reasonable_unix_time_s**(*v*)

ReasonableUnixTimeS format: time in unix seconds between Jan 1 2000 and Jan 1 3000

Raises
ValueError – if not ReasonableUnixTimeS format

Parameters
v (*int*) –

class gwatn.types.atn_params_report.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.atn_params_report.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**AtnParamsReport_Maker**(*g_node_alias*, *g_node_instance_id*, *atn_params_type_name*, *time_unix_s*, *irl_time_unix_s*, *params*)

Parameters

- **g_node_alias** (*str*) –
- **g_node_instance_id** (*str*) –
- **atn_params_type_name** (*str*) –
- **time_unix_s** (*int*) –
- **irl_time_unix_s** (*int | None*) –
- **params** (AtnParams) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (AtnParamsReport) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
AtnParamsReport

1.7.7 AtnParamsSimpleresistivehydronic

Python pydantic class corresponding to json type `atn.params.simpleresistivehydronic`.

class gwatn.types.**AtnParamsSimpleresistivehydronic**(*, *GNodeAlias*, *HomeCity*, *TimezoneString*, *StorageSteps=100*, *FloSlices=48*, *SliceDurationMinutes=60*, *CurrencyUnit=RecognizedCurrencyUnit.USD*, *Tariff=DistributionTariff.VersantA1StorageHeatTariff*, *EnergyType=EnergySupplyType.RealtimeLocalLmp*, *StandardOfferPriceDollarsPerMwh=110*, *FlatDistributionTariffDollarsPerMwh=113*, *StoreSizeGallons=240*, *MaxStoreTempF=210*, *ElementMaxPowerKw=9.5*, *RequiredSourceWaterTempF=120*, *CirculatorPumpGpm=5.5*, *ReturnWaterDeltaTempF=20*, *AnnualHvacKwhTh=25000*, *AmbientPowerInKw=1.2*, *HouseWorstCaseTempF=-7*, *RoomTempF=68*, *StorePassiveLossRatio=0.005*, *AmbientTempStoreF=65*, *TypeName='atn.params.simpleresistivehydronic'*, *Version='000'*)

Parameters

- **GNodeAlias** (*str*) –
- **HomeCity** (*str*) –
- **TimezoneString** (*str*) –
- **StorageSteps** (*int*) –
- **FloSlices** (*int*) –
- **SliceDurationMinutes** (*int*) –
- **CurrencyUnit** (RecognizedCurrencyUnit) –
- **Tariff** (*DistributionTariff*) –
- **EnergyType** (*EnergySupplyType*) –
- **StandardOfferPriceDollarsPerMwh** (*int*) –
- **FlatDistributionTariffDollarsPerMwh** (*int*) –
- **StoreSizeGallons** (*int*) –
- **MaxStoreTempF** (*int*) –
- **ElementMaxPowerKw** (*float*) –
- **RequiredSourceWaterTempF** (*int*) –
- **CirculatorPumpGpm** (*float*) –
- **ReturnWaterDeltaTempF** (*int*) –
- **AnnualHvacKwhTh** (*int*) –
- **AmbientPowerInKw** (*float*) –
- **HouseWorstCaseTempF** (*int*) –

- **RoomTempF** (*int*) –
- **StorePassiveLossRatio** (*float*) –
- **AmbientTempStoreF** (*int*) –
- **TypeName** (*Literal['atn.params.simpleresistivehydronic']*) –
- **Version** (*str*) –

GNodeAlias:

- Description:
- Format: LeftRightDot

HomeCity:

- Description:

TimezoneString:

- Description:

StorageSteps:

- Description:

FloSlices:

- Description:

SliceDurationMinutes:

- Description:

CurrencyUnit:

- Description:

Tariff:

- Description:

EnergyType:

- Description:

StandardOfferPriceDollarsPerMwh:

- Description:

FlatDistributionTariffDollarsPerMwh:

- Description:

StoreSizeGallons:

- Description:

MaxStoreTempF:

- Description:

ElementMaxPowerKw:

- Description:

RequiredSourceWaterTempF:

- Description:

CirculatorPumpGpm:

- Description:

ReturnWaterDeltaTempF:

- Description:

AnnualHvacKwhTh:

- Description:

AmbientPowerInKw:

- Description:

HouseWorstCaseTempF:

- Description:

RoomTempF:

- Description:

StorePassiveLossRatio:

- Description:

AmbientTempStoreF:

- Description:

class gwatn.types.atn_params_simpleresistivehydronic.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**AtnParamsSimpleresistivehydronic_Maker**(*g_node_alias*, *home_city*, *timezone_string*, *storage_steps*, *flo_slices*, *slice_duration_minutes*, *currency_unit*, *tariff*, *energy_type*, *standard_offer_price_dollars_per_mwh*, *flat_distribution_tariff_dollars_per_mwh*, *store_size_gallons*, *max_store_temp_f*, *element_max_power_kw*, *required_source_water_temp_f*, *circulator_pump_gpm*, *return_water_delta_temp_f*, *annual_hvac_kwh_th*, *ambient_power_in_kw*, *house_worst_case_temp_f*, *room_temp_f*, *store_passive_loss_ratio*, *ambient_temp_store_f*)

Parameters

- **g_node_alias** (*str*) –
- **home_city** (*str*) –

- **timezone_string** (*str*) –
- **storage_steps** (*int*) –
- **flo_slices** (*int*) –
- **slice_duration_minutes** (*int*) –
- **currency_unit** (RecognizedCurrencyUnit) –
- **tariff** (*DistributionTariff*) –
- **energy_type** (*EnergySupplyType*) –
- **standard_offer_price_dollars_per_mwh** (*int*) –
- **flat_distribution_tariff_dollars_per_mwh** (*int*) –
- **store_size_gallons** (*int*) –
- **max_store_temp_f** (*int*) –
- **element_max_power_kw** (*float*) –
- **required_source_water_temp_f** (*int*) –
- **circulator_pump_gpm** (*float*) –
- **return_water_delta_temp_f** (*int*) –
- **annual_hvac_kwh_th** (*int*) –
- **ambient_power_in_kw** (*float*) –
- **house_worst_case_temp_f** (*int*) –
- **room_temp_f** (*int*) –
- **store_passive_loss_ratio** (*float*) –
- **ambient_temp_store_f** (*int*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (AtnParamsSimpleresistivehydronic) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
AtnParamsSimpleresistivehydronic

1.7.8 BasegnodeScadaCreate

Python pydantic class corresponding to json type `basegnode.scada.create`.

class gwatn.types.**BasegnodeScadaCreate**(*, *TaAlias*, *ScadaAddr*, *TaDaemonAddr*, *GNodeRegistryAddr*, *SignedProof*, *TypeName='basegnode.scada.create'*, *Version='000'*)

Scada BaseGNode Creation.

This is a payload designed to be sent from a TaOwner to the GNodeFactory. The TaOwner creates a private Algorand key and puts it on the Scada Device that will sense and control their TerminalAsset. The public address is associated with the Scada GNode by the GNodeFactory.

Parameters

- **TaAlias** (*str*) –
- **ScadaAddr** (*str*) –
- **TaDaemonAddr** (*str*) –
- **GNodeRegistryAddr** (*str*) –
- **SignedProof** (*str*) –
- **TypeName** (*Literal['basegnode.scada.create']*) –
- **Version** (*str*) –

classmethod check_axiom_1(*v*)

Axiom 1: TaOwner is SignedProof signer. The TaDaemonAddr provides the public address for the TaOwner. This TaOwnerAddr must match the signature on the SignedProof.

Parameters
v (*dict*) –

Return type
dict

classmethod check_axiom_2(*v*)

Axiom 2: TaAlias matches TaDeed. The TaDaemonAddr owns a TaDeed for the TaAlias.

Parameters
v (*dict*) –

Return type
dict

TaAlias:

- Description: TerminalAsset Alias. GNodeAlias of the TerminalAsset that will be controlled by the new SCADA GNode. The SCADA GNodeAlias will have '.scada' appended to this.
- Format: LeftRightDot

ScadaAddr:

- Description: Algorand address for the SCADA. The TaOwner makes the corresponding private key, puts it on the SCADA device, and then sends this address to the GNodeFactory.
- Format: AlgoAddressStringFormat

TaDaemonAddr:

- Description: Algorand address of the associated TaDaemon. The TaDaemonAddr will have the TaDeed, and can be used to verify the public address of the TaOwner
- Format: AlgoAddressStringFormat

GNodeRegistryAddr:

- Description: GNodeRegistry Algorand address. The GNodeRegistry that contains Make/Model information about the SCADA and TerminalAsset
- Format: AlgoAddressStringFormat

SignedProof:

- Description: Recent transaction signed by the TaOwner. These will be replaced by composite transactions in next gen code.
- Format: AlgoMsgPackEncoded

class gwatn.types.basegnode_scada_create.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.basegnode_scada_create.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.basegnode_scada_create.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**BasegnodeScadaCreate_Maker**(*ta_alias*, *scada_addr*, *ta_daemon_addr*, *g_node_registry_addr*, *signed_proof*)

Parameters

- **ta_alias** (*str*) –
- **scada_addr** (*str*) –
- **ta_daemon_addr** (*str*) –
- **g_node_registry_addr** (*str*) –
- **signed_proof** (*str*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (BasegnodeScadaCreate) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
BasegnodeScadaCreate

1.7.9 DiscoverycertAlgoCreate

Python pydantic class corresponding to json type `discoverycert.algo.create`.

class gwatn.types.**DiscoverycertAlgoCreate**(**, GNodeAlias, Role, OldChildAliasList, DiscovererAddr, SupportingMaterialHash, MicroLat=None, MicroLon=None, TypeName='discoverycert.algo.create', Version='000'*)

Parameters

- **GNodeAlias** (*str*) –
- **Role** (CoreGNodeRole) –
- **OldChildAliasList** (*List[str]*) –
- **DiscovererAddr** (*str*) –
- **SupportingMaterialHash** (*str*) –
- **MicroLat** (*int | None*) –
- **MicroLon** (*int | None*) –
- **TypeName** (*Literal['discoverycert.algo.create']*) –
- **Version** (*str*) –

GNodeAlias:

- Description:
- Format: LeftRightDot

Role:

- Description:

OldChildAliasList:

- Description:
- Format: LeftRightDot

DiscovererAddr:

- Description:

- Format: AlgoAddressStringFormat

SupportingMaterialHash:

- Description:

MicroLat:

- Description:

MicroLon:

- Description:

class gwatn.types.discoverycert_algo_create.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.discoverycert_algo_create.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.**DiscoverycertAlgoCreate_Maker**(*g_node_alias*, *role*, *old_child_alias_list*, *discoverer_addr*, *supporting_material_hash*, *micro_lat*, *micro_lon*)

Parameters

- **g_node_alias** (*str*) –
- **role** (CoreGNodeRole) –
- **old_child_alias_list** (*List[str]*) –
- **discoverer_addr** (*str*) –
- **supporting_material_hash** (*str*) –
- **micro_lat** (*int | None*) –
- **micro_lon** (*int | None*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (DiscoverycertAlgoCreate) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
DiscoverycertAlgoCreate

1.7.10 DispatchContractConfirmed

Python pydantic class corresponding to json type `dispatch.contract.confirmed`.

class gwatn.types.**DispatchContractConfirmed**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *AtnParamsTypeName*, *SignedProof*, *Params*, *TypeName='dispatch.contract.confirmed'*, *Version='000'*)

Message sent from AtomicTNode back to SCADA via Rabbit .

Paired with join.dispatch.contract. Sent from AtomicTNode back to SCADA once the AtomicTNode has successfully finished bootstrapping the Dispatch Contract and opted in. Once it has done this, the Dispatch Contract is ready

to collect audit information about heartbeats, dispatch, energy and power.

https://gridworks.readthedocs.io/en/latest/dispatch-contract.html

Parameters

- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **AtnParamsTypeName** (*str*) –
- **SignedProof** (*str*) –
- **Params** (AtnParams) –
- **TypeName** (*Literal['dispatch.contract.confirmed']*) –
- **Version** (*str*) –

classmethod check_axiom_1(*v*)

Axiom 1: AtnParamsTypeName matches AtnParams. AtnParams must have

Parameters
v (*dict*) –

Return type
dict

FromGNodeAlias:

- Description:
- Format: LeftRightDot

FromGNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

AtnParamsTypeName:

- Description:
- Format: LeftRightDot

SignedProof:

- Description:
- Format: AlgoMsgPackEncoded

Params:

- Description:

class gwatn.types.dispatch_contract_confirmed.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.dispatch_contract_confirmed.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.dispatch_contract_confirmed.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**DispatchContractConfirmed_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *atn_params_type_name*, *signed_proof*, *params*)

Parameters

- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **atn_params_type_name** (*str*) –
- **signed_proof** (*str*) –
- **params** (AtnParams) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (DispatchContractConfirmed) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
DispatchContractConfirmed

1.7.11 FloParams

Python pydantic class corresponding to json type `flo.params`.

class gwatn.types.**FloParams**(*, *GNodeAlias*, *FloParamsUid*, *HomeCity='MILLINOCKET_ME'*, *TimezoneString='US/Eastern'*, *StartYearUtc=2020*, *StartMonthUtc=1*, *StartDayUtc=1*, *StartHourUtc=0*, *StartMinuteUtc=0*, *TypeName='flo.params'*, *Version='000'*, ***extra_data*)

Base class for Forward Looking Optimizer params.

Derived classes are expected to have TypeNames enforced as literals that start with flo.params. E.g. flo.params.brickstorageheater. This container is used for sending messages that include flo.params (i.e, flo.params.report [More info](https://gridworks-atn.readthedocs.io/en/latest/flo.html#flo-params).

Parameters

- **GNodeAlias** (*str*) –
- **FloParamsUid** (*str*) –
- **HomeCity** (*str*) –
- **TimezoneString** (*str*) –
- **StartYearUtc** (*int*) –
- **StartMonthUtc** (*int*) –
- **StartDayUtc** (*int*) –
- **StartHourUtc** (*int*) –
- **StartMinuteUtc** (*int*) –
- **TypeName** (*str*) –
- **Version** (*str*) –
- **extra_data** (*Any*) –

GNodeAlias:

- Description:
- Format: LeftRightDot

FloParamsUid:

- Description:
- Format: UuidCanonicalTextual

HomeCity:

- Description:

TimezoneString:

- Description:

StartYearUtc:

- Description:

StartMonthUtc:

- Description:

StartDayUtc:

- Description:

StartHourUtc:

- Description:

StartMinuteUtc:

- Description:

class gwatn.types.flo_params.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.flo_params.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**FloParams_Maker**(*g_node_alias*, *flo_params_uid*, *home_city*, *timezone_string*, *start_year_utc*, *start_month_utc*, *start_day_utc*, *start_hour_utc*, *start_minute_utc*)

Parameters

- **g_node_alias** (*str*) –
- **flo_params_uid** (*str*) –
- **home_city** (*str*) –
- **timezone_string** (*str*) –
- **start_year_utc** (*int*) –
- **start_month_utc** (*int*) –
- **start_day_utc** (*int*) –
- **start_hour_utc** (*int*) –
- **start_minute_utc** (*int*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (FloParams) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
FloParams

1.7.12 FloParamsBrickstorageheater

Python pydantic class corresponding to json type `flo.params.brickstorageheater`.

class gwatn.types.**FloParamsBrickstorageheater**(**, MaxBrickTempC=190, RatedMaxPowerKw=13.5, ROff=0.08, ROn=0.15, RoomTempF=70, CurrencyUnit=RecognizedCurrencyUnit.USD, TempUnit=RecognizedTemperatureUnit.F, TimezoneString='US/Eastern', HomeCity='MILLINOCKET_ME', IsRegulating=False, StorageSteps=100, SliceDurationMinutes=[60], PowerRequiredByHouseFromSystemAvgKwList=[3.42], C=200, RealtimeElectricityPrice=[10.35], OutsideTempF=[-5.1], DistributionPrice=[40.0], RtElecPriceUid, RegulationPrice=[25.3], WeatherUid, DistPriceUid, RegPriceUid=None, StartYearUtc=2020, StartMonthUtc=1, StartDayUtc=1, StartHourUtc=0, StartMinuteUtc=0, StartingStoreIdx=50, AmbientPowerInKw=1.25, HouseWorstCaseTempF=-7, GNodeAlias, FloParamsUid, TypeName='flo.params.brickstorageheater', Version='000'*)

.

FloParams for the BrickStorageHeater AtomicTNode strategy. [More info](https://gridworks-atn.readthedocs.io/en/latest/brick-storage-heater.html).

Parameters

- **MaxBrickTempC** (*int*) –
- **RatedMaxPowerKw** (*float*) –
- **ROff** (*float*) –
- **ROn** (*float*) –
- **RoomTempF** (*int*) –
- **CurrencyUnit** (RecognizedCurrencyUnit) –
- **TempUnit** (*RecognizedTemperatureUnit*) –

- **TimezoneString** (*str*) –
- **HomeCity** (*str*) –
- **IsRegulating** (*bool*) –
- **StorageSteps** (*int*) –
- **SliceDurationMinutes** (*List[int]*) –
- **PowerRequiredByHouseFromSystemAvgKwList** (*List[float]*) –
- **C** (*float*) –
- **RealtimeElectricityPrice** (*List[float]*) –
- **OutsideTempF** (*List[float]*) –
- **DistributionPrice** (*List[float]*) –
- **RtElecPriceUid** (*str*) –
- **RegulationPrice** (*List[float]*) –
- **WeatherUid** (*str*) –
- **DistPriceUid** (*str*) –
- **RegPriceUid** (*str | None*) –
- **StartYearUtc** (*int*) –
- **StartMonthUtc** (*int*) –
- **StartDayUtc** (*int*) –
- **StartHourUtc** (*int*) –
- **StartMinuteUtc** (*int*) –
- **StartingStoreIdx** (*int*) –
- **AmbientPowerInKw** (*float*) –
- **HouseWorstCaseTempF** (*int*) –
- **GNodeAlias** (*str*) –
- **FloParamsUid** (*str*) –
- **TypeName** (*Literal['flo.params.brickstorageheater']*) –
- **Version** (*str*) –

MaxBrickTempC:

- Description:

RatedMaxPowerKw:

- Description:

ROff:

- Description:

ROn:

- Description:

RoomTempF:

- Description:

CurrencyUnit:

- Description:

TempUnit:

- Description:

TimezoneString:

- Description:

HomeCity:

- Description:

IsRegulating:

- Description:

StorageSteps:

- Description:

SliceDurationMinutes:

- Description:

PowerRequiredByHouseFromSystemAvgKwList:

- Description:

C:

- Description:

RealtimeElectricityPrice:

- Description:

OutsideTempF:

- Description:

DistributionPrice:

- Description:

RtElecPriceUid:

- Description:
- Format: UuidCanonicalTextual

RegulationPrice:

- Description:

WeatherUid:

- Description:
- Format: UuidCanonicalTextual

DistPriceUid:

- Description:
- Format: UuidCanonicalTextual

RegPriceUid:

- Description:
- Format: UuidCanonicalTextual

StartYearUtc:

- Description:

StartMonthUtc:

- Description:

StartDayUtc:

- Description:

StartHourUtc:

- Description:

StartMinuteUtc:

- Description:

StartingStoreIdx:

- Description:

AmbientPowerInKw:

- Description:

HouseWorstCaseTempF:

- Description:

GNodeAlias:

- Description:
- Format: LeftRightDot

FloParamsUid:

- Description:
- Format: UuidCanonicalTextual

class gwatn.types.flo_params_brickstorageheater.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.flo_params_brickstorageheater.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**FloParamsBrickstorageheater_Maker**(*max_brick_temp_c*, *rated_max_power_kw*, *r_off*, *r_on*, *room_temp_f*, *currency_unit*, *temp_unit*, *timezone_string*, *home_city*, *is_regulating*, *storage_steps*, *slice_duration_minutes*, *power_required_by_house_from_system_avg_kw_list*, *c*, *realtime_electricity_price*, *outside_temp_f*, *distribution_price*, *rt_elec_price_uid*, *regulation_price*, *weather_uid*, *dist_price_uid*, *reg_price_uid*, *start_year_utc*, *start_month_utc*, *start_day_utc*, *start_hour_utc*, *start_minute_utc*, *starting_store_idx*, *ambient_power_in_kw*, *house_worst_case_temp_f*, *g_node_alias*, *flo_params_uid*)

Parameters

- **max_brick_temp_c** (*int*) –
- **rated_max_power_kw** (*float*) –
- **r_off** (*float*) –
- **r_on** (*float*) –
- **room_temp_f** (*int*) –
- **currency_unit** (RecognizedCurrencyUnit) –
- **temp_unit** (*RecognizedTemperatureUnit*) –
- **timezone_string** (*str*) –
- **home_city** (*str*) –
- **is_regulating** (*bool*) –
- **storage_steps** (*int*) –
- **slice_duration_minutes** (*List[int]*) –
- **power_required_by_house_from_system_avg_kw_list** (*List[float]*) –
- **c** (*float*) –
- **realtime_electricity_price** (*List[float]*) –
- **outside_temp_f** (*List[float]*) –
- **distribution_price** (*List[float]*) –
- **rt_elec_price_uid** (*str*) –
- **regulation_price** (*List[float]*) –
- **weather_uid** (*str*) –
- **dist_price_uid** (*str*) –
- **reg_price_uid** (*str | None*) –
- **start_year_utc** (*int*) –
- **start_month_utc** (*int*) –
- **start_day_utc** (*int*) –
- **start_hour_utc** (*int*) –

- **start_minute_utc** (*int*) –
- **starting_store_idx** (*int*) –
- **ambient_power_in_kw** (*float*) –
- **house_worst_case_temp_f** (*int*) –
- **g_node_alias** (*str*) –
- **flo_params_uid** (*str*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (FloParamsBrickstorageheater) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
FloParamsBrickstorageheater

1.7.13 FloParamsReport

Python pydantic class corresponding to json type `flo.params.report`.

class gwatn.types.**FloParamsReport**(*, *GNodeAlias*, *GNodeInstanceId*, *FloParamsTypeName*, *FloParamsTypeVersion*, *ReportGeneratedTimeUnixS*, *IrlTimeUnixS=None*, *Params*, *TypeName='flo.params.report'*, *Version='000'*)

Flo Params Report.

Type used for a message provided by an AtomicTNode or SCADA actor re the flo parameters just used to run a FLO. [More info](https://gridworks-atn.readthedocs.io/en/latest/flo.html#flo-params).

Parameters

- **GNodeAlias** (*str*) –
- **GNodeInstanceId** (*str*) –
- **FloParamsTypeName** (*str*) –
- **FloParamsTypeVersion** (*str*) –
- **ReportGeneratedTimeUnixS** (*int*) –
- **IrlTimeUnixS** (*int | None*) –
- **Params** (FloParams) –
- **TypeName** (*Literal['flo.params.report']*) –
- **Version** (*str*) –

GNodeAlias:

- Description:
- Format: LeftRightDot

GNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

FloParamsTypeName:

- Description:
- Format: LeftRightDot

FloParamsTypeVersion:

- Description:

ReportGeneratedTimeUnixS:

- Description:
- Format: ReasonableUnixTimeS

IrlTimeUnixS:

- Description:
- Format: ReasonableUnixTimeS

Params:

- Description:

class gwatn.types.flo_params_report.**check_is_reasonable_unix_time_s**(*v*)

ReasonableUnixTimeS format: time in unix seconds between Jan 1 2000 and Jan 1 3000

Raises
ValueError – if not ReasonableUnixTimeS format

Parameters
v (*int*) –

class gwatn.types.flo_params_report.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.flo_params_report.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**FloParamsReport_Maker**(*g_node_alias*, *g_node_instance_id*, *flo_params_type_name*, *flo_params_type_version*, *report_generated_time_unix_s*, *irl_time_unix_s*, *params*)

Parameters

- **g_node_alias** (*str*) –
- **g_node_instance_id** (*str*) –
- **flo_params_type_name** (*str*) –
- **flo_params_type_version** (*str*) –
- **report_generated_time_unix_s** (*int*) –
- **irl_time_unix_s** (*int | None*) –
- **params** (FloParams) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (FloParamsReport) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
FloParamsReport

1.7.14 FloParamsSimpleresistivehydronic

Python pydantic class corresponding to json type `flo.params.simpleresistivehydronic`.

class gwatn.types.**FloParamsSimpleresistivehydronic**(*, *GNodeAlias*, *FloParamsUid*, *HomeCity='MILLINOCKET_ME'*, *TimezoneString='US/Eastern'*, *StartYearUtc=2020*, *StartMonthUtc=1*, *StartDayUtc=1*, *StartHourUtc=0*, *StartMinuteUtc=0*, *StorageSteps=100*, *StoreSizeGallons=240*, *MaxStoreTempF=210*, *RatedPowerKw=9.5*, *RequiredSourceWaterTempF=120*, *CirculatorPumpGpm=4.5*, *ReturnWaterDeltaTempF=20*, *RoomTempF=70*, *AmbientPowerInKw=1.2*, *HouseWorstCaseTempF=-7*, *StorePassiveLossRatio=0.005*, *PowerLostFromHouseKwList=[3.42]*, *AmbientTempStoreF=65*, *SliceDurationMinutes=[60]*, *RealtimeElectricityPrice=[10.35]*, *DistributionPrice=[40.0]*, *OutsideTempF=[-5.1]*, *RtElecPriceUid*, *DistPriceUid*, *WeatherUid*, *CurrencyUnit=RecognizedCurrencyUnit.USD*, *Tariff=DistributionTariff.VersantA1StorageHeatTariff*, *EnergyType=EnergySupplyType.RealtimeLocalLmp*, *StandardOfferPriceDollarsPerMwh=110*, *FlatDistributionTariffDollarsPerMwh=113*, *StartingStoreIdx=50*, *TypeName='flo.params.simpleresistivehydronic'*, *Version='000'*)

Parameters

- **GNodeAlias** (*str*) –
- **FloParamsUid** (*str*) –
- **HomeCity** (*str*) –
- **TimezoneString** (*str*) –
- **StartYearUtc** (*int*) –
- **StartMonthUtc** (*int*) –
- **StartDayUtc** (*int*) –
- **StartHourUtc** (*int*) –
- **StartMinuteUtc** (*int*) –
- **StorageSteps** (*int*) –
- **StoreSizeGallons** (*int*) –
- **MaxStoreTempF** (*int*) –

- **RatedPowerKw** (*float*) –
- **RequiredSourceWaterTempF** (*int*) –
- **CirculatorPumpGpm** (*float*) –
- **ReturnWaterDeltaTempF** (*int*) –
- **RoomTempF** (*int*) –
- **AmbientPowerInKw** (*float*) –
- **HouseWorstCaseTempF** (*float*) –
- **StorePassiveLossRatio** (*float*) –
- **PowerLostFromHouseKwList** (*List[float]*) –
- **AmbientTempStoreF** (*int*) –
- **SliceDurationMinutes** (*List[int]*) –
- **RealtimeElectricityPrice** (*List[float]*) –
- **DistributionPrice** (*List[float]*) –
- **OutsideTempF** (*List[float]*) –
- **RtElecPriceUid** (*str*) –
- **DistPriceUid** (*str*) –
- **WeatherUid** (*str*) –
- **CurrencyUnit** (RecognizedCurrencyUnit) –
- **Tariff** (*DistributionTariff*) –
- **EnergyType** (*EnergySupplyType*) –
- **StandardOfferPriceDollarsPerMwh** (*int*) –
- **FlatDistributionTariffDollarsPerMwh** (*int*) –
- **StartingStoreIdx** (*int*) –
- **TypeName** (*Literal['flo.params.simpleresistivehydronic']*) –
- **Version** (*str*) –

GNodeAlias:

- Description:
- Format: LeftRightDot

FloParamsUid:

- Description:
- Format: UuidCanonicalTextual

HomeCity:

- Description:

TimezoneString:

- Description:

StartYearUtc:

- Description:

StartMonthUtc:

- Description:

StartDayUtc:

- Description:

StartHourUtc:

- Description:

StartMinuteUtc:

- Description:

StorageSteps:

- Description:

StoreSizeGallons:

- Description:

MaxStoreTempF:

- Description:

RatedPowerKw:

- Description:

RequiredSourceWaterTempF:

- Description:

CirculatorPumpGpm:

- Description:

ReturnWaterDeltaTempF:

- Description:

RoomTempF:

- Description:

AmbientPowerInKw:

- Description:

HouseWorstCaseTempF:

- Description:

StorePassiveLossRatio:

- Description:

PowerLostFromHouseKwList:

- Description:

AmbientTempStoreF:

- Description:

SliceDurationMinutes:

- Description:

RealtimeElectricityPrice:

- Description:

DistributionPrice:

- Description:

OutsideTempF:

- Description:

RtElecPriceUid:

- Description:
- Format: UuidCanonicalTextual

DistPriceUid:

- Description:
- Format: UuidCanonicalTextual

WeatherUid:

- Description:
- Format: UuidCanonicalTextual

CurrencyUnit:

- Description:

Tariff:

- Description:

EnergyType:

- Description:

StandardOfferPriceDollarsPerMwh:

- Description:

FlatDistributionTariffDollarsPerMwh:

- Description:

StartingStoreIdx:

- Description:

class gwatn.types.flo_params_simpleresistivehydronic.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.flo_params_simpleresistivehydronic.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**FloParamsSimpleresistivehydronic_Maker**(*g_node_alias*, *flo_params_uid*, *home_city*, *timezone_string*, *start_year_utc*, *start_month_utc*, *start_day_utc*, *start_hour_utc*, *start_minute_utc*, *storage_steps*, *store_size_gallons*, *max_store_temp_f*, *rated_power_kw*, *required_source_water_temp_f*, *circulator_pump_gpm*, *return_water_delta_temp_f*, *room_temp_f*, *ambient_power_in_kw*, *house_worst_case_temp_f*, *store_passive_loss_ratio*, *power_lost_from_house_kw_list*, *ambient_temp_store_f*, *slice_duration_minutes*, *realtime_electricity_price*, *distribution_price*, *outside_temp_f*, *rt_elec_price_uid*, *dist_price_uid*, *weather_uid*, *currency_unit*, *tariff*, *energy_type*, *standard_offer_price_dollars_per_mwh*, *flat_distribution_tariff_dollars_per_mwh*, *starting_store_idx*)

Parameters

- **g_node_alias** (*str*) –
- **flo_params_uid** (*str*) –
- **home_city** (*str*) –
- **timezone_string** (*str*) –
- **start_year_utc** (*int*) –
- **start_month_utc** (*int*) –
- **start_day_utc** (*int*) –
- **start_hour_utc** (*int*) –
- **start_minute_utc** (*int*) –
- **storage_steps** (*int*) –
- **store_size_gallons** (*int*) –
- **max_store_temp_f** (*int*) –
- **rated_power_kw** (*float*) –
- **required_source_water_temp_f** (*int*) –
- **circulator_pump_gpm** (*float*) –
- **return_water_delta_temp_f** (*int*) –

- **room_temp_f** (*int*) –
- **ambient_power_in_kw** (*float*) –
- **house_worst_case_temp_f** (*float*) –
- **store_passive_loss_ratio** (*float*) –
- **power_lost_from_house_kw_list** (*List[float]*) –
- **ambient_temp_store_f** (*int*) –
- **slice_duration_minutes** (*List[int]*) –
- **realtime_electricity_price** (*List[float]*) –
- **distribution_price** (*List[float]*) –
- **outside_temp_f** (*List[float]*) –
- **rt_elec_price_uid** (*str*) –
- **dist_price_uid** (*str*) –
- **weather_uid** (*str*) –
- **currency_unit** (RecognizedCurrencyUnit) –
- **tariff** (*DistributionTariff*) –
- **energy_type** (*EnergySupplyType*) –
- **standard_offer_price_dollars_per_mwh** (*int*) –
- **flat_distribution_tariff_dollars_per_mwh** (*int*) –
- **starting_store_idx** (*int*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (FloParamsSimpleresistivehydronic) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
FloParamsSimpleresistivehydronic

1.7.15 InitialTadeedAlgoCreate

Python pydantic class corresponding to json type `initial.tadeed.algo.create`.

class gwatn.types.**InitialTadeedAlgoCreate**(*, *ValidatorAddr*, *HalfSignedDeedCreationMtx*, *TypeName='initial.tadeed.algo.create'*, *Version='000'*)

TaValidator sends to GNodeFactory to complete creation of an initial TaDeed.

If this message is valid, the GNodeFactory co-signs and submits the TaDeed creation. In addition, the GnodeFactory creates a TerminalAsset with GNodeStatus pending. For more information: [TaDeed](https://gridworks.readthedocs.io/en/latest/ta-deed.html) [TaValidator](https://gridworks.readthedocs.io/en/latest/ta-validator.html)

Parameters

- **ValidatorAddr** (*str*) –
- **HalfSignedDeedCreationMtx** (*str*) –
- **TypeName** (*Literal['initial.tadeed.algo.create']*) –
- **Version** (*str*) –

classmethod **check_axiom_1**(*v*)

Axiom 1: Is correct Multisig. Decoded HalfSignedDeedCreationMtx must have type MultisigTransaction from the 2-sig MultiAccount [GnfAdminAddr, ValidatorAddr]. [More info](https://gridworks.readthedocs.io/en/latest/g-node-factory.html#gnfadminaddr)

Parameters
v (*dict*) –

Return type
dict

classmethod **check_axiom_2**(*v*)

Axiom 2: Creates Initial ASA TaDeed. The transaction must create an Algorand Standard Asset

- Total is 1
- UnitName is TADEED
- Manager is GnfAdminAddr
- **AssetName has the following characteristics:**
  - length <= 32 characters
  - LeftRightDot format
  - final word is '.ta'

[More info](https://gridworks.readthedocs.io/en/latest/ta-deed.html#asa-tadeed-specs)

Parameters
v (*dict*) –

Return type
dict

classmethod **check_axiom_3**(*v*)

Axiom 3: Mtx signed by TaValidator.

Parameters
v (*dict*) –

Return type
dict

ValidatorAddr:

- Description: Address of the TaValidator. The Algorand address of the TaValidator who is going to validate the location, device type, and power metering of the TerminalAsset.
- Format: AlgoAddressStringFormat

HalfSignedDeedCreationMtx:

- Description: Algo mulit-transaction for TaDeed creation
- Format: AlgoMsgPackEncoded

class gwatn.types.initial_tadeed_algo_create.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.initial_tadeed_algo_create.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**InitialTadeedAlgoCreate_Maker**(*validator_addr*, *half_signed_deed_creation_mtx*)

Parameters

- **validator_addr** (*str*) –
- **half_signed_deed_creation_mtx** (*str*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (InitialTadeedAlgoCreate) –

Return type
str

classmethod **type_to_tuple**(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
InitialTadeedAlgoCreate

1.7.16 InitialTadeedAlgoOptin

Python pydantic class corresponding to json type `initial.tadeed.algo.optin`.

class gwatn.types.**InitialTadeedAlgoOptin**(*, *TerminalAssetAlias*, *TaOwnerAddr*, *ValidatorAddr*, *SignedInitialDaemonFundingTxn*, *TypeName='initial.tadeed.algo.optin'*, *Version='002'*)

Received by TaDaemon so that it can opt into intial TaDeed.

The TaDaemon must opt into the TaDeed before receiving it. This message prompts that action.

Parameters

- **TerminalAssetAlias** (*str*) –
- **TaOwnerAddr** (*str*) –
- **ValidatorAddr** (*str*) –
- **SignedInitialDaemonFundingTxn** (*str*) –
- **TypeName** (*Literal['initial.tadeed.algo.optin']*) –
- **Version** (*str*) –

classmethod **check_axiom_1**(*v*)

Axiom 1: Is correct Multisig. Decoded SignedInitialDaemonFundingTxn must be a SignedTransaction signed by TaOwnerAddr.

Parameters
v (*dict*) –

Return type
dict

classmethod **check_axiom_2**(*v*)

Axiom 2: TaDeed consistency. There is an ASA TaDeed created by and owned by the 2-sig MultiAccount [GnfAdminAddr, ValidatorAddr], where the TaDeed's AssetName is equal to the payload's TerminalAssetAlias.

Parameters
v (*dict*) –

Return type
dict

TerminalAssetAlias:

- Description: The GNodeAlias of the TerminalAsset
- Format: LeftRightDot

TaOwnerAddr:

- Description: The Algorand address of the owner for the TerminalAsset
- Format: AlgoAddressStringFormat

ValidatorAddr:

- Description: Address of the TaValidator. The Algorand address of the TaValidator who has validated the location, device type, and power metering of the TerminalAsset.
- Format: AlgoAddressStringFormat

SignedInitialDaemonFundingTxn:

- Description: . Funding transaction for the TaDaemon account, signed by the TaOwner.
- Format: AlgoMsgPackEncoded

class gwatn.types.initial_tadeed_algo_optin.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.initial_tadeed_algo_optin.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.initial_tadeed_algo_optin.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**InitialTadeedAlgoOptin_Maker**(*terminal_asset_alias*, *ta_owner_addr*, *validator_addr*, *signed_initial_daemon_funding_txn*)

Parameters

- **terminal_asset_alias** (*str*) –
- **ta_owner_addr** (*str*) –
- **validator_addr** (*str*) –
- **signed_initial_daemon_funding_txn** (*str*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (InitialTadeedAlgoOptin) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
InitialTadeedAlgoOptin

1.7.17 InitialTadeedAlgoTransfer

Python pydantic class corresponding to json type `initial.tadeed.algo.transfer`.

class gwatn.types.**InitialTadeedAlgoTransfer**(*, *MicroLat*, *MicroLon*, *ValidatorAddr*, *TaDaemonAddr*, *TaOwnerAddr*, *FirstDeedTransferMtx*, *TypeName='initial.tadeed.algo.transfer'*, *Version='000'*)

TaValidator sends to GNodeFactory after validating Transactive Device.

Once the TaValidator has done the initial on-site inspection of the Transactive Device including its location and the type and quality of its power and energy metering, the TaValidator lets the GNodeFactory know by sending this message. Note the message also includes the lat/lon of the Transactive Device. On receiving and validating this message, the GNodeFactory will co-sign the transfer and send the TaDeed to the TaDaemon address. In addition, the GNodeFactory creates and sends a TaTradingRights certificate to the TaDaemon address. Only once the GNodeFactory has verified that the TaDaemon address owns the TaDeed and TaTradingRights will it change the GNodeStatus of the associated TerminalAsset from Pending to Active. [GNodeStatus](https://gridworks.readthedocs.io/en/latest/g-node-status.html) [TaDeed](https://gridworks.readthedocs.io/en/latest/ta-deed.html) [TaTradingRights](https://gridworks.readthedocs.io/en/latest/ta-trading-rights.html) [TaValidator](https://gridworks.readthedocs.io/en/latest/ta-validator.html) [TerminalAsset](https://gridworks.readthedocs.io/en/latest/terminal-asset.html) [Transactive Device](https://gridworks.readthedocs.io/en/latest/transactive-device.html)

Parameters

- **MicroLat** (*int*) –
- **MicroLon** (*int*) –
- **ValidatorAddr** (*str*) –
- **TaDaemonAddr** (*str*) –
- **TaOwnerAddr** (*str*) –
- **FirstDeedTransferMtx** (*str*) –
- **TypeName** (*Literal['initial.tadeed.algo.transfer']*) –
- **Version** (*str*) –

classmethod **check_axiom_1**(*v*)

Axiom 1: Is correct Multisig. Decoded FirstDeedTransferMtx must have type MultisigTransaction from the 2-sig MultiAccount [GnfAdminAddr, ValidatorAddr]. [More info](https://gridworks.readthedocs.io/en/latest/g-node-factory.html#gnfadminaddr)

Parameters
v (*dict*) –

Return type
dict

classmethod **check_axiom_2**(*v*)

Axiom 2: TaDaemon funded by TaOwner. The TaDaemonAddr was created with funding from the TaOwnerAddr, and has sufficient funding according to the GNodeFactory.

Parameters
v (*dict*) –

Return type
dict

MicroLat:

- Description: . The Latitude of the Transactive Device, times 10^6

MicroLon:

- Description: . The Longitude of the Transactive Device, times 10^6

ValidatorAddr:

- **Description: . The Algoand address for the TaValidator who validated the location,** metering and type of the Transactive Device.
- Format: AlgoAddressStringFormat

TaDaemonAddr:

- **Description: . The Algorand address for the TaDaemon which will own the TaDeed**

  and initially the TaTradingRights), as well as holding funds on

  behalf of the TaOwner.
- Format: AlgoAddressStringFormat

TaOwnerAddr:

- **Description: . The Algorand address of the entity owning the Transactive Device, and** thus also the TerminalAsset
- Format: AlgoAddressStringFormat

FirstDeedTransferMtx:

- **Description: . The half-signed multi transaction for transferring the TaDeed to the** TaDaemon.
- Format: AlgoMsgPackEncoded

class gwatn.types.initial_tadeed_algo_transfer.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.initial_tadeed_algo_transfer.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**InitialTadeedAlgoTransfer_Maker**(*micro_lat*, *micro_lon*, *validator_addr*, *ta_daemon_addr*, *ta_owner_addr*, *first_deed_transfer_mtx*)

Parameters

- **micro_lat** (*int*) –
- **micro_lon** (*int*) –
- **validator_addr** (*str*) –

- **ta_daemon_addr** (*str*) –
- **ta_owner_addr** (*str*) –
- **first_deed_transfer_mtx** (*str*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (InitialTadeedAlgoTransfer) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
InitialTadeedAlgoTransfer

1.7.18 JoinDispatchContract

Python pydantic class corresponding to json type `join.dispatch.contract`.

class gwatn.types.**JoinDispatchContract**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *DispatchContractAppId*, *SignedProof*, *TypeName='join.dispatch.contract'*, *Version='000'*)

Sent from a Scada to its paired AtomicTNode on RabbitMQ.

This is sent as an invitation to join the DispatchContract. Upon receipt of this, the AtomicTNode can check that the DispatchContract has finished the first part of its bootstrapping. This means it is well-funded, and also has the Scada Cert Id and the Scada Addr publicly available. The AtomicTNode can check these against the signature provided by the SCADA in its invitation. An AtomicTNode actor accepts the invitation by finishing the Dispatch Contract bootstrap (which it can only do if its Algorand Account holds the associated TaTradingRights certificate) and then responding to the SCADA via RabbitMQ with a dispatch.contract.confirmed payload.

https://gridworks.readthedocs.io/en/latest/dispatch-contract.html

Parameters

- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **DispatchContractAppId** (*int*) –
- **SignedProof** (*str*) –
- **TypeName** (*Literal['join.dispatch.contract']*) –
- **Version** (*str*) –

classmethod check_axiom_0(*v*)

Axiom 0: ScadaCert matches FromGNodeAlias. The name in the ScadaCert should be the GNodeAlias of the TerminalAsset corresponding to the sending SCADA. Therefore, FromGNodeAlias should be equal to the name of the ScadaCert ASA with *.scada* appended.

Parameters
v (*dict*) –

Return type
dict

FromGNodeAlias:

- Description:
- Format: LeftRightDot

FromGNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

DispatchContractAppId:

- Description:

SignedProof:

- Description:
- Format: AlgoMsgPackEncoded

class gwatn.types.join_dispatch_contract.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.join_dispatch_contract.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.join_dispatch_contract.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**JoinDispatchContract_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *dispatch_contract_app_id*, *signed_proof*)

Parameters

- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **dispatch_contract_app_id** (*int*) –

- **signed_proof** (*str*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (JoinDispatchContract) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
JoinDispatchContract

1.7.19 LatestPrice

Python pydantic class corresponding to json type `latest.price`.

class gwatn.types.**LatestPrice**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *PriceTimes1000*, *PriceUnit*, *MarketSlotName*, *IrlTimeUtc=None*, *MessageId=None*, *TypeName='latest.price'*, *Version='000'*)

Latest Price for a MarketType, sent by a MarketMaker.

The price of the current MarketSlot [More info](https://gridworks.readthedocs.io/en/latest/market-slot.html).

Parameters

- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **PriceTimes1000** (*int*) –
- **PriceUnit** (MarketPriceUnit) –
- **MarketSlotName** (*str*) –
- **IrlTimeUtc** (*str | None*) –
- **MessageId** (*str | None*) –
- **TypeName** (*Literal['latest.price']*) –
- **Version** (*str*) –

FromGNodeAlias:

- Description:
- Format: LeftRightDot

FromGNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

PriceTimes1000:

- Description:

PriceUnit:

- Description:

MarketSlotName:

- Description:
- Format: MarketSlotNameLrdFormat

IrlTimeUtc:

- Description:
- Format: IsoFormat

MessageId:

- Description:
- Format: UuidCanonicalTextual

class gwatn.types.latest_price.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.latest_price.**check_is_iso_format**(*v*)

Parameters
v (*str*) –

class gwatn.types.latest_price.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.latest_price.**check_is_market_slot_name_lrd_format**(*v*)

MaketSlotNameLrdFormat: the format of a MarketSlotName.

- The first word must be a MarketTypeName
- The last word (unix time of market slot start) must

be a 10-digit integer divisible by 300 (i.e. all MarketSlots start at the top of 5 minutes) - More strictly, the last word must be the start of a MarketSlot for that MarketType (i.e. divisible by 3600 for hourly markets) - The middle words have LeftRightDot format (GNodeAlias of the MarketMaker)

Example: rt60gate5.d1.isone.ver.keene.1673539200

Parameters
v (*str*) –

class gwatn.types.**LatestPrice_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *price_times1000*, *price_unit*, *market_slot_name*, *irl_time_utc*, *message_id*)

Parameters

- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **price_times1000** (*int*) –
- **price_unit** (MarketPriceUnit) –
- **market_slot_name** (*str*) –
- **irl_time_utc** (*str | None*) –
- **message_id** (*str | None*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (LatestPrice) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
LatestPrice

1.7.20 MarketSlot

Python pydantic class corresponding to json type `market.slot`.

class gwatn.types.**MarketSlot**(***, *Type*, *MarketMakerAlias*, *StartUnixS*, *TypeName='market.slot'*, *Version='000'*)

[More info](https://gridworks.readthedocs.io/en/latest/market-slot.html).

Parameters

- **Type** (MarketTypeGt) –
- **MarketMakerAlias** (*str*) –
- **StartUnixS** (*int*) –
- **TypeName** (*Literal['market.slot']*) –
- **Version** (*str*) –

Type:

- Description:

MarketMakerAlias:

- Description:

- Format: LeftRightDot

StartUnixS:

- Description:
- Format: ReasonableUnixTimeS

class gwatn.types.market_slot.**check_is_reasonable_unix_time_s**(*v*)

ReasonableUnixTimeS format: time in unix seconds between Jan 1 2000 and Jan 1 3000

Raises
ValueError – if not ReasonableUnixTimeS format

Parameters
v (*int*) –

class gwatn.types.market_slot.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**MarketSlot_Maker**(*type*, *market_maker_alias*, *start_unix_s*)

Parameters

- **type** (MarketTypeGt) –
- **market_maker_alias** (*str*) –
- **start_unix_s** (*int*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (MarketSlot) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
MarketSlot

1.7.21 MarketTypeGt

Python pydantic class corresponding to json type `market.type.gt`.

class gwatn.types.**MarketTypeGt**(*, *Name*, *DurationMinutes*, *GateClosingSeconds*, *PriceUnit*, *QuantityUnit*, *CurrencyUnit*, *PriceMax*, *TypeName='market.type.gt'*, *Version='000'*)

Used by MarketMakers to simultaneously run several different types of Markets.

A [MarketMaker](https://gridworks.readthedocs.io/en/latest/market-maker.html) GNode can run several types of Markets. For example, it can run an

hourly real-time market and also an ancillary services market for Regulation. This is captured by the concept of MarketType.

[More info](https://gridworks.readthedocs.io/en/latest/market-type.html).

Parameters

- **Name** (MarketTypeName) –
- **DurationMinutes** (*int*) –
- **GateClosingSeconds** (*int*) –
- **PriceUnit** (MarketPriceUnit) –
- **QuantityUnit** (MarketQuantityUnit) –
- **CurrencyUnit** (RecognizedCurrencyUnit) –
- **PriceMax** (*int*) –
- **TypeName** (*Literal['market.type.gt']*) –
- **Version** (*str*) –

Name:

- Description: Name of the MarketType

DurationMinutes:

- Description: Duration of MarketSlots, in minutes

GateClosingSeconds:

- Description: Seconds before the start of a MarketSlot after which bids are not accepted

PriceUnit:

- Description: Price Unit for market (e.g. USD Per MWh)

QuantityUnit:

- Description: Quantity Unit for market (e.g. AvgMW)

CurrencyUnit:

- Description: Currency Unit for market (e.g. USD)

PriceMax:

- Description: PMax, required for defining bids

class gwatn.types.**MarketTypeGt_Maker**(*name*, *duration_minutes*, *gate_closing_seconds*, *price_unit*, *quantity_unit*, *currency_unit*, *price_max*)

Parameters

- **name** (MarketTypeName) –
- **duration_minutes** (*int*) –
- **gate_closing_seconds** (*int*) –
- **price_unit** (MarketPriceUnit) –
- **quantity_unit** (MarketQuantityUnit) –
- **currency_unit** (RecognizedCurrencyUnit) –
- **price_max** (*int*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (MarketTypeGt) –

Return type
str

classmethod **type_to_tuple**(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
MarketTypeGt

1.7.22 NewTadeedAlgoOptin

Python pydantic class corresponding to json type `new.tadeed.algo.optin`.

class gwatn.types.**NewTadeedAlgoOptin**(*, *NewTaDeedIdx*, *OldTaDeedIdx*, *TaDaemonAddr*, *ValidatorAddr*, *SignedTaDeedCreationTxn*, *TypeName='new.tadeed.algo.optin'*, *Version='000'*)

Parameters

- **NewTaDeedIdx** (*int*) –
- **OldTaDeedIdx** (*int*) –
- **TaDaemonAddr** (*str*) –
- **ValidatorAddr** (*str*) –
- **SignedTaDeedCreationTxn** (*str*) –
- **TypeName** (*Literal['new.tadeed.algo.optin']*) –
- **Version** (*str*) –

NewTaDeedIdx:

- Description:

OldTaDeedIdx:

- Description:

TaDaemonAddr:

- Description:
- Format: AlgoAddressStringFormat

ValidatorAddr:

- Description:
- Format: AlgoAddressStringFormat

SignedTaDeedCreationTxn:

- Description:
- Format: AlgoMsgPackEncoded

class gwatn.types.new_tadeed_algo_optin.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.new_tadeed_algo_optin.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**NewTadeedAlgoOptin_Maker**(*new_ta_deed_idx*, *old_ta_deed_idx*, *ta_daemon_addr*, *validator_addr*, *signed_ta_deed_creation_txn*)

Parameters

- **new_ta_deed_idx** (*int*) –
- **old_ta_deed_idx** (*int*) –
- **ta_daemon_addr** (*str*) –
- **validator_addr** (*str*) –
- **signed_ta_deed_creation_txn** (*str*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (NewTadeedAlgoOptin) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
NewTadeedAlgoOptin

1.7.23 NewTadeedSend

Python pydantic class corresponding to json type `new.tadeed.send`.

class gwatn.types.**NewTadeedSend**(**, NewTaDeedIdx, OldTaDeedIdx, TaDaemonAddr, ValidatorAddr, SignedTadeedOptinTxn, TypeName='new.tadeed.send', Version='000'*)

Parameters

- **NewTaDeedIdx** (*int*) –
- **OldTaDeedIdx** (*int*) –
- **TaDaemonAddr** (*str*) –
- **ValidatorAddr** (*str*) –
- **SignedTadeedOptinTxn** (*str*) –
- **TypeName** (*Literal['new.tadeed.send']*) –
- **Version** (*str*) –

NewTaDeedIdx:

- Description:

OldTaDeedIdx:

- Description:

TaDaemonAddr:

- Description:
- Format: AlgoAddressStringFormat

ValidatorAddr:

- Description:
- Format: AlgoAddressStringFormat

SignedTadeedOptinTxn:

- Description:
- Format: AlgoMsgPackEncoded

class gwatn.types.new_tadeed_send.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.new_tadeed_send.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**NewTadeedSend_Maker**(*new_ta_deed_idx*, *old_ta_deed_idx*, *ta_daemon_addr*, *validator_addr*, *signed_tadeed_optin_txn*)

Parameters

- **new_ta_deed_idx** (*int*) –
- **old_ta_deed_idx** (*int*) –
- **ta_daemon_addr** (*str*) –
- **validator_addr** (*str*) –
- **signed_tadeed_optin_txn** (*str*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (NewTadeedSend) –

Return type
str

classmethod **type_to_tuple**(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
NewTadeedSend

1.7.24 OldTadeedAlgoReturn

Python pydantic class corresponding to json type `old.tadeed.algo.return`.

class gwatn.types.**OldTadeedAlgoReturn**(*, *OldTaDeedIdx*, *TaDaemonAddr*, *ValidatorAddr*, *SignedNewDeedTransferTxn*, *TypeName='old.tadeed.algo.return'*, *Version='000'*)

Parameters

- **OldTaDeedIdx** (*int*) –
- **TaDaemonAddr** (*str*) –
- **ValidatorAddr** (*str*) –
- **SignedNewDeedTransferTxn** (*str*) –

- **TypeName** (*Literal['old.tadeed.algo.return']*) –
- **Version** (*str*) –

OldTaDeedIdx:

- Description:

TaDaemonAddr:

- Description:
- Format: AlgoAddressStringFormat

ValidatorAddr:

- Description:
- Format: AlgoAddressStringFormat

SignedNewDeedTransferTxn:

- Description:
- Format: AlgoMsgPackEncoded

class gwatn.types.old_tadeed_algo_return.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.old_tadeed_algo_return.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**OldTadeedAlgoReturn_Maker**(*old_ta_deed_idx*, *ta_daemon_addr*, *validator_addr*, *signed_new_deed_transfer_txn*)

Parameters

- **old_ta_deed_idx** (*int*) –
- **ta_daemon_addr** (*str*) –
- **validator_addr** (*str*) –
- **signed_new_deed_transfer_txn** (*str*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (OldTadeedAlgoReturn) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
OldTadeedAlgoReturn

1.7.25 PriceQuantity

Python pydantic class corresponding to json type `price.quantity`.

class gwatn.types.**PriceQuantity**(*, *PriceTimes1000*, *QuantityTimes1000*, *PriceUnit*, *QuantityUnit*, *InjectionIsPositive*, *TypeName='price.quantity'*, *Version='000'*)

Parameters

- **PriceTimes1000** (*int*) –
- **QuantityTimes1000** (*int*) –
- **PriceUnit** (MarketPriceUnit) –
- **QuantityUnit** (MarketQuantityUnit) –
- **InjectionIsPositive** (*bool*) –
- **TypeName** (*Literal['price.quantity']*) –
- **Version** (*str*) –

PriceTimes1000:

- Description:

QuantityTimes1000:

- Description:

PriceUnit:

- Description:

QuantityUnit:

- Description:

InjectionIsPositive:

- Description:

class gwatn.types.**PriceQuantity_Maker**(*price_times1000*, *quantity_times1000*, *price_unit*, *quantity_unit*, *injection_is_positive*)

Parameters

- **price_times1000** (*int*) –
- **quantity_times1000** (*int*) –
- **price_unit** (MarketPriceUnit) –
- **quantity_unit** (MarketQuantityUnit) –
- **injection_is_positive** (*bool*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (PriceQuantity) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
PriceQuantity

1.7.26 PriceQuantityUnitless

Python pydantic class corresponding to json type `price.quantity.unitless`.

class gwatn.types.**PriceQuantityUnitless**(**, PriceTimes1000, QuantityTimes1000, TypeName='price.quantity.unitless', Version='000'*)

Parameters

- **PriceTimes1000** (*int*) –
- **QuantityTimes1000** (*int*) –
- **TypeName** (*Literal['price.quantity.unitless']*) –
- **Version** (*str*) –

PriceTimes1000:

- Description:

QuantityTimes1000:

- Description:

class gwatn.types.**PriceQuantityUnitless_Maker**(*price_times1000*, *quantity_times1000*)

Parameters

- **price_times1000** (*int*) –
- **quantity_times1000** (*int*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (PriceQuantityUnitless) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
PriceQuantityUnitless

1.7.27 ScadaCertTransfer

Python pydantic class corresponding to json type `scada.cert.transfer`.

class gwatn.types.**ScadaCertTransfer**(*, *TaAlias*, *SignedProof*, *TypeName='scada.cert.transfer'*, *Version='000'*)

Scada Certificate Transfer.

This is a payload designed to be sent from a SCADA device to the GNodeFactory after the SCADA has opted into its certificate.

Parameters

- **TaAlias** (*str*) –
- **SignedProof** (*str*) –
- **TypeName** (*Literal['scada.cert.transfer']*) –
- **Version** (*str*) –

classmethod check_axiom_1(*v*)

Axiom 1: Scada is SignedProof signer. Axiom 1: Scada is SignedProof signer. There is a ScadaCert created by the Gnf with this ta_alias, and the txn is the OptIn.

Parameters
v (*dict*) –

Return type
dict

TaAlias:

- Description: TerminalAsset Alias. GNodeAlias of the TerminalAsset for which the SCADA certificate is issued. The ScadaCert can be found from this.
- Format: LeftRightDot

SignedProof:

- Description: Signed Proof from the SCADA Actor. The Scada GNode has a ScadaAlgoAddr in the GNodeFactory database, and the identity of the SCADA actor can be verified by this.
- Format: AlgoMsgPackEncoded

class gwatn.types.scada_cert_transfer.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.scada_cert_transfer.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**ScadaCertTransfer_Maker**(*ta_alias*, *signed_proof*)

Parameters

- **ta_alias** (*str*) –
- **signed_proof** (*str*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (ScadaCertTransfer) –

Return type
str

classmethod **type_to_tuple**(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
ScadaCertTransfer

1.7.28 SimplesimDriverData

Python pydantic class corresponding to json type `simplesim.driver.data`.

class gwatn.types.**SimplesimDriverData**(*, *FromGNodeAlias*, *TypeName='simplesim.driver.data'*, *Version='000'*, ***extra_data*)

Generic driver report for a simple Scada Simulation

Parameters

- **FromGNodeAlias** (*str*) –
- **TypeName** (*str*) –
- **Version** (*str*) –
- **extra_data** (*Any*) –

FromGNodeAlias:

- Description:
- Format: LeftRightDot

class gwatn.types.simplesim_driver_data.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**SimplesimDriverData_Maker**(*from_g_node_alias*)

Parameters
from_g_node_alias (*str*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (SimplesimDriverData) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
SimplesimDriverData

1.7.29 SimplesimDriverDataBsh

Python pydantic class corresponding to json type `simplesim.driver.data.bsh`.

class gwatn.types.**SimplesimDriverDataBsh**(*, *FromGNodeAlias*, *PowerWatts*, *StoreKwh*, *MaxStoreKwh*, *TypeName='simplesim.driver.data.bsh'*, *Version='000'*)

Parameters

- **FromGNodeAlias** (*str*) –
- **PowerWatts** (*int*) –
- **StoreKwh** (*int*) –
- **MaxStoreKwh** (*int*) –
- **TypeName** (*Literal['simplesim.driver.data.bsh']*) –
- **Version** (*str*) –

FromGNodeAlias:

- Description:
- Format: LeftRightDot

PowerWatts:

- Description:

StoreKwh:

- Description:

MaxStoreKwh:

- Description:

class gwatn.types.simplesim_driver_data_bsh.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**SimplesimDriverDataBsh_Maker**(*from_g_node_alias*, *power_watts*, *store_kwh*, *max_store_kwh*)

Parameters

- **from_g_node_alias** (*str*) –
- **power_watts** (*int*) –
- **store_kwh** (*int*) –
- **max_store_kwh** (*int*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (SimplesimDriverDataBsh) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
SimplesimDriverDataBsh

1.7.30 SimplesimDriverReport

Python pydantic class corresponding to json type `simplesim.driver.report`.

class gwatn.types.**SimplesimDriverReport**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *DriverDataTypeName*, *DriverData*, *TypeName='simplesim.driver.report'*, *Version='000'*)

Parameters

- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **DriverDataTypeName** (*str*) –
- **DriverData** (SimplesimDriverData) –
- **TypeName** (*Literal['simplesim.driver.report']*) –

- **Version** (*str*) –

FromGNodeAlias:

- Description:
- Format: LeftRightDot

FromGNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

DriverDataTypeName:

- Description:
- Format: LeftRightDot

DriverData:

- Description:

class gwatn.types.simplesim_driver_report.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.simplesim_driver_report.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**SimplesimDriverReport_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *driver_data_type_name*, *driver_data*)

Parameters

- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **driver_data_type_name** (*str*) –
- **driver_data** (SimplesimDriverData) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (SimplesimDriverReport) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
SimplesimDriverReport

1.7.31 SimplesimSnapshotBrickstorageheater

Python pydantic class corresponding to json type `simplesim.snapshot.brickstorageheater`.

class gwatn.types.**SimplesimSnapshotBrickstorageheater**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *PowerWatts*, *StoreKwh*, *MaxStoreKwh*, *AboutTerminalAssetAlias*, *TypeName='simplesim.snapshot.brickstorageheater'*, *Version='000'*)

Parameters

- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **PowerWatts** (*int*) –
- **StoreKwh** (*int*) –
- **MaxStoreKwh** (*int*) –
- **AboutTerminalAssetAlias** (*str*) –
- **TypeName** (*Literal['simplesim.snapshot.brickstorageheater']*) –
- **Version** (*str*) –

FromGNodeAlias:

- Description:
- Format: LeftRightDot

FromGNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

PowerWatts:

- Description:

StoreKwh:

- Description:

MaxStoreKwh:

- Description:

AboutTerminalAssetAlias:

- Description:
- Format: LeftRightDot

class gwatn.types.simplesim_snapshot_brickstorageheater.**check_is_uuid_canonical_textual**(*v*)

UuidCanonicalTextual format: A string of hex words separated by hyphens of length 8-4-4-4-12.

Raises
ValueError – if not UuidCanonicalTextual format

Parameters
v (*str*) –

class gwatn.types.simplesim_snapshot_brickstorageheater.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**SimplesimSnapshotBrickstorageheater_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *power_watts*, *store_kwh*, *max_store_kwh*, *about_terminal_asset_alias*)

Parameters

- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **power_watts** (*int*) –
- **store_kwh** (*int*) –
- **max_store_kwh** (*int*) –
- **about_terminal_asset_alias** (*str*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (SimplesimSnapshotBrickstorageheater) –

Return type
str

classmethod **type_to_tuple**(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
SimplesimSnapshotBrickstorageheater

1.7.32 SlaEnter

Python pydantic class corresponding to json type `sla.enter`.

class gwatn.types.**SlaEnter**(*, *TerminalAssetAlias*, *TypeName='sla.enter'*, *Version='000'*)

Parameters

- **TerminalAssetAlias** (*str*) –
- **TypeName** (*Literal['sla.enter']*) –
- **Version** (*str*) –

TerminalAssetAlias:

- Description:
- Format: LeftRightDot

class gwatn.types.sla_enter.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**SlaEnter_Maker**(*terminal_asset_alias*)

Parameters
terminal_asset_alias (*str*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (SlaEnter) –

Return type
str

classmethod **type_to_tuple**(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
SlaEnter

1.7.33 TadeedSpecsHack

Python pydantic class corresponding to json type `tadeed.specs.hack`.

class gwatn.types.**TadeedSpecsHack**(*, *TerminalAssetAlias*, *MicroLat*, *MicroLon*, *DaemonPort*, *TypeName='tadeed.specs.hack'*, *Version='000'*)

Parameters

- **TerminalAssetAlias** (*str*) –
- **MicroLat** (*int*) –
- **MicroLon** (*int*) –
- **DaemonPort** (*int*) –
- **TypeName** (*Literal['tadeed.specs.hack']*) –
- **Version** (*str*) –

TerminalAssetAlias:

- Description:
- Format: LeftRightDot

MicroLat:

- Description:

MicroLon:

- Description:

DaemonPort:

- Description:

class gwatn.types.tadeed_specs_hack.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.**TadeedSpecsHack_Maker**(*terminal_asset_alias*, *micro_lat*, *micro_lon*, *daemon_port*)

Parameters

- **terminal_asset_alias** (*str*) –
- **micro_lat** (*int*) –
- **micro_lon** (*int*) –
- **daemon_port** (*int*) –

classmethod **tuple_to_type**(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (TadeedSpecsHack) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
TadeedSpecsHack

1.7.34 TavalidatorcertAlgoCreate

Python pydantic class corresponding to json type `tavalidatorcert.algo.create`.

class gwatn.types.**TavalidatorcertAlgoCreate**(*, *ValidatorAddr*, *HalfSignedCertCreationMtx*, *TypeName='tavalidatorcert.algo.create'*, *Version='000'*)

Used for Step 1 of TaValidator certification.

Meant to be sent from a pending TaValidator to the GNodeFactory (Gnf), to initiate the process of certifying the pending TaValidator. [More info](https://gridworks.readthedocs.io/en/latest/ta-validator.html).

Parameters

- **ValidatorAddr** (*str*) –
- **HalfSignedCertCreationMtx** (*str*) –
- **TypeName** (*Literal['tavalidatorcert.algo.create']*) –
- **Version** (*str*) –

classmethod check_axiom_1(*v*)

Axiom 1: Is correct Multisig. Decoded HalfSignedCertCreationMtx must have type MultisigTransaction from the 2-sig MultiAccount [GnfAdminAddr, ValidatorAddr], signed by ValidatorAddr. [More info](https://gridworks.readthedocs.io/en/latest/g-node-factory.html#gnfadminaddr)

Parameters
v (*dict*) –

Return type
dict

classmethod check_half_signed_cert_creation_mtx(*v*)

Axioms 2, 3:

Axiom 2: Is AssetConfigTxn. The transaction must have type AssetConfigTxn.

Axiom 3: Is ValidatorCert. For the asset getting created: Total is 1, Decimals is 0, UnitName is VLDTR, Manager is GnfAdminAddr, AssetName is not blank. [More info](https://gridworks.readthedocs.io/en/latest/ta-validator.html#tavalidator-certificate)

Parameters
v (*str*) –

Return type
str

classmethod check_validator_addr(*v*)

Axiom 5: Uniqueness. There must not already be a TaValidatorCert belonging to the 2-sig [GnfAdminAddr, ValidatorAddr] address.

Parameters
v (*str*) –

Return type
str

ValidatorAddr:

- Description: The address of the pending TaValidator
- Format: AlgoAddressStringFormat

HalfSignedCertCreationMtx:

- Description: Algo multi-transaction for certificate creation, with 1 of 2 signatures
- Format: AlgoMsgPackEncoded

class gwatn.types.tavalidatorcert_algo_create.check_is_algo_address_string_format(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.tavalidatorcert_algo_create.check_is_algo_msg_pack_encoded(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.TavalidatorcertAlgoCreate_Maker(*validator_addr*, *half_signed_cert_creation_mtx*)

Parameters

- **validator_addr** (*str*) –
- **half_signed_cert_creation_mtx** (*str*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (TavalidatorcertAlgoCreate) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
TavalidatorcertAlgoCreate

1.7.35 TavalidatorcertAlgoTransfer

Python pydantic class corresponding to json type `tavalidatorcert.algo.transfer`.

class gwatn.types.**TavalidatorcertAlgoTransfer**(*, *ValidatorAddr*, *HalfSignedCertTransferMtx*, *TypeName='tavalidatorcert.algo.transfer'*, *Version='000'*)

Used for Step 2 of TaValidator certification.

Meant to be sent from a pending TaValidator to the GNodeFactory (Gnf), so the Gnf will transfer its ValidatorCert to the pending TaValidator's Algorand address. [More info](https://gridworks.readthedocs.io/en/latest/ta-validator.html).

Parameters

- **ValidatorAddr** (*str*) –
- **HalfSignedCertTransferMtx** (*str*) –
- **TypeName** (*Literal['tavalidatorcert.algo.transfer']*) –
- **Version** (*str*) –

classmethod check_axiom_1(*v*)

Axiom 1: Is correct Multisig. Decoded HalfSignedCertTransferMtx must have type MultisigTransaction from the 2-sig MultiAccount [GnfAdminAddr, ValidatorAddr], signed by the ValidatorAddr [More info](https://gridworks.readthedocs.io/en/latest/g-node-factory.html#gnfadminaddr)

Parameters
v (*dict*) –

Return type
dict

classmethod check_axiom_2(*v*)

Axiom 2: Transfers correct certificate.

- The transaction must be the transfer of an Algorand Standard Asset

-The sender must be the 2-sig Multi [GnfAdminAddr, TaValidatorAddr], which also created and owns the ASA - It must be getting sent to the ValidatorAddr -The ASA must have:

- Total = 1
- UnitName=VLDITR
- GnfAdminAddr as manage
- AssetName not blank

- The transfer amount must be 1

[More info](https://gridworks.readthedocs.io/en/latest/ta-validator.html#tavalidator-certificate)

Parameters
v (*dict*) –

Return type
dict

classmethod check_axiom_3(*v*)

Axiom 3: TaValidator has opted in. ValidatorAddr must be opted into the transferring ASA.

Parameters
v (*dict*) –

Return type
dict

classmethod check_validator_addr(*v*)

Axiom 4: TaValidator has sufficient Algos. MultiAccount [GnfAdminAddr, ValidatorAddr] must have enough Algos to meet the GNodeFactory criterion.

Parameters
v (*str*) –

Return type
str

ValidatorAddr:

- Description: The address of the pending TaValidator
- Format: AlgoAddressStringFormat

HalfSignedCertTransferMtx:

- Description: Algo multi-transaction for certificate transfer, with 1 of 2 signatures
- Format: AlgoMsgPackEncoded

class gwatn.types.tavalidatorcert_algo_transfer.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.tavalidatorcert_algo_transfer.**check_is_algo_msg_pack_encoded**(*v*)

AlgoMSgPackEncoded format: the format of an transaction sent to the Algorand blockchain.

Raises
ValueError – if not AlgoMSgPackEncoded format

Parameters
v (*str*) –

class gwatn.types.**TavalidatorcertAlgoTransfer_Maker**(*validator_addr*, *half_signed_cert_transfer_mtx*)

Parameters

- **validator_addr** (*str*) –
- **half_signed_cert_transfer_mtx** (*str*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (TavalidatorcertAlgoTransfer) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
TavalidatorcertAlgoTransfer

1.7.36 TerminalassetCertifyHack

Python pydantic class corresponding to json type `terminalasset.certify.hack`.

class gwatn.types.**TerminalassetCertifyHack**(*, *TerminalAssetAlias*, *TaDaemonApiPort*, *TaDaemonApiFqdn*, *TaDaemonAddr*, *TypeName='terminalasset.certify.hack'*, *Version='000'*)

Parameters

- **TerminalAssetAlias** (*str*) –
- **TaDaemonApiPort** (*str*) –
- **TaDaemonApiFqdn** (*str*) –
- **TaDaemonAddr** (*str*) –
- **TypeName** (*Literal['terminalasset.certify.hack']*) –
- **Version** (*str*) –

TerminalAssetAlias:

- Description:
- Format: LeftRightDot

TaDaemonApiPort:

- Description:

TaDaemonApiFqdn:

- Description:

TaDaemonAddr:

- Description:
- Format: AlgoAddressStringFormat

class gwatn.types.terminalasset_certify_hack.**check_is_left_right_dot**(*v*)

LeftRightDot format: Lowercase alphanumeric words separated by periods, most significant word (on the left) starting with an alphabet character.

Raises
ValueError – if not LeftRightDot format

Parameters
v (*str*) –

class gwatn.types.terminalasset_certify_hack.**check_is_algo_address_string_format**(*v*)

AlgoAddressStringFormat format: The public key of a private/public Ed25519 key pair, transformed into an Algorand address, by adding a 4-byte checksum to the end of the public key and then encoding in base32.

Raises
ValueError – if not AlgoAddressStringFormat format

Parameters
v (*str*) –

class gwatn.types.**TerminalassetCertifyHack_Maker**(*terminal_asset_alias*, *ta_daemon_api_port*, *ta_daemon_api_fqdn*, *ta_daemon_addr*)

Parameters

- **terminal_asset_alias** (*str*) –
- **ta_daemon_api_port** (*str*) –
- **ta_daemon_api_fqdn** (*str*) –
- **ta_daemon_addr** (*str*) –

classmethod tuple_to_type(*tuple*)

Given a Python class object, returns the serialized JSON type object

Parameters
tuple (TerminalassetCertifyHack) –

Return type
str

classmethod type_to_tuple(*t*)

Given a serialized JSON type object, returns the Python class object

Parameters
t (*str*) –

Return type
TerminalassetCertifyHack

1.8 Contributor Guide

Thank you for your interest in improving this project. This project welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- Source Code
- Documentation
- Issue Tracker
- *Code of Conduct*

1.8.1 How to report a bug

Report bugs on the Issue Tracker.

When filing an issue, make sure to answer these questions:

- Which operating system and Python version are you using?
- Which version of this project are you using?
- What did you do?
- What did you expect to see?
- What did you see instead?

The best way to get your bug fixed is to provide a test case, and/or steps to reproduce the issue.

1.8.2 How to request a feature

Request features on the Issue Tracker.

1.8.3 How to set up your development environment

You need Python10+ and the following tools:

- Poetry
- Nox
- nox-poetry

Install the package with development requirements:

```
$ poetry install
```

You can now run an interactive Python session, or the command-line interface:

```
$ poetry run python
$ poetry run gridworks-atn
```

1.8.4 How to test the project

Run the full test suite:

```
$ nox
```

List the available Nox sessions:

```
$ nox --list-sessions
```

You can also run a specific Nox session. For example, invoke the unit test suite like this:

```
$ nox --session=tests
```

Unit tests are located in the *tests* directory, and are written using the pytest testing framework.

1.8.5 How to submit changes

Open a pull request to submit changes to this project.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains 100% code coverage.
- If your changes add functionality, update the documentation accordingly.

Feel free to submit early, though—we can always iterate on this.

To run linting and code formatting checks before committing your change, you can install pre-commit as a Git hook by running the following command:

```
$ nox --session=pre-commit -- install
```

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

1.9 GridWorks Energy Consulting Code of Conduct

1.9.1 Basic Truth

All humans are worthy.

1.9.2 Scope

This Code of Conduct applies to moderation of comments, issues and commits within this repository to support its alignment to the above basic truth.

1.9.3 Enforcement Responsibilities

GridWorks Energy Consulting LLC (gridworks@gridworks-consulting.com) owns and administers this repository, and is ultimately responsible for enforcement of standards of behavior. They are responsible for merges to dev and main branches, and maintain the right and responsibility to remove, edit, or reject comments, commits, code, docuentation edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

If you read something in this repo that you want GridWorks to consider moderating, please send an email to them at gridworks@gridworks-consulting.com. All complaints will be reviewed and investigated, and GridWorks will respect the privacy and security of the reporter of any incident.

1.9.4 What not to add to this repo

Ways to trigger GridWorks moderation enforcement:

- Publish others' private information, such as a physical or email address, without their explicit permission
- Use of sexualized language or imagery, or make sexual advances
- Troll

1.9.5 Suggestions

- Empathize
- Recognize you are worthy of contributing, and do so in the face of confusion and doubt; you can help clarify things for everyone
- Be interested in differing opinions, viewpoints, and experiences
- Give and accept constructive feedback
- Accept responsibility for your mistakes and learn from them
- Recognize everybody makes mistakes, and forgive
- Focus on the highest good for all

1.9.6 Enforcement Escalation

1. Correction

A private, written request from GridWorks to change or edit a comment, commit, or issue.

2. Warning

With a warning, GridWorks may remove your comments, commits or issues. They may also freeze a conversation.

3. Temporary Ban

A temporary ban from any sort of interaction or public communication within the repository for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

A permanent ban from any sort of interaction within the repository.

1.9.7 Attribution

This Code of Conduct is loosely adapted from the Contributor Covenant, version 2.1, available at https://www.contributor-covenant.org/version/2/1/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

1.10 License


```
MIT License

Copyright © 2022 GridWorks

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

PYTHON MODULE INDEX

g

gridworks.data_classes.base_g_node, 76
gridworks.data_classes.g_node, 77
gridworks.data_classes.g_node_instance, 78
gridworks.data_classes.gps_point, 79
gridworks.data_classes.market_type, 79
gridworks.enums, 79
gwatn.types, 86

INDEX

A

AcceptedBid (*class in gwatn.types*), 86
AcceptedBid_Maker (*class in gwatn.types*), 87
AlgoCertType (*class in gridworks.enums*), 79
AtnBid (*class in gwatn.types*), 87
AtnBid_Maker (*class in gwatn.types*), 89
AtnOutsideTempRegrCoeffs (*class in gwatn.types*), 90
AtnOutsideTempRegrCoeffs_Maker (*class in gwatn.types*), 91
AtnParams (*class in gwatn.types*), 91
AtnParams_Maker (*class in gwatn.types*), 92
AtnParamsBrickstorageheater (*class in gwatn.types*), 92
AtnParamsBrickstorageheater_Maker (*class in gwatn.types*), 95
AtnParamsReport (*class in gwatn.types*), 96
AtnParamsReport_Maker (*class in gwatn.types*), 97
AtnParamsSimpleresistivehydronic (*class in gwatn.types*), 98
AtnParamsSimpleresistivehydronic_Maker (*class in gwatn.types*), 101

B

BaseGNode (*class in gridworks.data_classes.base_g_node*), 76
BasegnodeScadaCreate (*class in gwatn.types*), 103
BasegnodeScadaCreate_Maker (*class in gwatn.types*), 104

C

check_axiom_0() (*gwatn.types.JoinDispatchContract class method*), 131
check_axiom_1() (*gwatn.types.AtnBid class method*), 88
check_axiom_1() (*gwatn.types.BasegnodeScadaCreate class method*), 103
check_axiom_1() (*gwatn.types.DispatchContractConfirmed class method*), 107
check_axiom_1() (*gwatn.types.InitialTadeedAlgoCreate class method*), 125
check_axiom_1() (*gwatn.types.InitialTadeedAlgoOptin class method*), 127
check_axiom_1() (*gwatn.types.InitialTadeedAlgoTransfer class method*), 129
check_axiom_1() (*gwatn.types.ScadaCertTransfer class method*), 145
check_axiom_1() (*gwatn.types.TavalidatorcertAlgoCreate class method*), 154
check_axiom_1() (*gwatn.types.TavalidatorcertAlgoTransfer class method*), 156
check_axiom_2() (*gwatn.types.AtnBid class method*), 88
check_axiom_2() (*gwatn.types.BasegnodeScadaCreate class method*), 103
check_axiom_2() (*gwatn.types.InitialTadeedAlgoCreate class method*), 125
check_axiom_2() (*gwatn.types.InitialTadeedAlgoOptin class method*), 127
check_axiom_2() (*gwatn.types.InitialTadeedAlgoTransfer class method*), 129
check_axiom_2() (*gwatn.types.TavalidatorcertAlgoTransfer class method*), 156
check_axiom_3() (*gwatn.types.InitialTadeedAlgoCreate class method*), 125
check_axiom_3() (*gwatn.types.TavalidatorcertAlgoTransfer class method*), 157
check_half_signed_cert_creation_mtx() (*gwatn.types.TavalidatorcertAlgoCreate class method*), 154
check_is_algo_address_string_format (*class in gwatn.types.atn_bid*), 89
check_is_algo_address_string_format (*class in gwatn.types.basegnode_scada_create*), 104
check_is_algo_address_string_format (*class in gwatn.types.discoverycert_algo_create*), 106
check_is_algo_address_string_format (*class in gwatn.types.initial_tadeed_algo_create*), 126
check_is_algo_address_string_format (*class in gwatn.types.initial_tadeed_algo_optin*), 128
check_is_algo_address_string_format (*class in gwatn.types.initial_tadeed_algo_transfer*), 130
check_is_algo_address_string_format (*class in gwatn.types.new_tadeed_algo_optin*), 139
check_is_algo_address_string_format (*class in*

gwatn.types.new_tadeed_send), 140
check_is_algo_address_string_format (*class in gwatn.types.old_tadeed_algo_return*), 142
check_is_algo_address_string_format (*class in gwatn.types.tavalidatorcert_algo_create*), 155
check_is_algo_address_string_format (*class in gwatn.types.tavalidatorcert_algo_transfer*), 157
check_is_algo_address_string_format (*class in gwatn.types.terminalasset_certify_hack*), 159
check_is_algo_msg_pack_encoded (*class in gwatn.types.atn_bid*), 89
check_is_algo_msg_pack_encoded (*class in gwatn.types.basegnode_scada_create*), 104
check_is_algo_msg_pack_encoded (*class in gwatn.types.dispatch_contract_confirmed*), 108
check_is_algo_msg_pack_encoded (*class in gwatn.types.initial_tadeed_algo_create*), 126
check_is_algo_msg_pack_encoded (*class in gwatn.types.initial_tadeed_algo_optin*), 128
check_is_algo_msg_pack_encoded (*class in gwatn.types.initial_tadeed_algo_transfer*), 130
check_is_algo_msg_pack_encoded (*class in gwatn.types.join_dispatch_contract*), 132
check_is_algo_msg_pack_encoded (*class in gwatn.types.new_tadeed_algo_optin*), 139
check_is_algo_msg_pack_encoded (*class in gwatn.types.new_tadeed_send*), 141
check_is_algo_msg_pack_encoded (*class in gwatn.types.old_tadeed_algo_return*), 142
check_is_algo_msg_pack_encoded (*class in gwatn.types.scada_cert_transfer*), 146
check_is_algo_msg_pack_encoded (*class in gwatn.types.tavalidatorcert_algo_create*), 155
check_is_algo_msg_pack_encoded (*class in gwatn.types.tavalidatorcert_algo_transfer*), 157
check_is_iso_format (*class in gwatn.types.latest_price*), 134
check_is_left_right_dot (*class in gwatn.types.accepted_bid*), 86
check_is_left_right_dot (*class in gwatn.types.atn_bid*), 89
check_is_left_right_dot (*class in gwatn.types.atn_params*), 92
check_is_left_right_dot (*class in gwatn.types.atn_params_brickstorageheater*), 95
check_is_left_right_dot (*class in gwatn.types.atn_params_report*), 97
check_is_left_right_dot (*class in gwatn.types.atn_params_simpleresistivehydronic*), 101
check_is_left_right_dot (*class in gwatn.types.basegnode_scada_create*), 104
check_is_left_right_dot (*class in gwatn.types.discoverycert_algo_create*), 106
check_is_left_right_dot (*class in gwatn.types.dispatch_contract_confirmed*), 108
check_is_left_right_dot (*class in gwatn.types.flo_params*), 110
check_is_left_right_dot (*class in gwatn.types.flo_params_brickstorageheater*), 114
check_is_left_right_dot (*class in gwatn.types.flo_params_report*), 117
check_is_left_right_dot (*class in gwatn.types.flo_params_simpleresistivehydronic*), 122
check_is_left_right_dot (*class in gwatn.types.initial_tadeed_algo_optin*), 128
check_is_left_right_dot (*class in gwatn.types.join_dispatch_contract*), 132
check_is_left_right_dot (*class in gwatn.types.latest_price*), 134
check_is_left_right_dot (*class in gwatn.types.market_slot*), 136
check_is_left_right_dot (*class in gwatn.types.scada_cert_transfer*), 145
check_is_left_right_dot (*class in gwatn.types.simplesim_driver_data*), 146
check_is_left_right_dot (*class in gwatn.types.simplesim_driver_data_bsh*), 148
check_is_left_right_dot (*class in gwatn.types.simplesim_driver_report*), 149
check_is_left_right_dot (*class in gwatn.types.simplesim_snapshot_brickstorageheater*), 151
check_is_left_right_dot (*class in gwatn.types.sla_enter*), 152
check_is_left_right_dot (*class in gwatn.types.tadeed_specs_hack*), 153
check_is_left_right_dot (*class in gwatn.types.terminalasset_certify_hack*), 158
check_is_market_slot_name_lrd_format (*class in gwatn.types.accepted_bid*), 86
check_is_market_slot_name_lrd_format (*class in gwatn.types.atn_bid*), 89
check_is_market_slot_name_lrd_format (*class in gwatn.types.latest_price*), 134

check_is_reasonable_unix_time_s (*class in gwatn.types.atn_params_report*), 97
check_is_reasonable_unix_time_s (*class in gwatn.types.flo_params_report*), 117
check_is_reasonable_unix_time_s (*class in gwatn.types.market_slot*), 136
check_is_uuid_canonical_textual (*class in gwatn.types.atn_bid*), 89
check_is_uuid_canonical_textual (*class in gwatn.types.atn_params_report*), 97
check_is_uuid_canonical_textual (*class in gwatn.types.dispatch_contract_confirmed*), 108
check_is_uuid_canonical_textual (*class in gwatn.types.flo_params*), 110
check_is_uuid_canonical_textual (*class in gwatn.types.flo_params_brickstorageheater*), 114
check_is_uuid_canonical_textual (*class in gwatn.types.flo_params_report*), 117
check_is_uuid_canonical_textual (*class in gwatn.types.flo_params_simpleresistivehydronic*), 122
check_is_uuid_canonical_textual (*class in gwatn.types.join_dispatch_contract*), 132
check_is_uuid_canonical_textual (*class in gwatn.types.latest_price*), 134
check_is_uuid_canonical_textual (*class in gwatn.types.simplesim_driver_report*), 149
check_is_uuid_canonical_textual (*class in gwatn.types.simplesim_snapshot_brickstorageheater*), 150
check_validator_addr() (*gwatn.types.TavalidatorcertAlgoCreate class method*), 154
check_validator_addr() (*gwatn.types.TavalidatorcertAlgoTransfer class method*), 157
children() (*gridworks.data_classes.base_g_node.BaseGNode method*), 76
children() (*gridworks.data_classes.g_node.GNode method*), 77
children() (*gridworks.data_classes.g_node_instance.GNodeInstance method*), 78
CoreGNodeRole (*class in gridworks.enums*), 79

D

default() (*gridworks.enums.AlgoCertType class method*), 79
default() (*gridworks.enums.CoreGNodeRole class method*), 79
default() (*gridworks.enums.GniStatus class method*), 81
default() (*gridworks.enums.GNodeRole class method*), 80
default() (*gridworks.enums.GNodeStatus class method*), 81
default() (*gridworks.enums.MarketPriceUnit class method*), 81
default() (*gridworks.enums.MarketQuantityUnit class method*), 82
default() (*gridworks.enums.MarketTypeName class method*), 82
default() (*gridworks.enums.MessageCategory class method*), 83
default() (*gridworks.enums.MessageCategorySymbol class method*), 83
default() (*gridworks.enums.RecognizedCurrencyUnit class method*), 83
default() (*gridworks.enums.StrategyName class method*), 84
default() (*gridworks.enums.SupervisorContainerStatus class method*), 85
default() (*gridworks.enums.UniverseType class method*), 85
DiscoverycertAlgoCreate (*class in gwatn.types*), 105
DiscoverycertAlgoCreate_Maker (*class in gwatn.types*), 106
DispatchContractConfirmed (*class in gwatn.types*), 107
DispatchContractConfirmed_Maker (*class in gwatn.types*), 108

F

FloParams (*class in gwatn.types*), 109
FloParams_Maker (*class in gwatn.types*), 110
FloParamsBrickstorageheater (*class in gwatn.types*), 111
FloParamsBrickstorageheater_Maker (*class in gwatn.types*), 114
FloParamsReport (*class in gwatn.types*), 116
FloParamsReport_Maker (*class in gwatn.types*), 117
FloParamsSimpleresistivehydronic (*class in gwatn.types*), 119
FloParamsSimpleresistivehydronic_Maker (*class in gwatn.types*), 123

G

GniStatus (*class in gridworks.enums*), 81
GNode (*class in gridworks.data_classes.g_node*), 77
GNodeInstance (*class in gridworks.data_classes.g_node_instance*), 78
GNodeRole (*class in gridworks.enums*), 79
GNodeStatus (*class in gridworks.enums*), 80
gridworks.data_classes.base_g_node module, 76
gridworks.data_classes.g_node

module, 77
gridworks.data_classes.g_node_instance
module, 78
gridworks.data_classes.gps_point
module, 79
gridworks.data_classes.market_type
module, 79
gridworks.enums
module, 79
gwatn.types
module, 86

I

InitialTadeedAlgoCreate (*class in gwatn.types*), 125
InitialTadeedAlgoCreate_Maker (*class in gwatn.types*), 126
InitialTadeedAlgoOptin (*class in gwatn.types*), 127
InitialTadeedAlgoOptin_Maker (*class in gwatn.types*), 128
InitialTadeedAlgoTransfer (*class in gwatn.types*), 129
InitialTadeedAlgoTransfer_Maker (*class in gwatn.types*), 130
is_copper() (*gridworks.data_classes.base_g_node.BaseGNode method*), 76

J

JoinDispatchContract (*class in gwatn.types*), 131
JoinDispatchContract_Maker (*class in gwatn.types*), 132

L

LatestPrice (*class in gwatn.types*), 133
LatestPrice_Maker (*class in gwatn.types*), 134

M

MarketPriceUnit (*class in gridworks.enums*), 81
MarketQuantityUnit (*class in gridworks.enums*), 81
MarketSlot (*class in gwatn.types*), 135
MarketSlot_Maker (*class in gwatn.types*), 136
MarketTypeGt (*class in gwatn.types*), 137
MarketTypeGt_Maker (*class in gwatn.types*), 137
MarketTypeName (*class in gridworks.enums*), 82
MessageCategory (*class in gridworks.enums*), 82
MessageCategorySymbol (*class in gridworks.enums*), 83
module
gridworks.data_classes.base_g_node, 76
gridworks.data_classes.g_node, 77
gridworks.data_classes.g_node_instance, 78
gridworks.data_classes.gps_point, 79
gridworks.data_classes.market_type, 79
gridworks.enums, 79
gwatn.types, 86

N

NewTadeedAlgoOptin (*class in gwatn.types*), 138
NewTadeedAlgoOptin_Maker (*class in gwatn.types*), 139
NewTadeedSend (*class in gwatn.types*), 140
NewTadeedSend_Maker (*class in gwatn.types*), 141

O

OldTadeedAlgoReturn (*class in gwatn.types*), 141
OldTadeedAlgoReturn_Maker (*class in gwatn.types*), 142

P

parent() (*gridworks.data_classes.base_g_node.BaseGNode method*), 76
parent() (*gridworks.data_classes.g_node.GNode method*), 77
parent() (*gridworks.data_classes.g_node_instance.GNodeInstance method*), 78
parent_from_alias() (*gridworks.data_classes.base_g_node.BaseGNode class method*), 77
parent_from_alias() (*gridworks.data_classes.g_node.GNode class method*), 77
parent_from_alias() (*gridworks.data_classes.g_node_instance.GNodeInstance class method*), 78
PriceQuantity (*class in gwatn.types*), 143
PriceQuantity_Maker (*class in gwatn.types*), 143
PriceQuantityUnitless (*class in gwatn.types*), 144
PriceQuantityUnitless_Maker (*class in gwatn.types*), 144

R

RecognizedCurrencyUnit (*class in gridworks.enums*), 83

S

ScadaCertTransfer (*class in gwatn.types*), 145
ScadaCertTransfer_Maker (*class in gwatn.types*), 146
SimplesimDriverData (*class in gwatn.types*), 146
SimplesimDriverData_Maker (*class in gwatn.types*), 147
SimplesimDriverDataBsh (*class in gwatn.types*), 147
SimplesimDriverDataBsh_Maker (*class in gwatn.types*), 148
SimplesimDriverReport (*class in gwatn.types*), 148
SimplesimDriverReport_Maker (*class in gwatn.types*), 149
SimplesimSnapshotBrickstorageheater (*class in gwatn.types*), 150

SimplesimSnapshotBrickstorageheater_Maker (*class in gwatn.types*), 151
SlaEnter (*class in gwatn.types*), 152
SlaEnter_Maker (*class in gwatn.types*), 152
StrategyName (*class in gridworks.enums*), 84
SupervisorContainerStatus (*class in gridworks.enums*), 85

T

TadeedSpecsHack (*class in gwatn.types*), 153
TadeedSpecsHack_Maker (*class in gwatn.types*), 153
TavalidatorcertAlgoCreate (*class in gwatn.types*), 154
TavalidatorcertAlgoCreate_Maker (*class in gwatn.types*), 155
TavalidatorcertAlgoTransfer (*class in gwatn.types*), 156
TavalidatorcertAlgoTransfer_Maker (*class in gwatn.types*), 157
TerminalassetCertifyHack (*class in gwatn.types*), 158
TerminalassetCertifyHack_Maker (*class in gwatn.types*), 159
tuple_to_type() (*gwatn.types.AcceptedBid_Maker class method*), 87
tuple_to_type() (*gwatn.types.AtnBid_Maker class method*), 90
tuple_to_type() (*gwatn.types.AtnOutsideTempRegrCoeffs_Maker class method*), 91
tuple_to_type() (*gwatn.types.AtnParams_Maker class method*), 92
tuple_to_type() (*gwatn.types.AtnParamsBrickstorageheater_Maker class method*), 96
tuple_to_type() (*gwatn.types.AtnParamsReport_Maker class method*), 98
tuple_to_type() (*gwatn.types.AtnParamsSimpleresistivehydronic_Maker class method*), 102
tuple_to_type() (*gwatn.types.BasegnodeScadaCreate_Maker class method*), 104
tuple_to_type() (*gwatn.types.DiscoverycertAlgoCreate_Maker class method*), 106
tuple_to_type() (*gwatn.types.DispatchContractConfirmed_Maker class method*), 108
tuple_to_type() (*gwatn.types.FloParams_Maker class method*), 110
tuple_to_type() (*gwatn.types.FloParamsBrickstorageheater_Maker class method*), 116
tuple_to_type() (*gwatn.types.FloParamsReport_Maker class method*), 118
tuple_to_type() (*gwatn.types.FloParamsSimpleresistivehydronic_Maker class method*), 124
tuple_to_type() (*gwatn.types.InitialTadeedAlgoCreate_Maker class method*), 126
tuple_to_type() (*gwatn.types.InitialTadeedAlgoOptin_Maker class method*), 128
tuple_to_type() (*gwatn.types.InitialTadeedAlgoTransfer_Maker class method*), 131
tuple_to_type() (*gwatn.types.JoinDispatchContract_Maker class method*), 133
tuple_to_type() (*gwatn.types.LatestPrice_Maker class method*), 135
tuple_to_type() (*gwatn.types.MarketSlot_Maker class method*), 136
tuple_to_type() (*gwatn.types.MarketTypeGt_Maker class method*), 138
tuple_to_type() (*gwatn.types.NewTadeedAlgoOptin_Maker class method*), 139
tuple_to_type() (*gwatn.types.NewTadeedSend_Maker class method*), 141
tuple_to_type() (*gwatn.types.OldTadeedAlgoReturn_Maker class method*), 142
tuple_to_type() (*gwatn.types.PriceQuantity_Maker class method*), 143
tuple_to_type() (*gwatn.types.PriceQuantityUnitless_Maker class method*), 144
tuple_to_type() (*gwatn.types.ScadaCertTransfer_Maker class method*), 146
tuple_to_type() (*gwatn.types.SimplesimDriverData_Maker class method*), 147
tuple_to_type() (*gwatn.types.SimplesimDriverDataBsh_Maker class method*), 148
tuple_to_type() (*gwatn.types.SimplesimDriverReport_Maker class method*), 149
tuple_to_type() (*gwatn.types.SimplesimSnapshotBrickstorageheater_Maker class method*), 151
tuple_to_type() (*gwatn.types.SlaEnter_Maker class method*), 152
tuple_to_type() (*gwatn.types.TadeedSpecsHack_Maker class method*), 153
tuple_to_type() (*gwatn.types.TavalidatorcertAlgoCreate_Maker class method*), 155
tuple_to_type() (*gwatn.types.TavalidatorcertAlgoTransfer_Maker class method*), 157
tuple_to_type() (*gwatn.types.TerminalassetCertifyHack_Maker class method*), 159
type_to_tuple() (*gwatn.types.AcceptedBid_Maker class method*), 87
type_to_tuple() (*gwatn.types.AtnBid_Maker class method*), 90
type_to_tuple() (*gwatn.types.AtnOutsideTempRegrCoeffs_Maker class method*), 91
type_to_tuple() (*gwatn.types.AtnParams_Maker class method*), 92
type_to_tuple() (*gwatn.types.AtnParamsBrickstorageheater_Maker class method*), 96
type_to_tuple() (*gwatn.types.AtnParamsReport_Maker class method*), 98

type_to_tuple() (*gwatn.types.AtnParamsSimpleresistivehydronic_Maker class method*), 102
type_to_tuple() (*gwatn.types.BasegnodeScadaCreate_Maker class method*), 105
type_to_tuple() (*gwatn.types.DiscoverycertAlgoCreate_Maker class method*), 106
type_to_tuple() (*gwatn.types.DispatchContractConfirmed_Maker class method*), 108
type_to_tuple() (*gwatn.types.FloParams_Maker class method*), 111
type_to_tuple() (*gwatn.types.FloParamsBrickstorageheater_Maker class method*), 116
type_to_tuple() (*gwatn.types.FloParamsReport_Maker class method*), 118
type_to_tuple() (*gwatn.types.FloParamsSimpleresistivehydronic_Maker class method*), 124
type_to_tuple() (*gwatn.types.InitialTadeedAlgoCreate_Maker class method*), 126
type_to_tuple() (*gwatn.types.InitialTadeedAlgoOptin_Maker class method*), 128
type_to_tuple() (*gwatn.types.InitialTadeedAlgoTransfer_Maker class method*), 131
type_to_tuple() (*gwatn.types.JoinDispatchContract_Maker class method*), 133
type_to_tuple() (*gwatn.types.LatestPrice_Maker class method*), 135
type_to_tuple() (*gwatn.types.MarketSlot_Maker class method*), 136
type_to_tuple() (*gwatn.types.MarketTypeGt_Maker class method*), 138
type_to_tuple() (*gwatn.types.NewTadeedAlgoOptin_Maker class method*), 139
type_to_tuple() (*gwatn.types.NewTadeedSend_Maker class method*), 141
type_to_tuple() (*gwatn.types.OldTadeedAlgoReturn_Maker class method*), 142
type_to_tuple() (*gwatn.types.PriceQuantity_Maker class method*), 144
type_to_tuple() (*gwatn.types.PriceQuantityUnitless_Maker class method*), 144
type_to_tuple() (*gwatn.types.ScadaCertTransfer_Maker class method*), 146
type_to_tuple() (*gwatn.types.SimplesimDriverData_Maker class method*), 147
type_to_tuple() (*gwatn.types.SimplesimDriverDataBsh_Maker class method*), 148
type_to_tuple() (*gwatn.types.SimplesimDriverReport_Maker class method*), 149
type_to_tuple() (*gwatn.types.SimplesimSnapshotBrickstorageheater_Maker class method*), 151
type_to_tuple() (*gwatn.types.SlaEnter_Maker class method*), 152
type_to_tuple() (*gwatn.types.TadeedSpecsHack_Maker class method*), 154
type_to_tuple() (*gwatn.types.TavalidatorcertAlgoCreate_Maker class method*), 155
type_to_tuple() (*gwatn.types.TavalidatorcertAlgoTransfer_Maker class method*), 158
type_to_tuple() (*gwatn.types.TerminalassetCertifyHack_Maker class method*), 159

U

UniverseType (*class in gridworks.enums*), 85

V

values() (*gridworks.enums.AlgoCertType class method*), 79
values() (*gridworks.enums.CoreGNodeRole class method*), 79
values() (*gridworks.enums.GniStatus class method*), 81
values() (*gridworks.enums.GNodeRole class method*), 80
values() (*gridworks.enums.GNodeStatus class method*), 81
values() (*gridworks.enums.MarketPriceUnit class method*), 81
values() (*gridworks.enums.MarketQuantityUnit class method*), 82
values() (*gridworks.enums.MarketTypeName class method*), 82
values() (*gridworks.enums.MessageCategory class method*), 83
values() (*gridworks.enums.MessageCategorySymbol class method*), 83
values() (*gridworks.enums.RecognizedCurrencyUnit class method*), 84
values() (*gridworks.enums.StrategyName class method*), 84
values() (*gridworks.enums.SupervisorContainerStatus class method*), 85
values() (*gridworks.enums.UniverseType class method*), 85