Gridworks AtomicTNode

gridworks

Apr 29, 2023

CODE SUPPORT

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

1.2 Type API Specs

1.2.1 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": {
    "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",
```

(continues on next page)

(continued from previous page)

```
      "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
    }
  }
}
```

1.2.2 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": {
```

(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"
    },
    "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",
          "description": ""
        }
      ]
    },
    "MarketTypeName000": {
      "type": "string",
      "name": "market.type.name.000",
      "description": "Categorizes different markets run by MarketMaker",
      "oneOf": [
        {
          "const": "00000000",
          "title": "unknown",
          "description": "Default unknown"
        },
        {
```


(continues on next page)

(continued from previous page)

```
          "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",
          "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": ""
        },
```

(continues on next page)

(continued from previous page)

```
        {
          "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": "",
      "required": true
    },
    "PriceUnit": {
      "type": "string",
      "format": "MarketPriceUnit000",
      "title": "",
      "required": true
    },
    "QuantityUnit": {
      "type": "string",
      "format": "MarketQuantityUnit000",
      "title": "",
      "required": true
    },
    "SignedMarketFeeTxn": {
```

(continues on next page)

(continued from previous page)

```
      "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": ""
    }
  }
}
```

1.2.3 AtnParamsHeatpumpwithbooststore

```
{
  "gwapi": "001",
  "type_name": "atn.params.heatpumpwithbooststore",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "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": "VersantStorageHeatTariff",
          "description": ""
        },
        {
          "const": "ea5c675a",
          "title": "VersantATariff",
          "description": ""
        }
      ]
    },
    "EmitterPumpFeedbackModel000": {
      "type": "string",
      "name": "emitter.pump.feedback.model.000",
      "description": "",
      "oneOf": [
        {
          "const": "00000000",
          "title": "ConstantDeltaT",
          "description": ""
        },
        {
          "const": "f6bde4fa",
          "title": "ConstantGpm",
          "description": ""
        }
      ]
    },
    "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": {
```


(continues on next page)

(continued from previous page)

```
      "type": "string",
      "name": "recognized.temperature.unit.000",
      "description": "Unit of temperature",
      "oneOf": [
        {
          "const": "00000000",
          "title": "C",
          "description": "Celcius"
        },
        {
          "const": "6f16ee63",
          "title": "F",
          "description": "Fahrenheit"
        }
      ]
    },
    "MixingValveFeedbackModel000": {
      "type": "string",
      "name": "mixing.valve.feedback.model.000",
      "description": "Control mechanism for a mixing valve, used by Spaceheat SCADAs",
      "oneOf": [
        {
          "const": "00000000",
          "title": "ConstantSwt",
          "description": "Constant Source Water Temp"
        },
        {
          "const": "0397c1df",
          "title": "NaiveVariableSwt",
          "description": "Variable Source Water Temp, naive assumptions about␣
↪distribution system capabilities"
        },
        {
          "const": "6a668ab8",
          "title": "CautiousVariableSwt",
          "description": "Variable Source Water Temp, conservative assumptions about␣
↪distribution system capabilities"
        }
      ]
    },
    "EnergySupplyType000": {
      "type": "string",
      "name": "energy.supply.type.000",
      "description": "",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": ""
        },
        {
          "const": "cb18f937",
```

(continues on next page)

(continued from previous page)

```
          "title": "StandardOffer",
          "description": ""
        },
        {
          "const": "e9dc99a6",
          "title": "RealtimeLocalLmp",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "StoreSizeGallons": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "MaxStoreTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StoreMaxPowerKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RatedHeatpumpElectricityKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "MaxHeatpumpSourceWaterTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "SystemMaxHeatOutputSwtF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "SystemMaxHeatOutputDeltaTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "SystemMaxHeatOutputGpm": {
      "type": "number",
      "title": "",
      "required": true
    },
```

(continues on next page)

(continued from previous page)

```
    "EmitterMaxSafeSwtF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "CirculatorPumpMaxGpm": {
      "type": "number",
      "title": "",
      "required": true
    },
    "HeatpumpTariff": {
      "type": "string",
      "format": "DistributionTariff000",
      "title": "",
      "required": true
    },
    "HeatpumpEnergySupplyType": {
      "type": "string",
      "format": "EnergySupplyType000",
      "title": "",
      "required": true
    },
    "BoostTariff": {
      "type": "string",
      "format": "DistributionTariff000",
      "title": "",
      "required": true
    },
    "BoostEnergySupplyType": {
      "type": "string",
      "format": "EnergySupplyType000",
      "title": "",
      "required": true
    },
    "StandardOfferPriceDollarsPerMwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "DistributionTariffDollarsPerMwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AmbientTempStoreF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StorePassiveLossRatio": {
      "type": "number",
      "title": "",
```

(continues on next page)

(continued from previous page)


```
      "required": true
    },
    "RoomTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AmbientPowerInKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "ZeroPotentialEnergyWaterTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "EmitterPumpFeedbackModel": {
      "type": "string",
      "format": "EmitterPumpFeedbackModel000",
      "title": "",
      "required": true
    },
    "MixingValveFeedbackModel": {
      "type": "string",
      "format": "MixingValveFeedbackModel000",
      "title": "",
      "required": true
    },
    "CautiousMixingValveTempDeltaF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "Cop1TempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "Cop4TempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "CurrencyUnit": {
      "type": "string",
      "format": "RecognizedCurrencyUnit000",
      "title": "",
      "required": true
    },
    "TempUnit": {
      "type": "string",
```


(continues on next page)

(continued from previous page)

```
      "format": "RecognizedTemperatureUnit000",
      "title": "",
      "required": true
    },
    "TimezoneString": {
      "type": "string",
      "title": "",
      "required": true
    },
    "HomeCity": {
      "type": "string",
      "title": "",
      "required": true
    },
    "StorageSteps": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "FloSlices": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "SliceDurationMinutes": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "HouseWorstCaseTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AnnualHvacKwhTh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "BetaOt": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "HouseHeatingCapacity": {
      "type": "number",
      "title": "",
      "required": true
    },
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
```

(continues on next page)

(continued from previous page)

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

1.2.4 AtnParamsReportHeatpumpwithbooststore

```
{
  "gwapi": "001",
  "type_name": "atn.params.report.heatpumpwithbooststore",
  "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"
    }
  },
```

(continues on next page)

(continued from previous page)


```
  "properties": {
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "GNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "TimeUnixS": {
      "type": "integer",
      "format": "ReasonableUnixTimeS",
      "title": "",
      "required": true
    },
    "IrlTimeUnixS": {
      "type": "integer",
      "format": "ReasonableUnixTimeS",
      "title": "",
      "required": false
    },
    "AtnParams": {
      "type": "atn.params.heatpumpwithbooststore.000",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "atn.params.report.heatpumpwithbooststore.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.2.5 BaseGNodeGt


```
{
  "gwapi": "001",
  "type_name": "base.g.node.gt",
  "version": "002",
  "owner": "gridworks@gridworks-consulting.com",
  "description": ". BaseGNode. Authority is GNodeFactory.",
  "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"
    },
    "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": ""
        },
        {
          "const": "0f8872f7",
          "title": "TerminalAsset",
          "description": ""
        },
        {
          "const": "d9823442",
          "title": "AtomicTNode",
          "description": ""
        },
        {
          "const": "86f21dd2",
          "title": "MarketMaker",
          "description": ""
        },
```


(continues on next page)

(continued from previous page)


```
        {
          "const": "9521af06",
          "title": "AtomicMeteringNode",
          "description": ""
        },
        {
          "const": "4502e355",
          "title": "ConductorTopologyNode",
          "description": ""
        },
        {
          "const": "d67e564e",
          "title": "InterconnectionComponent",
          "description": ""
        },
        {
          "const": "7a8e4046",
          "title": "Scada",
          "description": ""
        }
      ]
    },
    "GNodeStatus100": {
      "type": "string",
      "name": "g.node.status.100",
      "description": "Enum for managing GNode lifecycle",
      "url": "https://gridworks.readthedocs.io/en/latest/g-node-status.html",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": "Default value"
        },
        {
          "const": "153d3475",
          "title": "Pending",
          "description": "The GNode exists but cannot be used yet."
        },
        {
          "const": "a2cfc2f7",
          "title": "Active",
          "description": "The GNode can be used."
        },
        {
          "const": "839b38db",
          "title": "PermanentlyDeactivated",
          "description": "The GNode can no longer be used, now or in the future."
        },
        {
          "const": "f5831e1d",
          "title": "Suspended",
          "description": "The GNode cannot be used, but may become active in the future."
```


(continues on next page)

(continued from previous page)

```
        }
      ]
    }
  },
  "properties": {
    "GNodeId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "Alias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "Status": {
      "type": "string",
      "format": "GNodeStatus100",
      "title": "",
      "required": true
    },
    "Role": {
      "type": "string",
      "format": "CoreGNodeRole000",
      "title": "",
      "required": true
    },
    "GNodeRegistryAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": true
    },
    "PrevAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": false
    },
    "GpsPointId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": false
    },
    "OwnershipDeedId": {
      "type": "integer",
      "minimum": 0,
      "title": "",
      "required": false
```

(continues on next page)

(continued from previous page)

```
    },
    "OwnershipDeedValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": false
    },
    "OwnerAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": false
    },
    "DaemonAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": false
    },
    "TradingRightsId": {
      "type": "integer",
      "minimum": 0,
      "title": "",
      "required": false
    },
    "ScadaAlgoAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": false
    },
    "ScadaCertId": {
      "type": "integer",
      "minimum": 0,
      "title": "",
      "required": false
    },
    "TypeName": {
      "type": "string",
      "value": "base.g.node.gt.002",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "002",
      "required": true
    }
  }
}
```

1.2.6 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="
    }
  },
  "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
    },
```


(continues on next page)

(continued from previous page)

```
    "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",
      "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."
    }
  }
}
```

1.2.7 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": ""
        },
        {
          "const": "0f8872f7",
          "title": "TerminalAsset",
          "description": ""
        },
        {
          "const": "d9823442",
          "title": "AtomicTNode",
          "description": ""
        },
        {
          "const": "86f21dd2",
          "title": "MarketMaker",
          "description": ""
        },
        {
          "const": "9521af06",
          "title": "AtomicMeteringNode",
          "description": ""
        },
```


(continues on next page)

(continued from previous page)

```
        {
          "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",
      "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
```

(continues on next page)

(continued from previous page)

```
    },
    "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
    }
  }
}
```

1.2.8 DispatchContractConfirmedHeatpumpwithbooststore

```
{
  "gwapi": "001",
  "type_name": "dispatch.contract.confirmed.heatpumpwithbooststore",
  "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␣
↪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":
```

(continues on next page)

(continued from previous page)


```
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i
↪mCnNHKfhkdYMCD5jxWejHRmPCrR8U9z/FBVsoCGbjDTTk2L1k7n/eVlumEk/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xseU
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "SignedProof": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "",
      "required": true
    },
    "AtnParams": {
      "type": "atn.params.heatpumpwithbooststore.000",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "dispatch.contract.confirmed.heatpumpwithbooststore.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.2.9 FloParamsHeatpumpwithbooststore


```
{
  "gwapi": "001",
  "type_name": "flo.params.heatpumpwithbooststore",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "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": "VersantStorageHeatTariff",
          "description": ""
        },
        {
          "const": "ea5c675a",
          "title": "VersantATariff",
          "description": ""
        }
      ]
    },
    "EmitterPumpFeedbackModel000": {
      "type": "string",
      "name": "emitter.pump.feedback.model.000",
      "description": "",
      "oneOf": [
        {
          "const": "00000000",
          "title": "ConstantDeltaT",
          "description": ""
        },
        {
          "const": "f6bde4fa",
          "title": "ConstantGpm",
          "description": ""
        }
      ]
    },
    "RecognizedCurrencyUnit000": {
      "type": "string",
      "name": "recognized.currency.unit.000",
      "description": "Unit of currency",
      "oneOf": [
```


(continues on next page)

(continued from previous page)

```
        {
          "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"
        }
      ]
    },
    "MixingValveFeedbackModel000": {
      "type": "string",
      "name": "mixing.valve.feedback.model.000",
      "description": "Control mechanism for a mixing valve, used by Spaceheat SCADAs",
      "oneOf": [
        {
          "const": "00000000",
          "title": "ConstantSwt",
          "description": "Constant Source Water Temp"
        },
        {
          "const": "0397c1df",
          "title": "NaiveVariableSwt",
          "description": "Variable Source Water Temp, naive assumptions about␣
↪distribution system capabilities"
        },
        {
          "const": "6a668ab8",
```

(continues on next page)

(continued from previous page)

```
          "title": "CautiousVariableSwt",
          "description": "Variable Source Water Temp, conservative assumptions about␣
↪distribution system capabilities"
        }
      ]
    },
    "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": {
    "StoreSizeGallons": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "MaxStoreTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StoreMaxPowerKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RatedHeatpumpElectricityKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "MaxHeatpumpSourceWaterTempF": {
      "type": "integer",
```

(continues on next page)

(continued from previous page)

```
      "title": "",
      "required": true
    },
    "SystemMaxHeatOutputSwtF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "SystemMaxHeatOutputDeltaTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "SystemMaxHeatOutputGpm": {
      "type": "number",
      "title": "",
      "required": true
    },
    "EmitterMaxSafeSwtF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "CirculatorPumpMaxGpm": {
      "type": "number",
      "title": "",
      "required": true
    },
    "HeatpumpTariff": {
      "type": "string",
      "format": "DistributionTariff000",
      "title": "",
      "required": true
    },
    "HeatpumpEnergySupplyType": {
      "type": "string",
      "format": "EnergySupplyType000",
      "title": "",
      "required": true
    },
    "BoostTariff": {
      "type": "string",
      "format": "DistributionTariff000",
      "title": "",
      "required": true
    },
    "BoostEnergySupplyType": {
      "type": "string",
      "format": "EnergySupplyType000",
      "title": "",
      "required": true
    },
```

(continues on next page)

(continued from previous page)


```
    "StandardOfferPriceDollarsPerMwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "DistributionTariffDollarsPerMwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AmbientTempStoreF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StorePassiveLossRatio": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RoomTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AmbientPowerInKw": {
      "type": "number",
      "title": "",
      "required": true
    },
    "ZeroPotentialEnergyWaterTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "EmitterPumpFeedbackModel": {
      "type": "string",
      "format": "EmitterPumpFeedbackModel000",
      "title": "",
      "required": true
    },
    "MixingValveFeedbackModel": {
      "type": "string",
      "format": "MixingValveFeedbackModel000",
      "title": "",
      "required": true
    },
    "CautiousMixingValveTempDeltaF": {
      "type": "integer",
      "title": "",
      "required": true
    },
```


(continues on next page)

(continued from previous page)

```
    "Cop1TempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "Cop4TempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "CurrencyUnit": {
      "type": "string",
      "format": "RecognizedCurrencyUnit000",
      "title": "",
      "required": true
    },
    "TempUnit": {
      "type": "string",
      "format": "RecognizedTemperatureUnit000",
      "title": "",
      "required": true
    },
    "TimezoneString": {
      "type": "string",
      "title": "",
      "required": true
    },
    "HomeCity": {
      "type": "string",
      "title": "",
      "required": true
    },
    "StorageSteps": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "HouseWorstCaseTempF": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "SystemMaxHeatOutputKwAvg": {
      "type": "number",
      "title": "",
      "required": true
    },
    "K": {
      "type": "number",
      "title": "",
      "description": "Rate at which temperature falls in the pipes, when divided by gpm",
      "required": true
```

(continues on next page)

(continued from previous page)


```
    },
    "IsRegulating": {
      "type": "boolean",
      "title": "",
      "required": true
    },
    "SliceDurationMinutes": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "PowerRequiredByHouseFromSystemAvgKwList": {
      "type": "number",
      "title": "",
      "required": true
    },
    "OutsideTempF": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RealtimeElectricityPrice": {
      "type": "number",
      "title": "",
      "required": true
    },
    "DistributionPrice": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RegulationPrice": {
      "type": "number",
      "title": "",
      "required": true
    },
    "RtElecPriceUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "WeatherUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "DistPriceUid": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
```


(continues on next page)

(continued from previous page)


```
      "required": false
    },
    "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": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartMinuteUtc": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StartingStoreIdx": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "flo.params.heatpumpwithbooststore.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.2.10 GNodeGt


```
{
  "gwapi": "001",
  "type_name": "g.node.gt",
  "version": "002",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Used to send and receive updates about GNodes. GNodes are the building␣
↪blocks of Gridworks. They have slowly-changing state that must be kept in sync across␣
↪a distributed system. Therefore, they require a global registry to act as Single␣
↪Source of Truth (SSoT). This class is used for that SSoT to share information with␣
↪actors about their GNodes, and the GNodes that they will observe and communicate with.
↪",
  "url": "https://gridworks.readthedocs.io/en/latest/g-node.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"
    },
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    }
  },
  "enums": {
    "GNodeRole000": {
      "type": "string",
      "name": "g.node.role.000",
      "description": "Categorizes GNodes by their function within GridWorks",
      "url": "https://gridworks.readthedocs.io/en/latest/g-node-role.html",
      "oneOf": [
        {
          "const": "00000000",
          "title": "GNode",
          "description": "Default value"
        },
        {
          "const": "bdeaa0b1",
          "title": "TerminalAsset",
          "url": "https://gridworks.readthedocs.io/en/latest/transactive-device.html",
          "description": "An avatar for a real-word Transactive Device"
        },
        {
          "const": "8021dcad",
          "title": "AtomicTNode",
```


(continues on next page)

(continued from previous page)

```
          "description": "Transacts in markets on behalf of, and controlling the power␣
↪use of, a TerminalAsset"
        },
        {
          "const": "304890c5",
          "title": "MarketMaker",
          "description": "Runs energy markets at its Node in the GNodeTree"
        },
        {
          "const": "8eb5b9e1",
          "title": "AtomicMeteringNode",
          "description": "Role of a GNode that will become an AtomicTNode, prior to it␣
↪owning TaTradingRights"
        },
        {
          "const": "234cfaa2",
          "title": "ConductorTopologyNode",
          "description": "An avatar for a real-world electric grid node - e.g. a␣
↪substation or transformer"
        },
        {
          "const": "fec0c127",
          "title": "InterconnectionComponent",
          "description": "An avatar for a cable or wire on the electric grid"
        },
        {
          "const": "3901c7d2",
          "title": "World",
          "description": "Adminstrative GNode responsible for managing and authorizing␣
↪instances"
        },
        {
          "const": "c499943c",
          "title": "TimeCoordinator",
          "description": "Responsible for managing time in simulations"
        },
        {
          "const": "88112a93",
          "title": "Supervisor",
          "description": "Responsible for GNode actors running in a container"
        },
        {
          "const": "674ad859",
          "title": "Scada",
          "description": "GNode associated to the device and code that directly monitors␣
↪and actuates a Transactive Device"
        },
        {
          "const": "2161739f",
          "title": "PriceService",
          "description": "Provides price forecasts for markets run by MarketMakers"
        },
```

(continues on next page)

(continued from previous page)


```
        {
          "const": "1dce1efd",
          "title": "WeatherService",
          "description": "Provides weather forecasts"
        },
        {
          "const": "db57d184",
          "title": "AggregatedTNode",
          "description": "An aggregation of AtomicTNodes"
        },
        {
          "const": "07f28817",
          "title": "Persister",
          "description": "Responsible for acking events with delivery guarantees"
        }
      ]
    },
    "GNodeStatus100": {
      "type": "string",
      "name": "g.node.status.100",
      "description": "Enum for managing GNode lifecycle",
      "url": "https://gridworks.readthedocs.io/en/latest/g-node-status.html",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": "Default value"
        },
        {
          "const": "153d3475",
          "title": "Pending",
          "description": "The GNode exists but cannot be used yet."
        },
        {
          "const": "a2cfc2f7",
          "title": "Active",
          "description": "The GNode can be used."
        },
        {
          "const": "839b38db",
          "title": "PermanentlyDeactivated",
          "description": "The GNode can no longer be used, now or in the future."
        },
        {
          "const": "f5831e1d",
          "title": "Suspended",
          "description": "The GNode cannot be used, but may become active in the future."
        }
      ]
    }
  },
  "properties": {
```


(continues on next page)

(continued from previous page)

```
    "GNodeId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "Immutable identifier for GNode",
      "required": true
    },
    "Alias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "Structured mutable identifier for GNode",
      "description": "The GNode Aliases are used for organizing how actors in Gridworks␣
↪communicate. Together, they also encode the known topology of the electric grid.",
      "required": true
    },
    "Status": {
      "type": "string",
      "format": "GNodeStatus100",
      "title": "Lifecycle indicator",
      "required": true
    },
    "Role": {
      "type": "string",
      "format": "GNodeRole000",
      "title": "Role within Gridworks",
      "required": true
    },
    "GNodeRegistryAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "Algorand address for GNodeRegistry",
      "description": "For actors in a Gridworks world, the GNodeRegistry is the Single␣
↪Source of Truth for existence and updates to GNodes.",
      "required": true
    },
    "PrevAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "Previous GNodeAlias",
      "description": "As the topology of the grid updates, GNodeAliases will change to␣
↪reflect that. This may happen a handful of times over the life of a GNode.",
      "required": false
    },
    "GpsPointId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "Lat/lon of GNode",
      "description": "Some GNodes, in particular those acting as avatars for physical␣
↪devices that are part of or are attached to the electric grid, have physical locations.
↪ These locations are used to help validate the grid topology.",
      "required": false
    },
    "OwnershipDeedId": {
```

(continues on next page)

(continued from previous page)

```
      "type": "integer",
      "minimum": 0,
      "title": "Algorand Id of ASA Deed",
      "description": "The Id of the TaDeed Algorand Standard Asset if the GNode is a␣
↪TerminalAsset.",
      "required": false
    },
    "OwnershipDeedValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "Algorand address of Validator",
      "description": "Deeds are issued by the GNodeFactory, in partnership with third␣
↪party Validators.",
      "required": false
    },
    "OwnerAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "Algorand address of the deed owner",
      "required": false
    },
    "DaemonAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "Algorand address of the daemon app",
      "description": "Some GNodes have Daemon applications associated to them to handle␣
↪blockchain operations.",
      "required": false
    },
    "TradingRightsId": {
      "type": "integer",
      "minimum": 0,
      "title": "Algorand Id of ASA TradingRights",
      "description": "The Id of the TradingRights Algorand Standard Asset.",
      "required": false
    },
    "ScadaAlgoAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "",
      "required": false
    },
    "ScadaCertId": {
      "type": "integer",
      "minimum": 0,
      "title": "",
      "required": false
    },
    "ComponentId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "Unique identifier for GNode's Component",
```

(continues on next page)

(continued from previous page)

```
      "description": "Used if a GNode is an avatar for a physical device. The serial␣
↪number of a device is different from its make/model. The ComponentId captures the␣
↪specific instance of the device.",
      "required": false
    },
    "DisplayName": {
      "type": "string",
      "title": "Display Name",
      "required": false
    },
    "TypeName": {
      "type": "string",
      "value": "g.node.gt.002",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "002",
      "required": true
    }
  },
  "example": {
    "GNodeId": "575f374f-8533-4733-baf7-91146c607445",
    "Alias": "d1.isone.ver.keene",
    "StatusGtEnumSymbol": "a2cfc2f7",
    "RoleGtEnumSymbol": "234cfaa2",
    "GNodeRegistryAddr": "MONSDN5MXG4VMIOHJNCJJBVASG7HEZQSCEIKJAPEPVI5ZJUMQGXQKSOAYU",
    "TypeName": "g.node.gt",
    "Version": "000"
  }
}
```

1.2.11 GNodeInstanceGt

```
{
  "gwapi": "001",
  "type_name": "g.node.instance.gt",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Used to send and receive updates about GNodeInstances. One of the␣
↪layers of abstraction connecting a GNode with a running app in a Docker container.",
  "url": "https://gridworks.readthedocs.io/en/latest/g-node-instance.html",
  "formats": {
    "ReasonableUnixTimeS": {
      "type": "string",
      "description": "Integer reflecting unix time seconds between 1970 and 3000",
      "example": ""
    },
    "UuidCanonicalTextual": {
```

(continues on next page)

(continued from previous page)

```
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    }
  },
  "enums": {
    "GniStatus000": {
      "type": "string",
      "name": "gni.status.000",
      "description": "Enum for managing GNodeInstance lifecycle",
      "url": "https://gridworks.readthedocs.io/en/latest/g-node-instance.html",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": "Default Value"
        },
        {
          "const": "7890ab0a",
          "title": "Pending",
          "description": "Has been created by the World, but has not yet finished␣
↪provisioning"
        },
        {
          "const": "69241259",
          "title": "Active",
          "description": "Active in its GridWorks world. If the GNodeInstance has an␣
↪actor, that actor is communicating"
        },
        {
          "const": "8222421f",
          "title": "Done",
          "description": "No longer represents the GNode."
        }
      ]
    },
    "StrategyName000": {
      "type": "string",
      "name": "strategy.name.000",
      "description": "Used to assign code to run a particular GNodeInstance",
      "oneOf": [
        {
          "const": "00000000",
          "title": "NoActor",
          "description": "Assigned to GNodes that do not have actors"
        },
        {
```

(continues on next page)

(continued from previous page)

```
          "const": "642c83d3",
          "title": "WorldA",
          "description": "Basic GridWorks world strategy"
        },
        {
          "const": "4bb2cf7e",
          "title": "SupervisorA",
          "description": "Supervisor A Strategy, in [gridworks-atn package](https://pypi.
↪org/project/gridworks-atn/)"
        },
        {
          "const": "f5961401",
          "title": "AtnHeatPumpWithBoostStore",
          "description": "GridWorks strategy for an AtomicTNode representing a type of␣
↪heat pump storage heating system"
        },
        {
          "const": "73fbe6ab",
          "title": "TcGlobalA",
          "description": "Basic GridWorks TimeCoordinator strategy, in [gridworks-
↪timecoordinator repo](https://github.com/thegridelectric/gridworks-timecoordinator)"
        },
        {
          "const": "5e18a52e",
          "title": "MarketMakerA",
          "description": "Simple GridWorks MarketMaker strategy, in [gridworks-
↪marketmaker repo](https://github.com/thegridelectric/gridworks-marketmaker)"
        }
      ]
    }
  },
  "properties": {
    "GNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "Immutable identifier for GNodeInstance (Gni)",
      "required": true
    },
    "GNode": {
      "type": "g.node.gt.002",
      "title": "The GNode represented by the Gni",
      "required": true
    },
    "Strategy": {
      "type": "string",
      "format": "StrategyName000",
      "title": "Used to determine the code running in a GNode actor application",
      "required": true
    },
    "Status": {
      "type": "string",
      "format": "GniStatus000",
```

(continues on next page)

(continued from previous page)

```
      "title": "Lifecycle Status for Gni",
      "required": true
    },
    "SupervisorContainerId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "The Id of the docker container where the Gni runs",
      "required": true
    },
    "StartTimeUnixS": {
      "type": "integer",
      "format": "ReasonableUnixTimeS",
      "title": "When the gni starts representing the GNode",
      "description": "Specifically, when the Status changes from Pending to Active. Note␣
↪that this is time in the GNode's World, which may not be real time if it is a␣
↪simulation.",
      "required": true
    },
    "EndTimeUnixS": {
      "type": "integer",
      "title": "When the gni stops representing the GNode",
      "description": "Specifically, when the Status changes from Active to Done.",
      "required": true
    },
    "AlgoAddress": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "Algorand address for Gni",
      "required": false
    },
    "TypeName": {
      "type": "string",
      "value": "g.node.instance.gt.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.2.12 GtDispatchBoolean

```
{
  "gwapi": "001",
  "type_name": "gt.dispatch.boolean",
  "version": "110",
  "owner": "gridworks@gridworks-consulting.com",
  "description": ". Boolean dispatch command sent over the cloud broker from one GNode␣
↪(FromGNodeAlias/ FromGNodeId) to another (ToGNodeAlias). AboutNodeAlias is the node␣
↪getting dispatched. It may be a GNode, but it can also be a SpaceHeatNode.",
  "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"
    },
    "ReasonableUnixTimeMs": {
      "type": "string",
      "description": "An integer reflecting unix time in ms between midnight Jan 1 2000␣
↪and midnight Jan 1 3000 UTC",
      "example": ""
    }
  },
  "properties": {
    "AboutNodeName": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "The Spaceheat Node getting dispatched",
      "required": true
    },
    "ToGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "GNodeAlias of the SCADA",
      "required": true
    },
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "GNodeAlias of AtomicTNode",
      "required": true
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "GNodeInstance of the AtomicTNode",
      "required": true
```

(continues on next page)

(continued from previous page)

```
    },
    "RelayState": {
      "type": "integer",
      "title": "0 or 1",
      "required": true
    },
    "SendTimeUnixMs": {
      "type": "integer",
      "format": "ReasonableUnixTimeMs",
      "title": "Time the AtomicTNode sends the dispatch, by its clock",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "gt.dispatch.boolean.110",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "110",
      "required": true
    }
  }
}
```

1.2.13 GwCertId

```
{
  "gwapi": "001",
  "type_name": "gw.cert.id",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Clarifies whether cert id is an Algorand Standard Asset or SmartSig",
  "formats": {
    "AlgoAddressStringFormat": {
      "type": "string",
      "description": "String of length 32, characters are all base32 digits.",
      "example": "RNMHG32VTIHTC7W3LZOEPTDGREL5IQGK46HKD3KBLZHYQUCAKLMT4G5ALI"
    }
  },
  "enums": {
    "AlgoCertType000": {
      "type": "string",
      "name": "algo.cert.type.000",
      "description": "Used to distinguish ASA vs SmartSignature certificates",
      "oneOf": [
        {
          "const": "00000000",
          "title": "ASA",
```

(continues on next page)

(continued from previous page)

```
          "description": "Certificate based on Algorand Standard Asset"
        },
        {
          "const": "086b5165",
          "title": "SmartSig",
          "description": "Certificate based on Algorand Smart Signature"
        }
      ]
    }
  },
  "properties": {
    "Type": {
      "type": "string",
      "format": "AlgoCertType000",
      "title": "",
      "required": true
    },
    "Idx": {
      "type": "integer",
      "minimum": 0,
      "title": "ASA Index",
      "required": false
    },
    "Addr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "Algorand Smart Signature Address",
      "required": false
    },
    "TypeName": {
      "type": "string",
      "value": "gw.cert.id.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  },
  "axioms": {
    "Axiom1": {
      "title": "Cert type consistency",
      "description": "If Type is ASA, then Id exists and Addr does not. Otherwise, Addr␣
↪exists and Id does not."
    }
  },
  "example": {
    "TypeGtEnumSymbol": "00000000",
    "Idx": 14,
    "TypeName": "gw.cert.id",
```

(continues on next page)

(continued from previous page)

```
    "Version": "000"
  }
}
```

1.2.14 HeartbeatA

```
{
  "gwapi": "001",
  "type_name": "heartbeat.a",
  "version": "100",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Used to check that an actor can both send and receive messages.␣
↪Payload for direct messages sent back and forth between actors, for example a␣
↪Supervisor and one of its subordinates.",
  "url": "https://gridworks.readthedocs.io/en/latest/g-node-instance.html",
  "formats": {
    "HexChar": {
      "type": "string",
      "description": "single-char string in '0123456789abcdefABCDEF'",
      "example": "d"
    }
  },
  "properties": {
    "MyHex": {
      "type": "string",
      "format": "HexChar",
      "title": "Hex character getting sent",
      "required": true
    },
    "YourLastHex": {
      "type": "string",
      "format": "HexChar",
      "title": "Last hex character received from heartbeat partner",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "heartbeat.a.100",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "100",
      "required": true
    }
  }
}
```

1.2.15 HeartbeatAlgoAudit


```
{
  "gwapi": "001",
  "type_name": "heartbeat.algo.audit",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": ". Algo payload with report of last HeartbeatB sent to partner via␣
↪Rabbit, to be sent to DispatchContract on Algo blockchain",
  "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": {
    "SignedProof": {
      "type": "string",
      "format": "AlgoMsgPackEncoded",
      "title": "Tiny signed payment to DispatchContract to prove identity",
      "description": "Can be a minimal payment, as long as it comes from the AtomicTNode␣
↪or SCADA.",
      "required": true
    },
    "Heartbeat": {
      "type": "heartbeat.b.000",
      "title": "Heartbeat sender last sent to its partner",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "heartbeat.algo.audit.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
```


(continues on next page)

(continued from previous page)

```
    }
  }
}
```

1.2.16 HeartbeatB

```
{
  "gwapi": "001",
  "type_name": "heartbeat.b",
  "version": "001",
  "owner": "gridworks@gridworks-consulting.com",
  "description": ". Heartbeat for Scada-AtomicTNode DispatchContract, to send via␣
↪RabbitMQ",
  "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"
    },
    "HexChar": {
      "type": "string",
      "description": "single-char string in '0123456789abcdefABCDEF'",
      "example": "d"
    },
    "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"
    },
    "ReasonableUnixTimeMs": {
      "type": "string",
      "description": "An integer reflecting unix time in ms between midnight Jan 1 2000␣
↪and midnight Jan 1 3000 UTC",
      "example": ""
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "My GNodeAlias",
      "required": true
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "My GNodeInstanceId",
      "required": true
    },
```

(continues on next page)

(continued from previous page)

```
    "MyHex": {
      "type": "string",
      "format": "HexChar",
      "title": "Hex character getting sent",
      "required": true
    },
    "YourLastHex": {
      "type": "string",
      "format": "HexChar",
      "title": "Last hex character received from heartbeat partner. If the heartbeat␣
↪initiator wants to start the sequence over, it does not include this.",
      "required": false
    },
    "LastReceivedTimeUnixMs": {
      "type": "integer",
      "format": "ReasonableUnixTimeMs",
      "title": "Time YourLastHex was received on my clock",
      "required": true
    },
    "SendTimeUnixMs": {
      "type": "integer",
      "format": "ReasonableUnixTimeMs",
      "title": "Time this message is made and sent on my clock",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "heartbeat.b.001",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "001",
      "required": true
    }
  }
}
```

1.2.17 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": ""
    }
  }
}
```

1.2.18 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."
    }
  }
}
```

1.2.19 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."
    }
  }
}
```

1.2.20 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."
    }
  }
}
```

1.2.21 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
    }
  }
}
```

1.2.22 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
    }
  }
}
```

1.2.23 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
    }
  }
}
```

1.2.24 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)

```
}
```

1.2.25 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
    }
  }
}
```


1.2.26 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
    }
  }
}
```

1.2.27 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
    }
  }
}
```

1.2.28 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
    }
  }
}
```

1.2.29 Ready


```
{
  "gwapi": "001",
  "type_name": "ready",
  "version": "001",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Used in simulations by TimeCoordinator GNodes. Only intended for␣
↪simulations that do not have sub-second TimeSteps. TimeCoordinators based on␣
↪```gridworks-timecoordinator``` have a notion of actors whose `Ready` must be received␣
↪before issuing the next TimeStep.",
  "url": "https://gridworks.readthedocs.io/en/latest/time-coordinator.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"
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "The GNodeAlias of the sender",
      "required": true
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "The GNodeInstanceId of the sender",
      "required": true
    },
    "TimeUnixS": {
      "type": "integer",
      "title": "Latest simulated time for sender",
      "description": "The time in unix seconds of the latest TimeStep received from the␣
↪TimeCoordinator by the actor that sent the payload.",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "ready.001",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
```


(continues on next page)

(continued from previous page)

```
      "default": "001",
      "required": true
    }
  }
}
```

1.2.30 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/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xse
↪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.",
```

(continues on next page)

(continued from previous page)

```
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "scada.cert.transfer.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "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."
    }
  }
}
```

1.2.31 SimScadaDriverReport

```
{
  "gwapi": "001",
  "type_name": "sim.scada.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
```

(continues on next page)

(continued from previous page)


```
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "",
      "required": true
    },
    "BoostPowerKwTimes1000": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "HeatpumpPowerKwTimes1000": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StoreKwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "CopTimes10": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "MaxStoreKwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "sim.scada.driver.report.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.2.32 SimTimestep


```
{
  "gwapi": "001",
  "type_name": "sim.timestep",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Sent by TimeCoordinators to coordinate time. For simulated actors,␣
↪time progresses discretely on receipt of these time steps.",
  "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"
    },
    "ReasonableUnixTimeMs": {
      "type": "string",
      "description": "An integer reflecting unix time in ms between midnight Jan 1 2000␣
↪and midnight Jan 1 3000 UTC",
      "example": ""
    }
  },
  "properties": {
    "FromGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "The GNodeAlias of the sender",
      "description": "The sender should always be a GNode Actor of role TimeCoordinator.
↪",
      "required": true
    },
    "FromGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "The GNodeInstanceId of the sender",
      "required": true
    },
    "TimeUnixS": {
      "type": "integer",
      "format": "ReasonableUnixTimeS",
      "title": "Current time in unix seconds",
      "required": true
```


(continues on next page)

(continued from previous page)

```
    },
    "TimestepCreatedMs": {
      "type": "integer",
      "format": "ReasonableUnixTimeMs",
      "title": "The real time created, in unix milliseconds",
      "required": true
    },
    "MessageId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "MessageId",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "sim.timestep.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.2.33 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
    },
```

(continues on next page)

(continued from previous page)

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

1.2.34 SnapshotHeatpumpwithbooststore

```
{
  "gwapi": "001",
  "type_name": "snapshot.heatpumpwithbooststore",
  "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
    },
    "BoostPowerKwTimes1000": {
      "type": "integer",
```

(continues on next page)

(continued from previous page)


```
      "title": "",
      "required": true
    },
    "HeatpumpPowerKwTimes1000": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "StoreKwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "CopTimes10": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "MaxStoreKwh": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "AboutTerminalAssetAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "snapshot.heatpumpwithbooststore.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.2.35 SuperStarter


```
{
  "gwapi": "001",
  "type_name": "super.starter",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Used by world to seed a docker container with data needed to spawn and␣
↪superviser GNodeInstances",
  "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": {
    "SupervisorContainer": {
      "type": "supervisor.container.gt.000",
      "title": "Key data about the docker container",
      "required": true
    },
    "GniList": {
      "type": "g.node.instance.gt.000",
      "title": "List of GNodeInstances (Gnis) run in the container",
      "required": true
    },
    "AliasWithKeyList": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "Aliases of Gnis that own Algorand secret keys",
      "required": true
    },
    "KeyList": {
      "type": "string",
      "title": "Algorand secret keys owned by Gnis",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "super.starter.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.2.36 SupervisorContainerGt


```
{
  "gwapi": "001",
  "type_name": "supervisor.container.gt",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "Used to send and receive updates about SupervisorContainers. Sent from␣
↪a GNodeRegistry to a World, and used also by the World as it spawns GNodeInstances in␣
↪docker instances (i.e., the SupervisorContainers).",
  "url": "https://gridworks.readthedocs.io/en/latest/supervisor.html",
  "formats": {
    "WorldInstanceNameFormat": {
      "type": "string",
      "description": "AlphanumericString + '__' + Integer",
      "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"
    }
  },
  "enums": {
    "SupervisorContainerStatus000": {
      "type": "string",
      "name": "supervisor.container.status.000",
      "description": "Manages lifecycle of the docker containers where GridWorks actors␣
↪run",
      "oneOf": [
        {
          "const": "00000000",
          "title": "Unknown",
          "description": "Default value"
        },
        {
          "const": "f48cff43",
          "title": "Authorized",
          "description": "World has created the information for starting the container"
        },
        {
          "const": "17c5cc54",
          "title": "Launching",
          "description": "World has launched the container"
        },
        {
          "const": "ec342324",
```


(continues on next page)

(continued from previous page)

```
          "title": "Provisioning",
          "description": "Container has started, but is going through its provisioning␣
↪process"
        },
        {
          "const": "cfde1b40",
          "title": "Running",
          "description": "GNode actors in the container are active"
        },
        {
          "const": "4e28b6ae",
          "title": "Stopped",
          "description": "Stopped"
        },
        {
          "const": "da2dafe0",
          "title": "Deleted",
          "description": "Deleted"
        }
      ]
    }
  },
  "properties": {
    "SupervisorContainerId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "Id of the docker SupervisorContainer",
      "required": true
    },
    "Status": {
      "type": "string",
      "format": "SupervisorContainerStatus000",
      "title": "",
      "required": true
    },
    "WorldInstanceName": {
      "type": "string",
      "format": "WorldInstanceNameFormat",
      "title": "Name of the WorldInstance",
      "description": "For example, d1__1 is a potential name for a World whose World␣
↪GNode has alias d1.",
      "required": true
    },
    "SupervisorGNodeInstanceId": {
      "type": "string",
      "format": "UuidCanonicalTextual",
      "title": "Id of the SupervisorContainer's prime actor (aka the Supervisor GNode)",
      "required": true
    },
    "SupervisorGNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
```

(continues on next page)

(continued from previous page)

```
      "title": "Alias of the SupervisorContainer's prime actor (aka the Supervisor GNode)
↪",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "supervisor.container.gt.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.2.37 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
```

(continues on next page)

(continued from previous page)

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

1.2.38 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="
    }
  },
```

(continues on next page)

(continued from previous page)

```
  "properties": {
    "ValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "The address of the pending TaValidator",
      "required": true
    },
    "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": {
```

(continues on next page)

(continued from previous page)


```
    "ValidatorAddr": "7QQT4GN3ZPAQEFCNWF5BMF7NULVK3CWICZVT4GM3BQRISD52YEDLWJ4MII",
    "HalfSignedCertCreationMtx":
↪"gqRtc2lng6ZzdWJzaWeSgaJwa8Qgi1hzb1WaDzF+215cR8xmiRfUQMrnjqHtQV5PiFBAUtmConBrxCD8IT4Zu8vBAhRNsXoWF+2i(
↪lXB23O98qsyuf7f9jqDu+WT2U/
↪KB53CPR+XSUWGh5nonEUdp63TDIEo3RocgKhdgGjdHhuiaRhcGFyhKJhbq9Nb2xseSBNZXRlcm1haWShbcQgi1hzb1WaDzF+215cR8
↪",
    "TypeName": "tavalidatorcert.algo.create",
    "Version": "000"
  }
}
```


1.2.39 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/
↪M1KSe48Jo3RocgKhdgGjdHhuiaRhcGFyhaJhbq9Nb2xseSBNZXRlcm1haWSiYXXZKWh0dHA6Ly9sb2NhbGhvc3Q6NTAwMC9tb2xseU
↪iF+bI4LU6UTgG4SIxyD10PS0/
↪vNAEa93OC5SVRFn6omx2zQQ5pG5vdGXEK01vbGx5IEluYyBUZWxlbWV0cnkgU3VydmV5b3JzIGFuZCBQdXJ2ZXlvcnOjc25kxCDHZ
↪H5mIku1s4ulDm3EmU6dYKXCWEB6R0eXBlpGFjZmc="
    }
  },
  "properties": {
    "ValidatorAddr": {
      "type": "string",
      "format": "AlgoAddressStringFormat",
      "title": "The address of the pending TaValidator",
      "required": true
    },
    "HalfSignedCertTransferMtx": {
      "type": "string",
```


(continues on next page)

(continued from previous page)

```
      "format": "AlgoMsgPackEncoded",
      "title": "Algo multi-transaction for certificate transfer, with 1 of 2 signatures",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "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."
    }
  }
}
```

1.2.40 TerminalassetCertifyHack


```
{
  "gwapi": "001",
  "type_name": "terminalasset.certify.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"
    },
    "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",
```


(continues on next page)

(continued from previous page)

```
      "required": true
    }
  }
}
```

1.3 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.

1.4 GridWorks DataClasses

GNode DataClass Definition

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

GNode DataClass Definition

`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

GpsPoint Definition

1.5 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
- MarketMaker: Runs energy markets at its Node in the GNodeTree
- 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
- TimeCoordinator: Responsible for managing time in simulations
- Supervisor: Responsible for GNode actors running in a container
- 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: Basic GridWorks world strategy
- SupervisorA: Supervisor A Strategy, in [gridworks-atn package](https://pypi.org/project/gridworks-atn/)
- AtnHeatPumpWithBoostStore: GridWorks strategy for an AtomicTNode representing a type of heat pump storage heating system
- TcGlobalA: Basic GridWorks TimeCoordinator strategy, in [gridworks-timecoordinator repo](https://github.com/thegridelectric/gridworks-timecoordinator)
- MarketMakerA: Simple GridWorks MarketMaker strategy, in [gridworks-marketmaker repo](https://github.com/thegridelectric/gridworks-marketmaker)

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]

1.6 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

1.6.1 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

1.6.2 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

1.6.3 AtnParamsHeatpumpwithbooststore

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

class gwatn.types.**AtnParamsHeatpumpwithbooststore**(*, *StoreSizeGallons=240*, *MaxStoreTempF=190*, *StoreMaxPowerKw=13.5*, *RatedHeatpumpElectricityKw=5.5*, *MaxHeatpumpSourceWaterTempF=135*, *SystemMaxHeatOutputSwtF=160*, *SystemMaxHeatOutputDeltaTempF=20*, *SystemMaxHeatOutputGpm=4*, *EmitterMaxSafeSwtF=170*, *CirculatorPumpMaxGpm=12*, *HeatpumpTariff=DistributionTariff.VersantStorageHeatTariff*, *HeatpumpEnergySupplyType=EnergySupplyType.RealtimeLocalLmp*, *BoostTariff=DistributionTariff.VersantStorageHeatTariff*, *BoostEnergySupplyType=EnergySupplyType.RealtimeLocalLmp*, *StandardOfferPriceDollarsPerMwh=110*, *DistributionTariffDollarsPerMwh=113*, *AmbientTempStoreF=65*, *StorePassiveLossRatio=0.001*, *RoomTempF=70*, *AmbientPowerInKw=1.25*, *ZeroPotentialEnergyWaterTempF=100*, *EmitterPumpFeedbackModel=EmitterPumpFeedbackModel.ConstantDeltaT*, *MixingValveFeedbackModel=MixingValveFeedbackModel.NaiveVariableSwt*, *CautiousMixingValveTempDeltaF=5*, *Cop1TempF=-20*, *Cop4TempF=70*, *CurrencyUnit=RecognizedCurrencyUnit.USD*, *TempUnit=RecognizedTemperatureUnit.F*, *TimezoneString='US/Eastern'*, *HomeCity='MILLINOCKET_ME'*, *StorageSteps=100*, *FloSlices*, *SliceDurationMinutes*, *HouseWorstCaseTempF=-7*, *AnnualHvacKwhTh=28125*, *BetaOt=158*, *HouseHeatingCapacity=4*, *GNodeAlias*, *GNodeInstanceId*, *TypeName='atn.params.heatpumpwithbooststore'*, *Version='000'*)

Parameters

- **StoreSizeGallons** (*int*) –
- **MaxStoreTempF** (*int*) –
- **StoreMaxPowerKw** (*float*) –
- **RatedHeatpumpElectricityKw** (*float*) –
- **MaxHeatpumpSourceWaterTempF** (*int*) –
- **SystemMaxHeatOutputSwtF** (*int*) –

- **SystemMaxHeatOutputDeltaTempF** (*int*) –
- **SystemMaxHeatOutputGpm** (*float*) –
- **EmitterMaxSafeSwtF** (*int*) –
- **CirculatorPumpMaxGpm** (*float*) –
- **HeatpumpTariff** (*DistributionTariff*) –
- **HeatpumpEnergySupplyType** (*EnergySupplyType*) –
- **BoostTariff** (*DistributionTariff*) –
- **BoostEnergySupplyType** (*EnergySupplyType*) –
- **StandardOfferPriceDollarsPerMwh** (*int*) –
- **DistributionTariffDollarsPerMwh** (*int*) –
- **AmbientTempStoreF** (*int*) –
- **StorePassiveLossRatio** (*float*) –
- **RoomTempF** (*int*) –
- **AmbientPowerInKw** (*float*) –
- **ZeroPotentialEnergyWaterTempF** (*int*) –
- **EmitterPumpFeedbackModel** (*EmitterPumpFeedbackModel*) –
- **MixingValveFeedbackModel** (*MixingValveFeedbackModel*) –
- **CautiousMixingValveTempDeltaF** (*int*) –
- **Cop1TempF** (*int*) –
- **Cop4TempF** (*int*) –
- **CurrencyUnit** (*RecognizedCurrencyUnit*) –
- **TempUnit** (*RecognizedTemperatureUnit*) –
- **TimezoneString** (*str*) –
- **HomeCity** (*str*) –
- **StorageSteps** (*int*) –
- **FloSlices** (*int*) –
- **SliceDurationMinutes** (*int*) –
- **HouseWorstCaseTempF** (*int*) –
- **AnnualHvacKwhTh** (*int*) –
- **BetaOt** (*int*) –
- **HouseHeatingCapacity** (*float*) –
- **GNodeAlias** (*str*) –
- **GNodeInstanceId** (*str*) –
- **TypeName** (*Literal['atn.params.heatpumpwithbooststore']*) –
- **Version** (*str*) –

StoreSizeGallons:

- Description:

MaxStoreTempF:

- Description:

StoreMaxPowerKw:

- Description:

RatedHeatpumpElectricityKw:

- Description:

MaxHeatpumpSourceWaterTempF:

- Description:

SystemMaxHeatOutputSwtF:

- Description:

SystemMaxHeatOutputDeltaTempF:

- Description:

SystemMaxHeatOutputGpm:

- Description:

EmitterMaxSafeSwtF:

- Description:

CirculatorPumpMaxGpm:

- Description:

HeatpumpTariff:

- Description:

HeatpumpEnergySupplyType:

- Description:

BoostTariff:

- Description:

BoostEnergySupplyType:

- Description:

StandardOfferPriceDollarsPerMwh:

- Description:

DistributionTariffDollarsPerMwh:

- Description:

AmbientTempStoreF:

- Description:

StorePassiveLossRatio:

- Description:

RoomTempF:

- Description:

AmbientPowerInKw:

- Description:

ZeroPotentialEnergyWaterTempF:

- Description:

EmitterPumpFeedbackModel:

- Description:

MixingValveFeedbackModel:

- Description:

CautiousMixingValveTempDeltaF:

- Description:

Cop1TempF:

- Description:

Cop4TempF:

- Description:

CurrencyUnit:

- Description:

TempUnit:

- Description:

TimezoneString:

- Description:

HomeCity:

- Description:

StorageSteps:

- Description:

FloSlices:

- Description:

SliceDurationMinutes:

- Description:

HouseWorstCaseTempF:

- Description:

AnnualHvacKwhTh:

- Description:

BetaOt:

- Description:

HouseHeatingCapacity:

- Description:

GNodeAlias:

- Description:
- Format: LeftRightDot

GNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

class gwatn.types.**AtnParamsHeatpumpwithbooststore_Maker**(*store_size_gallons*, *max_store_temp_f*, *store_max_power_kw*, *rated_heatpump_electricity_kw*, *max_heatpump_source_water_temp_f*, *system_max_heat_output_swt_f*, *system_max_heat_output_delta_temp_f*, *system_max_heat_output_gpm*, *emitter_max_safe_swt_f*, *circulator_pump_max_gpm*, *heatpump_tariff*, *heatpump_energy_supply_type*, *boost_tariff*, *boost_energy_supply_type*, *standard_offer_price_dollars_per_mwh*, *distribution_tariff_dollars_per_mwh*, *ambient_temp_store_f*, *store_passive_loss_ratio*, *room_temp_f*, *ambient_power_in_kw*, *zero_potential_energy_water_temp_f*, *emitter_pump_feedback_model*, *mixing_valve_feedback_model*, *cautious_mixing_valve_temp_delta_f*, *cop1_temp_f*, *cop4_temp_f*, *currency_unit*, *temp_unit*, *timezone_string*, *home_city*, *storage_steps*, *flo_slices*, *slice_duration_minutes*, *house_worst_case_temp_f*, *annual_hvac_kwh_th*, *beta_ot*, *house_heating_capacity*, *g_node_alias*, *g_node_instance_id*)

Parameters

- **store_size_gallons** (*int*) –
- **max_store_temp_f** (*int*) –
- **store_max_power_kw** (*float*) –
- **rated_heatpump_electricity_kw** (*float*) –
- **max_heatpump_source_water_temp_f** (*int*) –
- **system_max_heat_output_swt_f** (*int*) –
- **system_max_heat_output_delta_temp_f** (*int*) –
- **system_max_heat_output_gpm** (*float*) –

- **emitter_max_safe_swt_f** (*int*) –
- **circulator_pump_max_gpm** (*float*) –
- **heatpump_tariff** (*DistributionTariff*) –
- **heatpump_energy_supply_type** (*EnergySupplyType*) –
- **boost_tariff** (*DistributionTariff*) –
- **boost_energy_supply_type** (*EnergySupplyType*) –
- **standard_offer_price_dollars_per_mwh** (*int*) –
- **distribution_tariff_dollars_per_mwh** (*int*) –
- **ambient_temp_store_f** (*int*) –
- **store_passive_loss_ratio** (*float*) –
- **room_temp_f** (*int*) –
- **ambient_power_in_kw** (*float*) –
- **zero_potential_energy_water_temp_f** (*int*) –
- **emitter_pump_feedback_model** (*EmitterPumpFeedbackModel*) –
- **mixing_valve_feedback_model** (*MixingValveFeedbackModel*) –
- **cautious_mixing_valve_temp_delta_f** (*int*) –
- **cop1_temp_f** (*int*) –
- **cop4_temp_f** (*int*) –
- **currency_unit** (*RecognizedCurrencyUnit*) –
- **temp_unit** (*RecognizedTemperatureUnit*) –
- **timezone_string** (*str*) –
- **home_city** (*str*) –
- **storage_steps** (*int*) –
- **flo_slices** (*int*) –
- **slice_duration_minutes** (*int*) –
- **house_worst_case_temp_f** (*int*) –
- **annual_hvac_kwh_th** (*int*) –
- **beta_ot** (*int*) –
- **house_heating_capacity** (*float*) –
- **g_node_alias** (*str*) –
- **g_node_instance_id** (*str*) –

1.6.4 AtnParamsReportHeatpumpwithbooststore

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

class gwatn.types.**AtnParamsReportHeatpumpwithbooststore**(*, *GNodeAlias*, *GNodeInstanceId*, *TimeUnixS*, *IrlTimeUnixS=None*, *AtnParams*, *TypeName='atn.params.report.heatpumpwithbooststore'*, *Version='000'*)

AtomicTNode reporting its AtnParams.

Parameters like the size of the thermal store.

Parameters

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

GNodeAlias:

- Description:
- Format: LeftRightDot

GNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

TimeUnixS:

- Description:
- Format: ReasonableUnixTimeS

IrlTimeUnixS:

- Description:
- Format: ReasonableUnixTimeS

AtnParams:

- Description:

class gwatn.types.atn_params_report_heatpumpwithbooststore.**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_heatpumpwithbooststore.**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_heatpumpwithbooststore.**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.**AtnParamsReportHeatpumpwithbooststore_Maker**(*g_node_alias*, *g_node_instance_id*, *time_unix_s*, *irl_time_unix_s*, *atn_params*)

Parameters

- **g_node_alias** (*str*) –
- **g_node_instance_id** (*str*) –
- **time_unix_s** (*int*) –
- **irl_time_unix_s** (*int | None*) –
- **atn_params** (AtnParamsHeatpumpwithbooststore) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (AtnParamsReportHeatpumpwithbooststore) –

Return type
str

classmethod **type_to_tuple**(*t*)

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

Parameters
t (*str*) –

Return type
AtnParamsReportHeatpumpwithbooststore

1.6.5 BaseGNodeGt

Python pydantic class corresponding to json type `base.g.node.gt`.

class gwatn.types.**BaseGNodeGt**(*, *GNodeId*, *Alias*, *Status*, *Role*, *GNodeRegistryAddr*, *PrevAlias=None*, *GpsPointId=None*, *OwnershipDeedId=None*, *OwnershipDeedValidatorAddr=None*, *OwnerAddr=None*, *DaemonAddr=None*, *TradingRightsId=None*, *ScadaAlgoAddr=None*, *ScadaCertId=None*, *TypeName='base.g.node.gt'*, *Version='002'*)

.

BaseGNode. Authority is GNodeFactory.

Parameters

- **GNodeId** (*str*) –
- **Alias** (*str*) –
- **Status** (*GNodeStatus*) –
- **Role** (*CoreGNodeRole*) –
- **GNodeRegistryAddr** (*str*) –
- **PrevAlias** (*str | None*) –
- **GpsPointId** (*str | None*) –
- **OwnershipDeedId** (*int | None*) –
- **OwnershipDeedValidatorAddr** (*str | None*) –
- **OwnerAddr** (*str | None*) –
- **DaemonAddr** (*str | None*) –
- **TradingRightsId** (*int | None*) –
- **ScadaAlgoAddr** (*str | None*) –
- **ScadaCertId** (*int | None*) –
- **TypeName** (*Literal['base.g.node.gt']*) –
- **Version** (*str*) –

GNodeId:

- Description:
- Format: UuidCanonicalTextual

Alias:

- Description:
- Format: LeftRightDot

Status:

- Description:

Role:

- Description:

GNodeRegistryAddr:

- Description:
- Format: AlgoAddressStringFormat

PrevAlias:

- Description:
- Format: LeftRightDot

GpsPointId:

- Description:
- Format: UuidCanonicalTextual

OwnershipDeedId:

- Description:

OwnershipDeedValidatorAddr:

- Description:
- Format: AlgoAddressStringFormat

OwnerAddr:

- Description:
- Format: AlgoAddressStringFormat

DaemonAddr:

- Description:
- Format: AlgoAddressStringFormat

TradingRightsId:

- Description:

ScadaAlgoAddr:

- Description:
- Format: AlgoAddressStringFormat

ScadaCertId:

- Description:

class gwatn.types.base_g_node_gt.**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.base_g_node_gt.**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.base_g_node_gt.**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.**BaseGNodeGt_Maker**(*g_node_id*, *alias*, *status*, *role*, *g_node_registry_addr*, *prev_alias*, *gps_point_id*, *ownership_deed_id*, *ownership_deed_validator_addr*, *owner_addr*, *daemon_addr*, *trading_rights_id*, *scada_algo_addr*, *scada_cert_id*)

Parameters

- **g_node_id** (*str*) –
- **alias** (*str*) –
- **status** (*GNodeStatus*) –
- **role** (*CoreGNodeRole*) –
- **g_node_registry_addr** (*str*) –
- **prev_alias** (*str | None*) –
- **gps_point_id** (*str | None*) –
- **ownership_deed_id** (*int | None*) –
- **ownership_deed_validator_addr** (*str | None*) –
- **owner_addr** (*str | None*) –
- **daemon_addr** (*str | None*) –
- **trading_rights_id** (*int | None*) –
- **scada_algo_addr** (*str | None*) –
- **scada_cert_id** (*int | None*) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (BaseGNodeGt) –

Return type
str

classmethod **type_to_tuple**(*t*)

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

Parameters
t (*str*) –

Return type
BaseGNodeGt

1.6.6 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

1.6.7 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

1.6.8 DispatchContractConfirmedHeatpumpwithbooststore

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

class gwatn.types.**DispatchContractConfirmedHeatpumpwithbooststore**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *SignedProof*, *AtnParams*, *TypeName='dispatch.contract.confirmed.heatpumpwi* *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*) –
- **SignedProof** (*str*) –
- **AtnParams** (AtnParamsHeatpumpwithbooststore) –
- **TypeName** (*Literal['dispatch.contract.confirmed.heatpumpwithbooststore']*) –
- **Version** (*str*) –

FromGNodeAlias:

- Description:
- Format: LeftRightDot

FromGNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

SignedProof:

- Description:
- Format: AlgoMsgPackEncoded

AtnParams:

- Description:

class gwatn.types.dispatch_contract_confirmed_heatpumpwithbooststore.**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_heatpumpwithbooststore.**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_heatpumpwithbooststore.**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.**DispatchContractConfirmedHeatpumpwithbooststore_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *signed_proof*, *atn_params*)

Parameters

- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **signed_proof** (*str*) –
- **atn_params** (AtnParamsHeatpumpwithbooststore) –

classmethod tuple_to_type(*tuple*)

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

Parameters
tuple (DispatchContractConfirmedHeatpumpwithbooststore) –

Return type
str

classmethod type_to_tuple(*t*)

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

Parameters
t (*str*) –

Return type
DispatchContractConfirmedHeatpumpwithbooststore

1.6.9 FloParamsHeatpumpwithbooststore

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

class gwatn.types.**FloParamsHeatpumpwithbooststore**(*, *StoreSizeGallons=240*, *MaxStoreTempF=190*, *StoreMaxPowerKw=9*, *RatedHeatpumpElectricityKw=5.5*, *MaxHeatpumpSourceWaterTempF=135*, *SystemMaxHeatOutputSwtF=160*, *SystemMaxHeatOutputDeltaTempF=20*, *SystemMaxHeatOutputGpm=6*, *EmitterMaxSafeSwtF=170*, *CirculatorPumpMaxGpm=10*, *HeatpumpTariff=DistributionTariff.VersantStorageHeatTariff*, *HeatpumpEnergySupplyType=EnergySupplyType.RealtimeLocalLmp*, *BoostTariff=DistributionTariff.VersantStorageHeatTariff*, *BoostEnergySupplyType=EnergySupplyType.RealtimeLocalLmp*, *StandardOfferPriceDollarsPerMwh=110*, *DistributionTariffDollarsPerMwh=113*, *AmbientTempStoreF=65*, *StorePassiveLossRatio=0.001*, *RoomTempF=70*, *AmbientPowerInKw=1.25*, *ZeroPotentialEnergyWaterTempF=100*, *EmitterPumpFeedbackModel=EmitterPumpFeedbackModel.ConstantDeltaT*, *MixingValveFeedbackModel=MixingValveFeedbackModel.NaiveVariableSwt*, *CautiousMixingValveTempDeltaF=5*, *Cop1TempF=-20*, *Cop4TempF=70*, *CurrencyUnit=RecognizedCurrencyUnit.USD*, *TempUnit=RecognizedTemperatureUnit.F*, *TimezoneString='US/Eastern'*, *HomeCity='MILLINOCKET_ME'*, *StorageSteps=100*, *HouseWorstCaseTempF=-7*, *SystemMaxHeatOutputKwAvg=11.68*, *K=-1.1*, *IsRegulating=False*, *SliceDurationMinutes=[60]*, *PowerRequiredByHouseFromSystemAvgKwList=[3.42]*, *OutsideTempF=[-5.1]*, *RealtimeElectricityPrice=[10.35]*, *DistributionPrice=[40.0]*, *RegulationPrice=[25.3]*, *RtElecPriceUid*, *WeatherUid*, *DistPriceUid=None*, *RegPriceUid=None*, *StartYearUtc=2020*, *StartMonthUtc=1*, *StartDayUtc=1*, *StartHourUtc=1*, *StartMinuteUtc=0*, *StartingStoreIdx=50*, *TypeName='flo.params.heatpumpwithbooststore'*, *Version='000'*)

Parameters

- **StoreSizeGallons** (*int*) –
- **MaxStoreTempF** (*int*) –

- **StoreMaxPowerKw** (*float*) –
- **RatedHeatpumpElectricityKw** (*float*) –
- **MaxHeatpumpSourceWaterTempF** (*int*) –
- **SystemMaxHeatOutputSwtF** (*int*) –
- **SystemMaxHeatOutputDeltaTempF** (*int*) –
- **SystemMaxHeatOutputGpm** (*float*) –
- **EmitterMaxSafeSwtF** (*int*) –
- **CirculatorPumpMaxGpm** (*float*) –
- **HeatpumpTariff** (*DistributionTariff*) –
- **HeatpumpEnergySupplyType** (*EnergySupplyType*) –
- **BoostTariff** (*DistributionTariff*) –
- **BoostEnergySupplyType** (*EnergySupplyType*) –
- **StandardOfferPriceDollarsPerMwh** (*int*) –
- **DistributionTariffDollarsPerMwh** (*int*) –
- **AmbientTempStoreF** (*int*) –
- **StorePassiveLossRatio** (*float*) –
- **RoomTempF** (*int*) –
- **AmbientPowerInKw** (*float*) –
- **ZeroPotentialEnergyWaterTempF** (*int*) –
- **EmitterPumpFeedbackModel** (*EmitterPumpFeedbackModel*) –
- **MixingValveFeedbackModel** (*MixingValveFeedbackModel*) –
- **CautiousMixingValveTempDeltaF** (*int*) –
- **Cop1TempF** (*int*) –
- **Cop4TempF** (*int*) –
- **CurrencyUnit** (*RecognizedCurrencyUnit*) –
- **TempUnit** (*RecognizedTemperatureUnit*) –
- **TimezoneString** (*str*) –
- **HomeCity** (*str*) –
- **StorageSteps** (*int*) –
- **HouseWorstCaseTempF** (*int*) –
- **SystemMaxHeatOutputKwAvg** (*float*) –
- **K** (*float*) –
- **IsRegulating** (*bool*) –
- **SliceDurationMinutes** (*List[int]*) –
- **PowerRequiredByHouseFromSystemAvgKwList** (*List[float]*) –
- **OutsideTempF** (*List[float]*) –

- **RealtimeElectricityPrice** (*List[float]*) –
- **DistributionPrice** (*List[float]*) –
- **RegulationPrice** (*List[float]*) –
- **RtElecPriceUid** (*str*) –
- **WeatherUid** (*str*) –
- **DistPriceUid** (*str | None*) –
- **RegPriceUid** (*str | None*) –
- **StartYearUtc** (*int*) –
- **StartMonthUtc** (*int*) –
- **StartDayUtc** (*int*) –
- **StartHourUtc** (*int*) –
- **StartMinuteUtc** (*int*) –
- **StartingStoreIdx** (*int*) –
- **TypeName** (*Literal['flo.params.heatpumpwithbooststore']*) –
- **Version** (*str*) –

StoreSizeGallons:

- Description:

MaxStoreTempF:

- Description:

StoreMaxPowerKw:

- Description:

RatedHeatpumpElectricityKw:

- Description:

MaxHeatpumpSourceWaterTempF:

- Description:

SystemMaxHeatOutputSwtF:

- Description:

SystemMaxHeatOutputDeltaTempF:

- Description:

SystemMaxHeatOutputGpm:

- Description:

EmitterMaxSafeSwtF:

- Description:

CirculatorPumpMaxGpm:

- Description:

HeatpumpTariff:

- Description:

HeatpumpEnergySupplyType:

- Description:

BoostTariff:

- Description:

BoostEnergySupplyType:

- Description:

StandardOfferPriceDollarsPerMwh:

- Description:

DistributionTariffDollarsPerMwh:

- Description:

AmbientTempStoreF:

- Description:

StorePassiveLossRatio:

- Description:

RoomTempF:

- Description:

AmbientPowerInKw:

- Description:

ZeroPotentialEnergyWaterTempF:

- Description:

EmitterPumpFeedbackModel:

- Description:

MixingValveFeedbackModel:

- Description:

CautiousMixingValveTempDeltaF:

- Description:

Cop1TempF:

- Description:

Cop4TempF:

- Description:

CurrencyUnit:

- Description:

TempUnit:

- Description:

TimezoneString:

- Description:

HomeCity:

- Description:

StorageSteps:

- Description:

HouseWorstCaseTempF:

- Description:

SystemMaxHeatOutputKwAvg:

- Description:

K:

- Description: . Rate at which temperature falls in the pipes, when divided by gpm

IsRegulating:

- Description:

SliceDurationMinutes:

- Description:

PowerRequiredByHouseFromSystemAvgKwList:

- Description:

OutsideTempF:

- Description:

RealtimeElectricityPrice:

- Description:

DistributionPrice:

- Description:

RegulationPrice:

- Description:

RtElecPriceUid:

- Description:
- Format: UuidCanonicalTextual

WeatherUid:

- Description:
- Format: UuidCanonicalTextual

DistPriceUid:

- Description:
- Format: UuidCanonicalTextual

RegPriceUid:

- Description:

- Format: UuidCanonicalTextual

StartYearUtc:

- Description:

StartMonthUtc:

- Description:

StartDayUtc:

- Description:

StartHourUtc:

- Description:

StartMinuteUtc:

- Description:

StartingStoreIdx:

- Description:

class gwatn.types.**FloParamsHeatpumpwithbooststore_Maker**(*store_size_gallons*, *max_store_temp_f*, *store_max_power_kw*, *rated_heatpump_electricity_kw*, *max_heatpump_source_water_temp_f*, *system_max_heat_output_swt_f*, *system_max_heat_output_delta_temp_f*, *system_max_heat_output_gpm*, *emitter_max_safe_swt_f*, *circulator_pump_max_gpm*, *heatpump_tariff*, *heatpump_energy_supply_type*, *boost_tariff*, *boost_energy_supply_type*, *standard_offer_price_dollars_per_mwh*, *distribution_tariff_dollars_per_mwh*, *ambient_temp_store_f*, *store_passive_loss_ratio*, *room_temp_f*, *ambient_power_in_kw*, *zero_potential_energy_water_temp_f*, *emitter_pump_feedback_model*, *mixing_valve_feedback_model*, *cautious_mixing_valve_temp_delta_f*, *cop1_temp_f*, *cop4_temp_f*, *currency_unit*, *temp_unit*, *timezone_string*, *home_city*, *storage_steps*, *house_worst_case_temp_f*, *system_max_heat_output_kw_avg*, *k*, *is_regulating*, *slice_duration_minutes*, *power_required_by_house_from_system_avg_kw_list*, *outside_temp_f*, *realtime_electricity_price*, *distribution_price*, *regulation_price*, *rt_elec_price_uid*, *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*)

Parameters

- **store_size_gallons** (*int*) –
- **max_store_temp_f** (*int*) –
- **store_max_power_kw** (*float*) –
- **rated_heatpump_electricity_kw** (*float*) –
- **max_heatpump_source_water_temp_f** (*int*) –
- **system_max_heat_output_swt_f** (*int*) –
- **system_max_heat_output_delta_temp_f** (*int*) –
- **system_max_heat_output_gpm** (*float*) –
- **emitter_max_safe_swt_f** (*int*) –
- **circulator_pump_max_gpm** (*float*) –
- **heatpump_tariff** (*DistributionTariff*) –
- **heatpump_energy_supply_type** (*EnergySupplyType*) –
- **boost_tariff** (*DistributionTariff*) –
- **boost_energy_supply_type** (*EnergySupplyType*) –
- **standard_offer_price_dollars_per_mwh** (*int*) –
- **distribution_tariff_dollars_per_mwh** (*int*) –
- **ambient_temp_store_f** (*int*) –
- **store_passive_loss_ratio** (*float*) –
- **room_temp_f** (*int*) –
- **ambient_power_in_kw** (*float*) –
- **zero_potential_energy_water_temp_f** (*int*) –
- **emitter_pump_feedback_model** (*EmitterPumpFeedbackModel*) –
- **mixing_valve_feedback_model** (*MixingValveFeedbackModel*) –
- **cautious_mixing_valve_temp_delta_f** (*int*) –
- **cop1_temp_f** (*int*) –
- **cop4_temp_f** (*int*) –
- **currency_unit** (*RecognizedCurrencyUnit*) –
- **temp_unit** (*RecognizedTemperatureUnit*) –
- **timezone_string** (*str*) –
- **home_city** (*str*) –
- **storage_steps** (*int*) –
- **house_worst_case_temp_f** (*int*) –
- **system_max_heat_output_kw_avg** (*float*) –
- **k** (*float*) –
- **is_regulating** (*bool*) –

- **slice_duration_minutes** (*List[int]*) –
- **power_required_by_house_from_system_avg_kw_list** (*List[float]*) –
- **outside_temp_f** (*List[float]*) –
- **realtime_electricity_price** (*List[float]*) –
- **distribution_price** (*List[float]*) –
- **regulation_price** (*List[float]*) –
- **rt_elec_price_uid** (*str*) –
- **weather_uid** (*str*) –
- **dist_price_uid** (*str | None*) –
- **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*) –

1.6.10 GNodeGt

Python pydantic class corresponding to json type `g.node.gt`.

class gwatn.types.**GNodeGt**(**, GNodeId, Alias, Status, Role, GNodeRegistryAddr, PrevAlias=None, GpsPointId=None, OwnershipDeedId=None, OwnershipDeedValidatorAddr=None, OwnerAddr=None, DaemonAddr=None, TradingRightsId=None, ScadaAlgoAddr=None, ScadaCertId=None, ComponentId=None, DisplayName=None, TypeName='g.node.gt', Version='002'*)

Used to send and receive updates about GNodes.

GNodes are the building blocks of Gridworks. They have slowly-changing state that must be kept in sync across a distributed system. Therefore, they require a global registry to act as Single Source of Truth (SSoT). This class is used for that SSoT to share information with actors about their GNodes, and the GNodes that they will observe and communicate with. [More info](https://gridworks.readthedocs.io/en/latest/g-node.html).

Parameters

- **GNodeId** (*str*) –
- **Alias** (*str*) –
- **Status** (*GNodeStatus*) –
- **Role** (*GNodeRole*) –
- **GNodeRegistryAddr** (*str*) –
- **PrevAlias** (*str | None*) –
- **GpsPointId** (*str | None*) –
- **OwnershipDeedId** (*int | None*) –

- **OwnershipDeedValidatorAddr** (*str | None*) –
- **OwnerAddr** (*str | None*) –
- **DaemonAddr** (*str | None*) –
- **TradingRightsId** (*int | None*) –
- **ScadaAlgoAddr** (*str | None*) –
- **ScadaCertId** (*int | None*) –
- **ComponentId** (*str | None*) –
- **DisplayName** (*str | None*) –
- **TypeName** (*Literal['g.node.gt']*) –
- **Version** (*str*) –

GNodeId:

- Description: Immutable identifier for GNode
- Format: UuidCanonicalTextual

Alias:

- Description: Structured mutable identifier for GNode. The GNode Aliases are used for organizing how actors in Gridworks communicate. Together, they also encode the known topology of the electric grid.
- Format: LeftRightDot

Status:

- Description: Lifecycle indicator

Role:

- Description: Role within Gridworks

GNodeRegistryAddr:

- Description: Algorand address for GNodeRegistry. For actors in a Gridworks world, the GNodeRegistry is the Single Source of Truth for existence and updates to GNodes.
- Format: AlgoAddressStringFormat

PrevAlias:

- Description: Previous GNodeAlias. As the topology of the grid updates, GNodeAliases will change to reflect that. This may happen a handful of times over the life of a GNode.
- Format: LeftRightDot

GpsPointId:

- Description: Lat/lon of GNode. Some GNodes, in particular those acting as avatars for physical devices that are part of or are attached to the electric grid, have physical locations. These locations are used to help validate the grid topology.
- Format: UuidCanonicalTextual

OwnershipDeedId:

- Description: Algorand Id of ASA Deed. The Id of the TaDeed Algorand Standard Asset if the GNode is a TerminalAsset.

OwnershipDeedValidatorAddr:

- Description: Algorand address of Validator. Deeds are issued by the GNodeFactory, in partnership with third party Validators.
- Format: AlgoAddressStringFormat

OwnerAddr:

- Description: Algorand address of the deed owner
- Format: AlgoAddressStringFormat

DaemonAddr:

- Description: Algorand address of the daemon app. Some GNodes have Daemon applications associated to them to handle blockchain operations.
- Format: AlgoAddressStringFormat

TradingRightsId:

- Description: Algorand Id of ASA TradingRights. The Id of the TradingRights Algorand Standard Asset.

ScadaAlgoAddr:

- Description:
- Format: AlgoAddressStringFormat

ScadaCertId:

- Description:

ComponentId:

- Description: Unique identifier for GNode's Component. Used if a GNode is an avatar for a physical device. The serial number of a device is different from its make/model. The ComponentId captures the specific instance of the device.
- Format: UuidCanonicalTextual

DisplayName:

- Description: Display Name

class gwatn.types.g_node_gt.**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.g_node_gt.**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.g_node_gt.**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.**GNodeGt_Maker**(*g_node_id*, *alias*, *status*, *role*, *g_node_registry_addr*, *prev_alias*, *gps_point_id*, *ownership_deed_id*, *ownership_deed_validator_addr*, *owner_addr*, *daemon_addr*, *trading_rights_id*, *scada_algo_addr*, *scada_cert_id*, *component_id*, *display_name*)

Parameters

- **g_node_id** (*str*) –
- **alias** (*str*) –
- **status** (*GNodeStatus*) –
- **role** (*GNodeRole*) –
- **g_node_registry_addr** (*str*) –
- **prev_alias** (*str | None*) –
- **gps_point_id** (*str | None*) –
- **ownership_deed_id** (*int | None*) –
- **ownership_deed_validator_addr** (*str | None*) –
- **owner_addr** (*str | None*) –
- **daemon_addr** (*str | None*) –
- **trading_rights_id** (*int | None*) –
- **scada_algo_addr** (*str | None*) –
- **scada_cert_id** (*int | None*) –
- **component_id** (*str | None*) –
- **display_name** (*str | None*) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (GNodeGt) –

Return type
str

classmethod **type_to_tuple**(*t*)

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

Parameters
t (*str*) –

Return type
GNodeGt

1.6.11 GNodeInstanceGt

Python pydantic class corresponding to json type `g.node.instance.gt`.

class gwatn.types.**GNodeInstanceGt**(*, *GNodeInstanceId*, *GNode*, *Strategy*, *Status*, *SupervisorContainerId*, *StartTimeUnixS*, *EndTimeUnixS*, *AlgoAddress=None*, *TypeName='g.node.instance.gt'*, *Version='000'*)

Used to send and receive updates about GNodeInstances.

One of the layers of abstraction connecting a GNode with a running app in a Docker container.

[More info](https://gridworks.readthedocs.io/en/latest/g-node-instance.html).

Parameters

- **GNodeInstanceId** (*str*) –
- **GNode** (GNodeGt) –
- **Strategy** (*StrategyName*) –
- **Status** (*GniStatus*) –
- **SupervisorContainerId** (*str*) –
- **StartTimeUnixS** (*int*) –
- **EndTimeUnixS** (*int*) –
- **AlgoAddress** (*str | None*) –
- **TypeName** (*Literal['g.node.instance.gt']*) –
- **Version** (*str*) –

GNodeInstanceId:

- Description: Immutable identifier for GNodeInstance (Gni)
- Format: UuidCanonicalTextual

GNode:

- Description: The GNode represented by the Gni

Strategy:

- Description: Used to determine the code running in a GNode actor application

Status:

- Description: Lifecycle Status for Gni

SupervisorContainerId:

- Description: The Id of the docker container where the Gni runs
- Format: UuidCanonicalTextual

StartTimeUnixS:

- Description: When the gni starts representing the GNode. Specifically, when the Status changes from Pending to Active. Note that this is time in the GNode's World, which may not be real time if it is a simulation.
- Format: ReasonableUnixTimeS

EndTimeUnixS:

- Description: When the gni stops representing the GNode. Specifically, when the Status changes from Active to Done.

AlgoAddress:

- Description: Algorand address for Gni
- Format: AlgoAddressStringFormat

class gwatn.types.g_node_instance_gt.**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.g_node_instance_gt.**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.g_node_instance_gt.**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.**GNodeInstanceGt_Maker**(*g_node_instance_id*, *g_node*, *strategy*, *status*, *supervisor_container_id*, *start_time_unix_s*, *end_time_unix_s*, *algo_address*)

Parameters

- **g_node_instance_id** (*str*) –
- **g_node** (GNodeGt) –
- **strategy** (*StrategyName*) –
- **status** (*GniStatus*) –
- **supervisor_container_id** (*str*) –
- **start_time_unix_s** (*int*) –
- **end_time_unix_s** (*int*) –
- **algo_address** (*str | None*) –

classmethod tuple_to_type(*tuple*)

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

Parameters
tuple (GNodeInstanceGt) –

Return type
str

classmethod type_to_tuple(*t*)

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

Parameters
t (*str*) –

Return type
GNodeInstanceGt

1.6.12 GtDispatchBoolean

Python pydantic class corresponding to json type `gt.dispatch.boolean`.

class gwatn.types.**GtDispatchBoolean**(**, AboutNodeName, ToGNodeAlias, FromGNodeAlias, FromGNodeInstanceId, RelayState, SendTimeUnixMs, TypeName='gt.dispatch.boolean', Version='110'*)

.

Boolean dispatch command sent over the cloud broker from one GNode (FromGNodeAlias/ FromGNodeId) to another (ToGNodeAlias). AboutNodeAlias is the node getting dispatched. It may be a GNode, but it can also be a SpaceHeatNode.

Parameters

- **AboutNodeName** (*str*) –
- **ToGNodeAlias** (*str*) –
- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **RelayState** (*int*) –
- **SendTimeUnixMs** (*int*) –
- **TypeName** (*Literal['gt.dispatch.boolean']*) –
- **Version** (*str*) –

AboutNodeName:

- Description: The Spaceheat Node getting dispatched
- Format: LeftRightDot

ToGNodeAlias:

- Description: GNodeAlias of the SCADA
- Format: LeftRightDot

FromGNodeAlias:

- Description: GNodeAlias of AtomicTNode
- Format: LeftRightDot

FromGNodeInstanceId:

- Description: GNodeInstance of the AtomicTNode

- Format: UuidCanonicalTextual

RelayState:

- Description: 0 or 1

SendTimeUnixMs:

- Description: Time the AtomicTNode sends the dispatch, by its clock
- Format: ReasonableUnixTimeMs

class gwatn.types.gt_dispatch_boolean.**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.gt_dispatch_boolean.**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.gt_dispatch_boolean.**check_is_reasonable_unix_time_ms**(*v*)

ReasonableUnixTimeMs format: time in unix milliseconds between Jan 1 2000 and Jan 1 3000

Raises
ValueError – if not ReasonableUnixTimeMs format

Parameters
v (*str*) –

class gwatn.types.**GtDispatchBoolean_Maker**(*about_node_name*, *to_g_node_alias*, *from_g_node_alias*, *from_g_node_instance_id*, *relay_state*, *send_time_unix_ms*)

Parameters

- **about_node_name** (*str*) –
- **to_g_node_alias** (*str*) –
- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **relay_state** (*int*) –
- **send_time_unix_ms** (*int*) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (GtDispatchBoolean) –

Return type
str

classmethod type_to_tuple(*t*)

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

Parameters
t (*str*) –

Return type
GtDispatchBoolean

1.6.13 GwCertId

Python pydantic class corresponding to json type `gw.cert.id`.

class gwatn.types.**GwCertId**(*, *Type*, *Idx=None*, *Addr=None*, *TypeName='gw.cert.id'*, *Version='000'*)

Clarifies whether cert id is an Algorand Standard Asset or SmartSig

Parameters

- **Type** (AlgoCertType) –
- **Idx** (*int | None*) –
- **Addr** (*str | None*) –
- **TypeName** (*Literal['gw.cert.id']*) –
- **Version** (*str*) –

classmethod check_axiom_1(*v*)

Axiom 1: Cert type consistency. If Type is ASA, then Id exists and Addr does not. Otherwise, Addr exists and Id does not.

Parameters
v (*dict*) –

Return type
dict

Type:

- Description:

Idx:

- Description: ASA Index

Addr:

- Description: Algorand Smart Signature Address
- Format: AlgoAddressStringFormat

class gwatn.types.gw_cert_id.**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.**GwCertId_Maker**(*type*, *idx*, *addr*)

Parameters

- **type** (AlgoCertType) –
- **idx** (*int | None*) –
- **addr** (*str | None*) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (GwCertId) –

Return type
str

classmethod **type_to_tuple**(*t*)

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

Parameters
t (*str*) –

Return type
GwCertId

1.6.14 HeartbeatA

Python pydantic class corresponding to json type `heartbeat.a`.

class gwatn.types.**HeartbeatA**(*, *MyHex='0'*, *YourLastHex='0'*, *TypeName='heartbeat.a'*, *Version='100'*)

Used to check that an actor can both send and receive messages.

Payload for direct messages sent back and forth between actors, for example a Supervisor and one of its subordinates.

[More info](https://gridworks.readthedocs.io/en/latest/g-node-instance.html).

Parameters

- **MyHex** (*str*) –
- **YourLastHex** (*str*) –
- **TypeName** (*Literal['heartbeat.a']*) –
- **Version** (*str*) –

MyHex:

- Description: Hex character getting sent
- Format: HexChar

YourLastHex:

- Description: Last hex character received from heartbeat partner
- Format: HexChar

class gwatn.types.heartbeat_a.**check_is_hex_char**(*v*)

HexChar format: single-char string in '0123456789abcdefABCDEF'

Raises
ValueError – if not HexChar format

Parameters
v (*str*) –

class gwatn.types.**HeartbeatA_Maker**(*my_hex*, *your_last_hex*)

Parameters

- **my_hex** (*str*) –
- **your_last_hex** (*str*) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (HeartbeatA) –

Return type
str

classmethod **type_to_tuple**(*t*)

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

Parameters
t (*str*) –

Return type
HeartbeatA

1.6.15 HeartbeatAlgoAudit

Python pydantic class corresponding to json type `heartbeat.algo.audit`.

class gwatn.types.**HeartbeatAlgoAudit**(*, *SignedProof*, *Heartbeat*, *TypeName='heartbeat.algo.audit'*, *Version='000'*)

.

Algo payload with report of last HeartbeatB sent to partner via Rabbit, to be sent to DispatchContract on Algo blockchain

Parameters

- **SignedProof** (*str*) –
- **Heartbeat** (HeartbeatB) –
- **TypeName** (*Literal['heartbeat.algo.audit']*) –
- **Version** (*str*) –

SignedProof:

- Description: Tiny signed payment to DispatchContract to prove identity. Can be a minimal payment, as long as it comes from the AtomicTNode or SCADA.
- Format: AlgoMsgPackEncoded

Heartbeat:

- Description: Heartbeat sender last sent to its partner

class gwatn.types.heartbeat_algo_audit.**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.heartbeat_algo_audit.**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.**HeartbeatAlgoAudit_Maker**(*signed_proof*, *heartbeat*)

Parameters

- **signed_proof** (*str*) –
- **heartbeat** (HeartbeatB) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (HeartbeatAlgoAudit) –

Return type
str

classmethod **type_to_tuple**(*t*)

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

Parameters
t (*str*) –

Return type
HeartbeatAlgoAudit

1.6.16 HeartbeatB

Python pydantic class corresponding to json type `heartbeat.b`.

class gwatn.types.**HeartbeatB**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *MyHex='0'*, *YourLastHex=None*, *LastReceivedTimeUnixMs*, *SendTimeUnixMs*, *TypeName='heartbeat.b'*, *Version='001'*)

.

Heartbeat for Scada-AtomicTNode DispatchContract, to send via RabbitMQ

Parameters

- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **MyHex** (*str*) –
- **YourLastHex** (*str | None*) –
- **LastReceivedTimeUnixMs** (*int*) –
- **SendTimeUnixMs** (*int*) –
- **TypeName** (*Literal['heartbeat.b']*) –
- **Version** (*str*) –

FromGNodeAlias:

- Description: My GNodeAlias
- Format: LeftRightDot

FromGNodeInstanceId:

- Description: My GNodeInstanceId
- Format: UuidCanonicalTextual

MyHex:

- Description: Hex character getting sent
- Format: HexChar

YourLastHex:

- Description: Last hex character received from heartbeat partner. If the heartbeat initiator wants to start the sequence over, it does not include this.
- Format: HexChar

LastReceivedTimeUnixMs:

- Description: Time YourLastHex was received on my clock
- Format: ReasonableUnixTimeMs

SendTimeUnixMs:

- Description: Time this message is made and sent on my clock
- Format: ReasonableUnixTimeMs

class gwatn.types.heartbeat_b.**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.heartbeat_b.**check_is_hex_char**(*v*)

HexChar format: single-char string in '0123456789abcdefABCDEF'

Raises
ValueError – if not HexChar format

Parameters
v (*str*) –

class gwatn.types.heartbeat_b.**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.heartbeat_b.**check_is_reasonable_unix_time_ms**(*v*)

ReasonableUnixTimeMs format: time in unix milliseconds between Jan 1 2000 and Jan 1 3000

Raises
ValueError – if not ReasonableUnixTimeMs format

Parameters
v (*str*) –

class gwatn.types.**HeartbeatB_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *my_hex*, *your_last_hex*, *last_received_time_unix_ms*, *send_time_unix_ms*)

Parameters

- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **my_hex** (*str*) –
- **your_last_hex** (*str | None*) –
- **last_received_time_unix_ms** (*int*) –
- **send_time_unix_ms** (*int*) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (HeartbeatB) –

Return type
str

classmethod **type_to_tuple**(*t*)

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

Parameters
t (*str*) –

Return type
HeartbeatB

1.6.17 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

1.6.18 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

1.6.19 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

1.6.20 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

1.6.21 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

1.6.22 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

1.6.23 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

1.6.24 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

1.6.25 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

1.6.26 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

1.6.27 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

1.6.28 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

1.6.29 Ready

Python pydantic class corresponding to json type `ready`.

class gwatn.types.**Ready**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *TimeUnixS*, *TypeName='ready'*, *Version='001'*)

Used in simulations by TimeCoordinator GNodes.

Only intended for simulations that do not have sub-second TimeSteps. TimeCoordinators based on `gridworks-timecoordinator` have a notion of actors whose *Ready* must be received before issuing the next TimeStep. [More info](https://gridworks.readthedocs.io/en/latest/time-coordinator.html).

Parameters

- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **TimeUnixS** (*int*) –
- **TypeName** (*Literal['ready']*) –
- **Version** (*str*) –

FromGNodeAlias:

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

FromGNodeInstanceId:

- Description: The GNodeInstanceId of the sender
- Format: UuidCanonicalTextual

TimeUnixS:

- Description: Latest simulated time for sender. The time in unix seconds of the latest TimeStep received from the TimeCoordinator by the actor that sent the payload.

class gwatn.types.ready.**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.ready.**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.**Ready_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *time_unix_s*)

Parameters

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

classmethod tuple_to_type(*tuple*)

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

Parameters
tuple (Ready) –

Return type
str

classmethod type_to_tuple(*t*)

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

Parameters
t (*str*) –

Return type
Ready

1.6.30 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

1.6.31 SimScadaDriverReport

Python pydantic class corresponding to json type `sim.scada.driver.report`.

class gwatn.types.**SimScadaDriverReport**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *BoostPowerKwTimes1000*, *HeatpumpPowerKwTimes1000*, *StoreKwh*, *CopTimes10*, *MaxStoreKwh*, *TypeName='sim.scada.driver.report'*, *Version='000'*)

Parameters

- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **BoostPowerKwTimes1000** (*int*) –
- **HeatpumpPowerKwTimes1000** (*int*) –
- **StoreKwh** (*int*) –
- **CopTimes10** (*int*) –
- **MaxStoreKwh** (*int*) –
- **TypeName** (*Literal['sim.scada.driver.report']*) –
- **Version** (*str*) –

FromGNodeAlias:

- Description:
- Format: LeftRightDot

FromGNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

BoostPowerKwTimes1000:

- Description:

HeatpumpPowerKwTimes1000:

- Description:

StoreKwh:

- Description:

CopTimes10:

- Description:

MaxStoreKwh:

- Description:

class gwatn.types.sim_scada_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.sim_scada_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.**SimScadaDriverReport_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *boost_power_kw_times1000*, *heatpump_power_kw_times1000*, *store_kwh*, *cop_times10*, *max_store_kwh*)

Parameters

- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **boost_power_kw_times1000** (*int*) –
- **heatpump_power_kw_times1000** (*int*) –
- **store_kwh** (*int*) –
- **cop_times10** (*int*) –
- **max_store_kwh** (*int*) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (SimScadaDriverReport) –

Return type
str

classmethod **type_to_tuple**(*t*)

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

Parameters
t (*str*) –

Return type
SimScadaDriverReport

1.6.32 SimTimestep

Python pydantic class corresponding to json type `sim.timestep`.

class gwatn.types.**SimTimestep**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *TimeUnixS*, *TimestepCreatedMs*, *MessageId*, *TypeName='sim.timestep'*, *Version='000'*)

Sent by TimeCoordinators to coordinate time.

For simulated actors, time progresses discretely on receipt of these time steps.

Parameters

- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **TimeUnixS** (*int*) –
- **TimestepCreatedMs** (*int*) –
- **MessageId** (*str*) –
- **TypeName** (*Literal['sim.timestep']*) –
- **Version** (*str*) –

FromGNodeAlias:

- Description: The GNodeAlias of the sender. The sender should always be a GNode Actor of role TimeCoordinator.
- Format: LeftRightDot

FromGNodeInstanceId:

- Description: The GNodeInstanceId of the sender
- Format: UuidCanonicalTextual

TimeUnixS:

- Description: Current time in unix seconds
- Format: ReasonableUnixTimeS

TimestepCreatedMs:

- Description: The real time created, in unix milliseconds
- Format: ReasonableUnixTimeMs

MessageId:

- Description: MessageId
- Format: UuidCanonicalTextual

class gwatn.types.sim_timestep.**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.sim_timestep.**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.sim_timestep.**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.sim_timestep.**check_is_reasonable_unix_time_ms**(*v*)

ReasonableUnixTimeMs format: time in unix milliseconds between Jan 1 2000 and Jan 1 3000

Raises
ValueError – if not ReasonableUnixTimeMs format

Parameters
v (*str*) –

class gwatn.types.**SimTimestep_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *time_unix_s*, *timestep_created_ms*, *message_id*)

Parameters

- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **time_unix_s** (*int*) –
- **timestep_created_ms** (*int*) –
- **message_id** (*str*) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (SimTimestep) –

Return type
str

classmethod **type_to_tuple**(*t*)

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

Parameters
t (*str*) –

Return type
SimTimestep

1.6.33 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

1.6.34 SnapshotHeatpumpwithbooststore

Python pydantic class corresponding to json type `snapshot.heatpumpwithbooststore`.

class gwatn.types.**SnapshotHeatpumpwithbooststore**(*, *FromGNodeAlias*, *FromGNodeInstanceId*, *BoostPowerKwTimes1000*, *HeatpumpPowerKwTimes1000*, *StoreKwh*, *CopTimes10*, *MaxStoreKwh*, *AboutTerminalAssetAlias*, *TypeName='snapshot.heatpumpwithbooststore'*, *Version='000'*)

Parameters

- **FromGNodeAlias** (*str*) –
- **FromGNodeInstanceId** (*str*) –
- **BoostPowerKwTimes1000** (*int*) –
- **HeatpumpPowerKwTimes1000** (*int*) –
- **StoreKwh** (*int*) –
- **CopTimes10** (*int*) –
- **MaxStoreKwh** (*int*) –
- **AboutTerminalAssetAlias** (*str*) –
- **TypeName** (*Literal['snapshot.heatpumpwithbooststore']*) –
- **Version** (*str*) –

FromGNodeAlias:

- Description:
- Format: LeftRightDot

FromGNodeInstanceId:

- Description:
- Format: UuidCanonicalTextual

BoostPowerKwTimes1000:

- Description:

HeatpumpPowerKwTimes1000:

- Description:

StoreKwh:

- Description:

CopTimes10:

- Description:

MaxStoreKwh:

- Description:

AboutTerminalAssetAlias:

- Description:

- Format: LeftRightDot

class gwatn.types.snapshot_heatpumpwithbooststore.**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.snapshot_heatpumpwithbooststore.**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.**SnapshotHeatpumpwithbooststore_Maker**(*from_g_node_alias*, *from_g_node_instance_id*, *boost_power_kw_times1000*, *heatpump_power_kw_times1000*, *store_kwh*, *cop_times10*, *max_store_kwh*, *about_terminal_asset_alias*)

Parameters

- **from_g_node_alias** (*str*) –
- **from_g_node_instance_id** (*str*) –
- **boost_power_kw_times1000** (*int*) –
- **heatpump_power_kw_times1000** (*int*) –
- **store_kwh** (*int*) –
- **cop_times10** (*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 (SnapshotHeatpumpwithbooststore) –

Return type
str

classmethod type_to_tuple(*t*)

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

Parameters
t (*str*) –

Return type
SnapshotHeatpumpwithbooststore

1.6.35 SuperStarter

Python pydantic class corresponding to json type `super.starter`.

class gwatn.types.**SuperStarter**(*, *SupervisorContainer*, *GniList*, *AliasWithKeyList*, *KeyList*, *TypeName='super.starter'*, *Version='000'*)

Used by world to seed a docker container with data needed to spawn and superviser GNodeInstances

Parameters

- **SupervisorContainer** (SupervisorContainerGt) –
- **GniList** (*List[GNodeInstanceGt]*) –
- **AliasWithKeyList** (*List[str]*) –
- **KeyList** (*List[str]*) –
- **TypeName** (*Literal['super.starter']*) –
- **Version** (*str*) –

SupervisorContainer:

- Description: Key data about the docker container

GniList:

- Description: List of GNodeInstances (Gnis) run in the container

AliasWithKeyList:

- Description: Aliases of Gnis that own Algorand secret keys
- Format: LeftRightDot

KeyList:

- Description: Algorand secret keys owned by Gnis

class gwatn.types.super_starter.**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.**SuperStarter_Maker**(*supervisor_container*, *gni_list*, *alias_with_key_list*, *key_list*)

Parameters

- **supervisor_container** (SupervisorContainerGt) –
- **gni_list** (*List[GNodeInstanceGt]*) –
- **alias_with_key_list** (*List[str]*) –
- **key_list** (*List[str]*) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (SuperStarter) –

Return type
str

classmethod type_to_tuple(*t*)

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

Parameters
t (*str*) –

Return type
SuperStarter

1.6.36 SupervisorContainerGt

Python pydantic class corresponding to json type `supervisor.container.gt`.

class gwatn.types.**SupervisorContainerGt**(*, *SupervisorContainerId*, *Status*, *WorldInstanceName*, *SupervisorGNodeInstanceId*, *SupervisorGNodeAlias*, *TypeName='supervisor.container.gt'*, *Version='000'*)

Used to send and receive updates about SupervisorContainers.

Sent from a GNodeRegistry to a World, and used also by the World as it spawns GNodeInstances in docker instances (i.e., the SupervisorContainers). [More info](https://gridworks.readthedocs.io/en/latest/supervisor.html).

Parameters

- **SupervisorContainerId** (*str*) –
- **Status** (*SupervisorContainerStatus*) –
- **WorldInstanceName** (*str*) –
- **SupervisorGNodeInstanceId** (*str*) –
- **SupervisorGNodeAlias** (*str*) –
- **TypeName** (*Literal['supervisor.container.gt']*) –
- **Version** (*str*) –

SupervisorContainerId:

- Description: Id of the docker SupervisorContainer
- Format: UuidCanonicalTextual

Status:

- Description:

WorldInstanceName:

- Description: Name of the WorldInstance. For example, d1__1 is a potential name for a World whose World GNode has alias d1.
- Format: WorldInstanceNameFormat

SupervisorGNodeInstanceId:

- Description: Id of the SupervisorContainer's prime actor (aka the Supervisor GNode)
- Format: UuidCanonicalTextual

SupervisorGNodeAlias:

- Description: Alias of the SupervisorContainer's prime actor (aka the Supervisor GNode)
- Format: LeftRightDot

class gwatn.types.supervisor_container_gt.**check_is_world_instance_name_format**(*v*)

WorldInstanceName format: A single alphanumerical word starting with an alphabet char (the root GNodeAlias) and an integer, seperated by '__'. For example 'd1__1'

Raises
ValueError – if not WorldInstanceNameFormat format

Parameters
v (*str*) –

class gwatn.types.supervisor_container_gt.**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.supervisor_container_gt.**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.**SupervisorContainerGt_Maker**(*supervisor_container_id*, *status*, *world_instance_name*, *supervisor_g_node_instance_id*, *supervisor_g_node_alias*)

Parameters

- **supervisor_container_id** (*str*) –
- **status** (*SupervisorContainerStatus*) –
- **world_instance_name** (*str*) –
- **supervisor_g_node_instance_id** (*str*) –
- **supervisor_g_node_alias** (*str*) –

classmethod tuple_to_type(*tuple*)

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

Parameters
tuple (SupervisorContainerGt) –

Return type
str

classmethod type_to_tuple(*t*)

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

Parameters
t (*str*) –

Return type
SupervisorContainerGt

1.6.37 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

1.6.38 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

1.6.39 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)

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

1.6.40 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

1.7 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*

1.7.1 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.

1.7.2 How to request a feature

Request features on the Issue Tracker.

1.7.3 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
```

1.7.4 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.

1.7.5 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.

1.8 GNodeFactory Repo Code of Conduct

1.8.1 Basic Truth

All humans are worthy.

1.8.2 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.

1.8.3 Enforcement Responsibilities

Jessica Millar (jmillar@gridworks-consulting.com) is responsible for clarifying and enforcing this repo's standards of behavior. She has the right and responsibility to remove, edit, or reject comments, commits, code, wiki 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 Jessica to consider moderating, please send an email to her at jmillar@gridworks-consulting.com. All complaints will be reviewed and investigated, and Jessica will respect the privacy and security of the reporter of any incident.

1.8.4 Suggestions

- Respect privacy
- Empathize
- 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

1.8.5 Enforcement Escalation

1. Correction

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

2. Warning

With a warning, Jessica may remove your comments, commits or issues. She 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.

1.8.6 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.

1.9 License

PYTHON MODULE INDEX

g

INDEX

D

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V