Gridworks Marketmaker

Jessica Millar

Jan 18, 2023

CODE SUPPORT

This is the Python SDK for building MarketMakers for GridWorks. GridWorks uses distributed actors to balance the electric grid, and MarketMakers are the actors brokering this grid balancing via the markets they run for energy and balancing services.

This SDK is available as the gridworks-marketmaker pypi package. Documentation specific to using this SDK is available here. If this is your first time with GridWorks code, please start with the main GridWorks docs.

MarketMakers support grid balancing by running markets. They are geared to serve millions of coordinated and intelligent Transactive Devices, represented in their markets by AtomicTNodes. The veracity of the ex-poste energy and power data provided by AtomicTNodes to the MarketMaker is backed up via a series of GridWorks Certificates globally visible on the Algorand blockchain. These include the foundational TaDeeds that establish ownership of the underlying Transactive Device, and the TaTradingRights that give the AtomicTNode authority to represent the Transactive Device in its MarketMaker's markets.

CHAPTER

ONE

INSTALLATION

Note: gridworks-marketmaker requires python 3.10 or higher.

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

1.1 Type API Specs

1.1.1 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",
      "format": "MarketSlotNameLrdFormat",
      "title": "",
      "required": true
```

(continues on next page)

(continued from previous page)

```
    },
    "BidderAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "BidList": {
      "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
    }
  }
}
```

1.1.2 AtnBid

```
{
  "gwapi": "001",
  "type_name": "atn.bid",
  "version": "001",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "AtomicTNode bid sent to a MarketMaker",
  "url": "https://gridworks.readthedocs.io/en/latest/market-bid.html",
  "formats": {
    "UuidCanonicalTextual": {
      "type": "string",
      "description": "A string of hex words separated by hyphens of length 8-4-4-4-12.",
      "example": "652ba6b0-c3bf-4f06-8a80-6b9832d60a25"
    },
    "LeftRightDot": {
      "type": "string",
      "description": "Lowercase alphanumeric words separated by periods, most␣
↪significant word (on the left) starting with an alphabet character.",
```

(continues on next page)

(continued from previous page)


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


(continues on next page)

(continued from previous page)

```
        },
        {
          "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": ""
        },
        {
          "const": "c272f3b3",
          "title": "AvgkW",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "BidderAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
```

(continues on next page)

(continued from previous page)

```
    "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": {
      "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
```

(continues on next page)

(continued from previous page)

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

1.1.3 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": {
```

(continues on next page)

(continued from previous page)

```
      "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",
          "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",
```

(continues on next page)

(continued from previous page)

```
          "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"
        },
        {
          "const": "1dce1efd",
          "title": "WeatherService",
          "description": "Provides weather forecasts"
        },
        {
          "const": "db57d184",
          "title": "AggregatedTNode",
          "description": "An aggregation of AtomicTNodes"
        }
      ]
    },
    "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."
        },
```

(continues on next page)

(continued from previous page)

```
        {
          "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": {
    "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",
```

(continues on next page)

(continued from previous page)

```
      "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": {
      "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",
```

(continues on next page)

(continued from previous page)


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

1.1.4 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
    }
  }
}
```

1.1.5 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",
          "title": "USDPerMWh",
          "description": ""
        }
      ]
    }
  },
  "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
    },
    "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"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.1.6 MarketBook


```
{
  "gwapi": "001",
  "type_name": "market.book",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "MarketMaker's list of bids for a MarketSlot",
  "properties": {
    "Slot": {
      "type": "market.slot.000",
      "title": "MarketSlot the book is for",
      "required": true
    },
    "Bids": {
      "type": "accepted.bid.000",
      "title": "List of bids in the book",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "market.book.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```


1.1.7 MarketMakerInfo


```
{
  "gwapi": "001",
  "type_name": "market.maker.info",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "properties": {
    "GNodeAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "MarketTypeList": {
      "type": "market.type.gt.000",
      "title": "",
```


(continues on next page)

(continued from previous page)

```
      "required": true
    },
    "SampleMarketName": {
      "type": "string",
      "title": "",
      "required": true
    },
    "SampleMarketSlotName": {
      "type": "string",
      "format": "MarketSlotNameLrdFormat",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "market.maker.info.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.1.8 MarketPrice

```
{
  "gwapi": "001",
  "type_name": "market.price",
  "version": "000",
  "owner": "gridworks@gridworks-consulting.com",
  "description": "",
  "enums": {
    "MarketPriceUnit000": {
      "type": "string",
      "name": "market.price.unit.000",
      "description": "Price unit assigned to MarketMaker MarketType",
      "oneOf": [
        {
          "const": "00000000",
          "title": "USDPerMWh",
          "description": ""
        }
      ]
    }
  },
  "properties": {
```

(continues on next page)

(continued from previous page)

```
    "ValueTimes1000": {
      "type": "integer",
      "title": "",
      "required": true
    },
    "Unit": {
      "type": "string",
      "format": "MarketPriceUnit000",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "market.price.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.1.9 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",
```

(continues on next page)

(continued from previous page)

```
      "title": "",
      "required": true
    },
    "MarketMakerAlias": {
      "type": "string",
      "format": "LeftRightDot",
      "title": "",
      "required": true
    },
    "StartUnixS": {
      "type": "integer",
      "format": "ReasonableUnixTimeS",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "market.slot.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```

1.1.10 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",
```

(continues on next page)

(continued from previous page)


```
          "title": "USDPerMWh",
          "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"
        }
      ]
    },
    "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",
```


(continues on next page)

(continued from previous page)

```
          "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": ""
        },
        {
          "const": "c272f3b3",
          "title": "AvgkW",
          "description": ""
        }
      ]
    }
  },
  "properties": {
    "Name": {
      "type": "string",
      "format": "MarketTypeName000",
      "title": "Name of the MarketType",
      "required": true
```

(continues on next page)

(continued from previous page)


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

1.1.11 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",
      "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",
```


(continues on next page)

(continued from previous page)


```
      "title": "",
      "required": true
    },
    "QuantityUnit": {
      "type": "string",
      "format": "MarketQuantityUnit000",
      "title": "",
      "required": true
    },
    "InjectionIsPositive": {
      "type": "boolean",
      "title": "",
      "required": true
    },
    "TypeName": {
      "type": "string",
      "value": "price.quantity.000",
      "title": "The type name"
    },
    "Version": {
      "type": "string",
      "title": "The type version",
      "default": "000",
      "required": true
    }
  }
}
```


1.1.12 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",
```


(continues on next page)

(continued from previous page)

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

1.1.13 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": {
```

(continues on next page)

(continued from previous page)

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

1.1.14 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",
```

(continues on next page)

(continued from previous page)


```
      "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
    },
    "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 GridWorks DataClasses

GNode Global DataClass Definition

1.3 GridWorks Enums

GwSchema Enums used in gwmm

class gwmm.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

classmethod default()

Returns default value GNode

Return type
GNodeRole

classmethod values()

Returns enum choices

Return type
List[str]

class gwmm.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 gwmm.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 gwmm.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 gwmm.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 gwmm.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 gwmm.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 gwmm.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 gwmm.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]

1.4 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 schema types

1.4.1 AcceptedBid

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

class gwmm.types.**AcceptedBid**(*, *MarketSlotName*, *BidderAlias*, *BidList*, *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*) –
- **BidList** (*List[PriceQuantityUnitless]*) –
- **ReceivedTimeUnixNs** (*int*) –
- **TypeName** (*Literal['accepted.bid']*) –
- **Version** (*str*) –

MarketSlotName:

- Description:
- Format: MarketSlotNameLrdFormat

BidderAlias:

- Description:
- Format: LeftRightDot

BidList:

- Description:

ReceivedTimeUnixNs:

- Description:

class gwmm.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 gwmm.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 gwmm.types.**AcceptedBid_Maker**(*market_slot_name*, *bidder_alias*, *bid_list*, *received_time_unix_ns*)

Parameters

- **market_slot_name** (*str*) –
- **bidder_alias** (*str*) –
- **bid_list** (*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

1.4.2 AtnBid

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

class gwmm.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 gwmm.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 gwmm.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 gwmm.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 gwmm.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 gwmm.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

1.4.3 GNodeGt

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

class gwmm.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** (*Optional[str]*) –
- **GpsPointId** (*Optional[str]*) –
- **OwnershipDeedId** (*Optional[int]*) –
- **OwnershipDeedValidatorAddr** (*Optional[str]*) –
- **OwnerAddr** (*Optional[str]*) –
- **DaemonAddr** (*Optional[str]*) –
- **TradingRightsId** (*Optional[int]*) –
- **ScadaAlgoAddr** (*Optional[str]*) –
- **ScadaCertId** (*Optional[int]*) –
- **ComponentId** (*Optional[str]*) –
- **DisplayName** (*Optional[str]*) –
- **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 gwmm.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 gwmm.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 gwmm.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 gwmm.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** (*Optional[str]*) –
- **gps_point_id** (*Optional[str]*) –
- **ownership_deed_id** (*Optional[int]*) –
- **ownership_deed_validator_addr** (*Optional[str]*) –
- **owner_addr** (*Optional[str]*) –
- **daemon_addr** (*Optional[str]*) –
- **trading_rights_id** (*Optional[int]*) –
- **scada_algo_addr** (*Optional[str]*) –
- **scada_cert_id** (*Optional[int]*) –
- **component_id** (*Optional[str]*) –
- **display_name** (*Optional[str]*) –

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

1.4.4 HeartbeatA

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

class gwmm.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 gwmm.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 gwmm.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

1.4.5 LatestPrice

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

class gwmm.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** (*Optional[str]*) –
- **MessageId** (*Optional[str]*) –
- **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 gwmm.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 gwmm.types.latest_price.**check_is_iso_format**(*v*)

Parameters
v (*str*) –

class gwmm.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 gwmm.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 gwmm.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** (*Optional[str]*) –
- **message_id** (*Optional[str]*) –

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

1.4.6 MarketBook

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

class gwmm.types.**MarketBook**(*, *Slot*, *Bids*, *TypeName='market.book'*, *Version='000'*)

MarketMaker's list of bids for a MarketSlot

Parameters

- **Slot** (MarketSlot) –
- **Bids** (*List[AcceptedBid]*) –
- **TypeName** (*Literal['market.book']*) –
- **Version** (*str*) –

Slot:

- Description: MarketSlot the book is for

Bids:

- Description: List of bids in the book

class gwmm.types.**MarketBook_Maker**(*slot*, *bids*)

Parameters

- **slot** (MarketSlot) –
- **bids** (*List[*AcceptedBid*]*) –

classmethod **tuple_to_type**(*tuple*)

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

Parameters
tuple (MarketBook) –

Return type
str

classmethod **type_to_tuple**(*t*)

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

Parameters
t (*str*) –

Return type
MarketBook

1.4.7 MarketMakerInfo

Python pydantic class corresponding to json type `market.maker.info`.

class gwmm.types.**MarketMakerInfo**(***, *GNodeAlias*, *MarketTypeList*, *SampleMarketName*, *SampleMarketSlotName*, *TypeName='market.maker.info'*, *Version='000'*)

Parameters

- **GNodeAlias** (*str*) –
- **MarketTypeList** (*List[*MarketTypeGt*]*) –
- **SampleMarketName** (*str*) –
- **SampleMarketSlotName** (*str*) –
- **TypeName** (*Literal['market.maker.info']*) –
- **Version** (*str*) –

GNodeAlias:

- Description:
- Format: LeftRightDot

MarketTypeList:

- Description:

SampleMarketName:

- Description:

SampleMarketSlotName:

- Description:
- Format: MarketSlotNameLrdFormat

class gwmm.types.**MarketMakerInfo_Maker**(*g_node_alias*, *market_type_list*, *sample_market_name*, *sample_market_slot_name*)

Parameters

- **g_node_alias** (*str*) –
- **market_type_list** (*List[MarketTypeGt]*) –
- **sample_market_name** (*str*) –
- **sample_market_slot_name** (*str*) –

classmethod tuple_to_type(*tuple*)

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

Parameters
tuple (MarketMakerInfo) –

Return type
str

classmethod type_to_tuple(*t*)

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

Parameters
t (*str*) –

Return type
MarketMakerInfo

1.4.8 MarketPrice

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

class gwmm.types.**MarketPrice**(*, *ValueTimes1000*, *Unit*, *TypeName='market.price'*, *Version='000'*)

Parameters

- **ValueTimes1000** (*int*) –
- **Unit** (MarketPriceUnit) –
- **TypeName** (*Literal['market.price']*) –
- **Version** (*str*) –

ValueTimes1000:

- Description:

Unit:

- Description:

class gwmm.types.**MarketPrice_Maker**(*value_times1000*, *unit*)

Parameters

- **value_times1000** (*int*) –
- **unit** (MarketPriceUnit) –

classmethod tuple_to_type(*tuple*)

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

Parameters
tuple (MarketPrice) –

Return type
str

classmethod type_to_tuple(*t*)

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

Parameters
t (*str*) –

Return type
MarketPrice

1.4.9 MarketSlot

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

class gwmm.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 gwmm.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 gwmm.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 gwmm.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

1.4.10 MarketTypeGt

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

class gwmm.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 gwmm.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

1.4.11 PriceQuantity

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

class gwmm.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 gwmm.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

1.4.12 PriceQuantityUnitless

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

class gwmm.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 gwmm.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

1.4.13 Ready

Python pydantic class corresponding to json type `ready`.

class gwmm.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 gwmm.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 gwmm.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 gwmm.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

1.4.14 SimTimestep

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

class gwmm.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 gwmm.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 gwmm.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 gwmm.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 gwmm.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 gwmm.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

1.5 MarketMaker API

1.6 MarketMakerBase

This is the base class for GNodes, used to communicate via RabbitMQ

1.7 Contributor Guide

Thank you for your interest in improving this project. This project is open-source under the MIT license and 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 Python 3.10+ 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 gwmm
```

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.
- 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

```
MIT License

Copyright © 2022 Jessica Millar

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

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

(continues on next page)

(continued from previous page)

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

PYTHON MODULE INDEX

g

INDEX

H

L

M

P

R

S

T

U

V