
GoodData Foreign Data Wrapper

Release 1.1.0

GoodData Corporation

Sep 08, 2022

CONTENTS:

- 1 Installation 3**
 - 1.1 Requirements 3
 - 1.2 Install Using Docker (Recommended) 3
 - 1.3 Install Using Pip 5
- 2 Get Started With PostgreSQL 7**
 - 2.1 Connect to PostgreSQL 7
- 3 Foreign Tables 9**
 - 3.1 Import GoodData Objects into PostgreSQL Schema 9
 - 3.2 Create Foreign Tables 10
 - 3.3 Push Down of Filters 11
 - 3.4 Known Limitations 11
- 4 API 13**
 - 4.1 gooddata_fdw 13
 - 4.2 gooddata_sdk 32
- Python Module Index 219**
- Index 221**

GoodData Foreign Data Wrapper delivers PostgreSQL foreign data wrapper extension built on top of [multicorn](#). The extension makes GoodData.CN insights, computations and ad-hoc report data available in PostgreSQL as tables. It can be selected like any other table using the SQL language.

INSTALLATION

You can build and run it as a service in your Docker environment (recommended) or install the package on your system directly using pip.

1.1 Requirements

- Python 3.7 or newer
- GoodData.CN or GoodData Cloud

1.2 Install Using Docker (Recommended)

The Python SDK comes with a Dockerfile which, when started, will run PostgreSQL 12 with multicorn and `gooddata-fdw` pre-installed. For an even better user experience there is a `docker-compose.yaml` file which contains both the `gooddata-fdw` and `gooddata-cn-ce` services.

1.2.1 Build and Run the Service

Execute the following command in your repository root folder:

```
docker-compose up --build -d
```

`gooddata-fdw` image is built from the Dockerfile and both services are started in background.

Note: Services in `docker-compose.yaml` contain a setup of various environment variables including `POSTGRES_PASSWORD`. Feel free to set the variables in your environment, before you execute the above command. Default value for `POSTGRES_PASSWORD` is `gooddata123`.

1.2.2 Maintenance

To rebuild the Foreign Data Wrapper image execute the following command:

```
docker-compose build
```

If you would like to purge a container completely (including the volume) and start from scratch, run the following helper scripts:

```
./rebuild.sh gooddata-cn-ce  
./rebuild.sh gooddata-fdw
```

1.2.3 Adding Your Own Data

Before you start playing with the Foreign Data Wrapper, you will need a content in the `gooddata-cn-ce`.

`docker-compose.yaml` launches the *upload-layout* service. Its purpose is to bootstrap the demo and testing content into `gooddata-cn-ce`. You can use this as a starting point.

But the `gooddata-cn-ce` service is not limited only to the demo content. You can fill the `gooddata-cn-ce` with your own content (LDM, metrics, insights). Follow our [Getting Started documentation](#) if you need help with that.

1.2.4 Connect with existing GoodData.CN installation

This use case is for users running a GoodData.CN image who want to connect it to the GoodData Foreign Data Wrapper. For connecting `gooddata-fdw` with GoodData.CN image both images have to run on the same network. You can create a new network and run both images there or use the default bridge network.

Note: Default network bridge does not support accessing services by their name. You need to use an IP address in the host when defining the GoodData.CN server. The IP address can be found using command `docker inspect <GoodData.CN container name>`.

1. Build the `gooddata-fdw` service from `docker-compose.yaml`:

```
docker-compose build gooddata-fdw
```

2. Create a new network:

```
docker network create --driver bridge gd-cn-net
```

3. Run GoodData.CN on created network and name it `gooddata-cn-ce`:

```
docker run --rm --name gooddata-cn-ce -p 3000:3000 -p 5432:5432 -v /data \\\n--network gd-cn-net \\\n-e LICENSE_AND_PRIVACY_POLICY_ACCEPTED=YES \\\n-e APP_LOGLEVEL=INFO \\\ngooddata/gooddata-cn-ce:latest
```

4. Run the `gooddata-fdw` service on created network and name it `postgres-fdw`:


```
docker run --rm --name postgres-fdw -p 2543:5432 --network gd-cn-net \
-e POSTGRES_DB=gooddata -e POSTGRES_USER=gooddata -e POSTGRES_PASSWORD=gooddata123 \
gooddata-python-sdk_gooddata-fdw:latest \
postgres -c "shared_preload_libraries=foreign_table_exposer" -c "log_statement=all" \
→ -c "client_min_messages=DEBUG1" -c "log_min_messages=DEBUG1"
```

1.3 Install Using Pip

Run the following command to install the gooddata-fdw package on your system:

```
pip install gooddata-fdw
```

Warning: For this use case, you also need to install and run PostgreSQL together with multicorner.

GET STARTED WITH POSTGRESQL

2.1 Connect to PostgreSQL

After the `gooddata-fdw` container starts, you can connect to the running PostgreSQL:

- From console using `psql --host localhost --port 2543 --user gooddata gooddata`
You will be asked to enter the password that you have specified when starting the script.
- From any other client using JDBC string: `jdbc:postgresql://localhost:2543/gooddata`
You will be asked to enter username (`gooddata`) and password.

Once connected you will be able to work with the GoodData.CN Foreign Data Wrapper. At first, you need to define your GoodData.CN server in PostgreSQL:

```
CREATE SERVER multicorn_gooddata FOREIGN DATA WRAPPER multicorn
OPTIONS (
  wrapper 'gooddata_fdw.GoodDataForeignDataWrapper',
  host 'https://gooddata-cn-ce:3000', -- host equal to name of container with GoodData.
  ↪CN.CE
  token 'YWRtaW46Ym9vdHN0cmFwOmFkbWluMTIz' -- default gooddata-cn-ce token, documented ↪
  ↪in public DOC as well
);
```

As of now the GoodData.CN community edition (single container deployment) supports only `localhost` as the target host. If you spin-up GoodData.CN and FDW using `docker-compose`, GoodData.CN host name is the service name in the `docker-compose`, e.g. `gooddata-cn-ce`. To enable such setup, we provide an option `header_host`:

```
CREATE SERVER multicorn_gooddata FOREIGN DATA WRAPPER multicorn
OPTIONS (
  wrapper 'gooddata_fdw.GoodDataForeignDataWrapper',
  host 'http://gooddata-cn-ce:3000', -- host equal to name of container with GoodData.
  ↪CN.CE
  token 'YWRtaW46Ym9vdHN0cmFwOmFkbWluMTIz', -- default gooddata-cn-ce token, ↪
  ↪documented in public DOC as well
  headers_host 'localhost'
);
```

Typically, you have to do this once per GoodData.CN installation. You may add as many servers as you need.

IMPORTANT: Do not forget to specify host including the schema (`http` or `https`).

FOREIGN TABLES

3.1 Import GoodData Objects into PostgreSQL Schema

You can import insights created in GoodData.CN Analytical Designer as PostgreSQL foreign tables. You can import insights from as many workspaces and/or GoodData.CN instances (servers) as you want.

You can also import your entire semantic model including MAQL metrics into a special *compute pseudo-table*. Doing SELECTs from this table will trigger computation of analytics on your GoodData.CN server based on the columns that you have specified on the SELECT.

Note: The *compute* is called pseudo-table for a reason. It does not adhere to the relational model. The columns that you SELECT map to facts, metrics and labels in your semantic model. Computing results for the select will automatically aggregate results on the columns that are mapped to labels in your semantic model. In other words cardinality of the *compute* table changes based on the columns that you SELECT.

For your convenience we prepared a stored procedure, which:

- (re)creates target schema
- imports currently existing insights and/or entire semantic model

You can re-execute the procedure to update foreign tables.

```
-- This maps all insights stored in GoodData.CN workspace `workspace_id` into the
-- PostgreSQL schema named `workspace_id`
CALL import_gooddata('workspace_id', 'insights');

-- By utilizing the third parameter you can override the name of the target PostgreSQL
-- schema
CALL import_gooddata('workspace_id', 'insights', 'custom_schema');

-- This imports the semantic model into the 'compute' pseudo-table.
CALL import_gooddata('workspace_id', 'compute');

-- This imports both insights and compute
CALL import_gooddata('workspace_id', 'all');

-- This is how you can extend max size of numeric columns in foreign tables (basically
-- to support larger numbers)
CALL import_gooddata(workspace := 'goodsales', object_type := 'all', numeric_max_size :=
-- 24);
```

(continues on next page)

(continued from previous page)

```
-- Specify custom foreign server name - this enables you importing from multiple servers.
↳ into the same FDW instance
CALL import_gooddata(workspace := 'goodsales', object_type := 'all', foreign_server :=
↳ 'multicorn_gooddata_stg');
```

Default max numeric size is 18, default digits after decimal point is 2 unless metric format defines more.

You will get a couple of 'NOTICE' messages as the import progresses. You can then check the imported tables by executing:

```
SELECT * FROM information_schema.foreign_tables WHERE foreign_table_schema = 'workspace_
↳ id';
```

IMPORTANT: Your semantic model may consist of multiple isolated segments that have no relationship between them. Attempting to compute results from multiple isolated segments will result in errors.

Warning: Imported tables reflect state of the workspace and insights in time of import. Any later change to the workspace can result in failing SQL queries against imported tables. The state can be fixed by re-importing the workspace insights and/or compute.

3.2 Create Foreign Tables

You can manually create your own foreign tables and map their columns to GoodData.CN semantic model. This is similar to creating normal tables except you have to provide table and column OPTIONS to establish the correct mapping. For instance:

```
CREATE FOREIGN TABLE custom_report (
  some_label VARCHAR OPTIONS (id 'label/some_label'),
  some_fact_sum NUMERIC(15,5) OPTIONS (id 'fact/some_fact', agg 'sum'),
  some_fact_avg NUMERIC(15,5) OPTIONS (id 'fact/some_fact', agg 'avg'),
  some_metric NUMERIC(15,5) OPTIONS (id 'metric/some_metric')
)
SERVER multicorn_gooddata
OPTIONS ( workspace 'workspace_id');
```

To explain:

- OPTIONS on foreign table must contain identifier of workspace to map to
- OPTIONS on each column must contain identifier of semantic model entity. The id is string but consisting of two parts <entity_type>/<entity_id>. Where entity_type is either label, fact or metric.

For columns that map to facts in your semantic model, you can also specify what aggregation function should be used when aggregating the fact values for the labels in your custom report table. You can use the following aggregation functions:

- sum
- avg
- min
- max

- median

The `agg` key is optional. If you do not specify it, then default `sum` aggregation will be used. The value of `agg` is case insensitive.

Note: If you do not specify the required options, the `CREATE` command will fail. If you specify wrong entity IDs, the failures will happen at `SELECT` time.

3.3 Push Down of Filters

When querying foreign tables, you can add `WHERE` clause filtering the result. For performance optimization, it makes sense to push such filters down to the GoodData.CN, so not all data has to be collected.

We are able to push only some filters down to GoodData.CN:

- Simple attribute(label) filters
 - Example: `WHERE region IN ('East', 'West')`
- Simple date filters
 - Only DAY granularity is supported
 - (NOT) IN operator is not supported
 - Example: `WHERE my_date BETWEEN '2021-01-01' AND '2021-02-01'`

If you use an `OR` between conditions, it is not pushed down. Push down is possible in case of custom tables and `compute` table, not in case of foreign tables imported from `insights`.

3.4 Known Limitations

It is not possible to reference a column in `WHERE` clause, which is not used in `SELECT` section. Example:

```
SELECT label1, metric FROM insight WHERE label2 = 'a';  
SELECT label1, metric FROM compute WHERE label2 = 'a';
```

While it is obvious in case of an `insight` (it does not contain the column at all), in case of `compute` we would like to support it, but we are not allowed due to lack of functionality in Multicorn - the filter is always applied on final result set and if it does not contain the column, it does not work.

gooddata_fdw

gooddata_sdk

The *gooddata-sdk* package aims to provide clean and convenient Python APIs to interact with GoodData.CN.

4.1 gooddata_fdw

Modules

gooddata_fdw.column_utils

gooddata_fdw.column_validation

gooddata_fdw.environment

This file exists because multicorn is not available as proper stand-alone python package that one could install and then use the different data types during testing.

gooddata_fdw.executor

gooddata_fdw.fdw

gooddata_fdw.filter

gooddata_fdw.import_workspace

gooddata_fdw.naming

gooddata_fdw.options

gooddata_fdw.pg_logging

gooddata_fdw.result_reader

4.1.1 gooddata_fdw.column_utils

Functions

<code>column_data_type_for(attribute)</code>	Determine what postgres type should be used for <i>attribute</i> .
<code>table_col_as_computable(col)</code>	

gooddata_fdw.column_utils.column_data_type_for

`gooddata_fdw.column_utils.column_data_type_for(attribute: Optional[CatalogAttribute]) → str`

Determine what postgres type should be used for *attribute*.

Parameters

attribute – catalog attribute instance

gooddata_fdw.column_utils.table_col_as_computable

`gooddata_fdw.column_utils.table_col_as_computable(col: ColumnDefinitionStub) → Union[Attribute, Metric]`

4.1.2 gooddata_fdw.column_validation

Functions

<code>validate_columns_in_table_def(table_columns, ...)</code>
--

gooddata_fdw.column_validation.validate_columns_in_table_def

`gooddata_fdw.column_validation.validate_columns_in_table_def(table_columns: dict[str, ColumnDefinition], query_columns: list[str]) → None`

Classes

<code>ColumnValidator()</code>
<code>IdOptionValidator(mandatory)</code>
<code>LocalIdOptionValidator()</code>

gooddata_fdw.column_validation.ColumnValidator**class** gooddata_fdw.column_validation.ColumnValidator

Bases: object

`__init__()`**Methods**

`__init__()`

`validate(column_name, column_def)`

gooddata_fdw.column_validation.IdOptionValidator**class** gooddata_fdw.column_validation.IdOptionValidator(*mandatory: bool*)Bases: *ColumnValidator*`__init__(mandatory: bool)`**Methods**

`__init__(mandatory)`

`validate(column_name, column_def)`

gooddata_fdw.column_validation.LocalIdOptionValidator**class** gooddata_fdw.column_validation.LocalIdOptionValidatorBases: *ColumnValidator*`__init__()`**Methods**

`__init__()`

`validate(column_name, column_def)`

4.1.3 gooddata_fdw.environment

This file exists because multicorn is not available as proper stand-alone python package that one could install and then use the different data types during testing.

The multicorn python code is part of the PostgreSQL extension installation.

Thus here is the layer of indirection that tries to import multicorn code and if that is not present (likely during test run) it will use stub implementations.

The stubbing only happens if the FDW code is called during test execution. Otherwise the import error is raised as usual to prevent some wicked behavior on mis-configured PostgreSQL.

Functions

log_to_postgres(msg, level)

gooddata_fdw.environment.log_to_postgres

`gooddata_fdw.environment.log_to_postgres(msg: str, level: int) → None`

Classes

ColumnDefinition alias of *ColumnDefinitionStub*

ColumnDefinitionStub(column_name, type_name,
...)

ForeignDataWrapper alias of *ForeignDataWrapperStub*

ForeignDataWrapperStub(options, columns)

Qual alias of *QualStub*

QualStub(field_name, operator, value)

TableDefinition alias of *TableDefinitionStub*

TableDefinitionStub(table_name, columns, op-
tions)

gooddata_fdw.environment.ColumnDefinition

`gooddata_fdw.environment.ColumnDefinition`

alias of *ColumnDefinitionStub*

gooddata_fdw.environment.ColumnDefinitionStub

```
class gooddata_fdw.environment.ColumnDefinitionStub(column_name: str, type_name: str, options:
                                                    dict[str, str])
```

Bases: object

```
__init__(column_name: str, type_name: str, options: dict[str, str]) → None
```

Methods

```
__init__(column_name, type_name, options)
```

gooddata_fdw.environment.ForeignDataWrapper

```
gooddata_fdw.environment.ForeignDataWrapper
```

alias of *ForeignDataWrapperStub*

gooddata_fdw.environment.ForeignDataWrapperStub

```
class gooddata_fdw.environment.ForeignDataWrapperStub(options: dict[str, str], columns: dict[str,
                                                                 ColumnDefinition])
```

Bases: object

```
__init__(options: dict[str, str], columns: dict[str, ColumnDefinition]) → None
```

Methods

```
__init__(options, columns)
```

```
execute(quals, columns[, sortkeys])
```

```
import_schema(schema, srv_options, options, ...)
```

gooddata_fdw.environment.Qual

```
gooddata_fdw.environment.Qual
```

alias of *QualStub*

gooddata_fdw.environment.QualStub

class gooddata_fdw.environment.**QualStub**(*field_name: str, operator: Union[str, tuple[str, str]], value: Any*)

Bases: object

__init__(*field_name: str, operator: Union[str, tuple[str, str]], value: Any*) → None

Methods

__init__(*field_name, operator, value*)

gooddata_fdw.environment.TableDefinition

gooddata_fdw.environment.**TableDefinition**

alias of *TableDefinitionStub*

gooddata_fdw.environment.TableDefinitionStub

class gooddata_fdw.environment.**TableDefinitionStub**(*table_name: str, columns: list[ColumnDefinition], options: dict[str, str]*)

Bases: object

__init__(*table_name: str, columns: list[ColumnDefinition], options: dict[str, str]*) → None

Methods

__init__(*table_name, columns, options*)

4.1.4 gooddata_fdw.executor

Classes

ComputeExecutor(inputs)

CustomExecutor(inputs)

Executor(inputs, column_validators)

ExecutorFactory()

InitData(sdk, server_options, table_options, ...)

InsightExecutor(inputs)

gooddata_fdw.executor.ComputeExecutor

class gooddata_fdw.executor.**ComputeExecutor**(inputs: *InitData*)

Bases: *Executor*

__init__(inputs: *InitData*) → None

Methods

__init__(inputs)

can_react(inputs)

execute(quals, columns[, sort_keys])

validate_columns_def()

gooddata_fdw.executor.CustomExecutor

class gooddata_fdw.executor.**CustomExecutor**(inputs: *InitData*)

Bases: *Executor*

__init__(inputs: *InitData*) → None

Methods

__init__(inputs)

can_react(inputs)

execute(quals, columns[, sort_keys])

validate_columns_def()

gooddata_fdw.executor.Executor

class gooddata_fdw.executor.**Executor**(inputs: *InitData*, column_validators:
list[col_val.ColumnValidator])

Bases: object

__init__(inputs: *InitData*, column_validators: list[col_val.ColumnValidator]) → None

Methods

`__init__(inputs, column_validators)`

`can_react(inputs)`

`execute(quals, columns[, sort_keys])`

`validate_columns_def()`

gooddata_fdw.executor.ExecutorFactory

class gooddata_fdw.executor.**ExecutorFactory**

Bases: object

`__init__()`

Methods

`__init__()`

`create(inputs)`

gooddata_fdw.executor.InitData

class gooddata_fdw.executor.**InitData**(*sdk, server_options, table_options, columns*)

Bases: tuple

`__init__()`

Methods

`__init__()`

<code>count(value, /)</code>	Return number of occurrences of value.
------------------------------	--

<code>index(value[, start, stop])</code>	Return first index of value.
--	------------------------------

Attributes

<code>columns</code>	Alias for field number 3
<code>sdk</code>	Alias for field number 0
<code>server_options</code>	Alias for field number 1
<code>table_options</code>	Alias for field number 2

property columns

Alias for field number 3

count(value, /)

Return number of occurrences of value.

index(value, start=0, stop=9223372036854775807, /)

Return first index of value.

Raises ValueError if the value is not present.

property sdk

Alias for field number 0

property server_options

Alias for field number 1

property table_options

Alias for field number 2

gooddata_fdw.executor.InsightExecutor

class gooddata_fdw.executor.InsightExecutor(inputs: InitData)

Bases: [Executor](#)

__init__(inputs: InitData) → None

Methods

<code>__init__(inputs)</code>
<code>can_react(inputs)</code>
<code>execute(quals, columns[, sort_keys])</code>
<code>validate_columns_def()</code>

4.1.5 gooddata_fdw.fdw

Module Attributes

USER_AGENT

Extra segment of the User-Agent header that will be appended to standard gooddata-sdk user agent.

gooddata_fdw.fdw.USER_AGENT

`gooddata_fdw.fdw.USER_AGENT = 'gooddata-fdw/1.1.0'`

Extra segment of the User-Agent header that will be appended to standard gooddata-sdk user agent.

Classes

GoodDataForeignDataWrapper(options, columns)

gooddata_fdw.fdw.GoodDataForeignDataWrapper

class gooddata_fdw.fdw.GoodDataForeignDataWrapper(*options: dict[str, str], columns: dict[str, ColumnDefinition]*)

Bases: *ForeignDataWrapperStub*

__init__(*options: dict[str, str], columns: dict[str, ColumnDefinition]*) → None

Methods

__init__(options, columns)

delete(oldvalues)

execute(quals, columns[, sortkeys])

import_schema(schema, srv_options, options, ...)

insert(values)

update(oldvalues, newvalues)

Attributes

rowid_column

4.1.6 gooddata_fdw.filter

Functions

<code>extract_filters_from_qual</code> s(quals, table_columns)	Convert quals to filters.
--	---------------------------

gooddata_fdw.filter.extract_filters_from_qual

`gooddata_fdw.filter.extract_filters_from_qual`(quals: list[Qual], table_columns: dict[str, ColumnDefinition]) → list[Filter]

Convert quals to filters. Now only simple attribute filters are supported.

Parameters

- **quals** – multicorner quals representing filters in SQL WHERE clause
- **table_columns** – list of table columns

Returns

list of filters

4.1.7 gooddata_fdw.import_workspace

Classes

`ImporterInitData`(sdk, workspace, ...)

`InsightsWorkspaceImporter`(data)

`SemanticLayerWorkspaceImporter`(data)

`WorkspaceImporter`(data)

`WorkspaceImportersLocator`()

gooddata_fdw.import_workspace.ImporterInitData

```
class gooddata_fdw.import_workspace.ImporterInitData(sdk, workspace, server_options,
                                                    import_options, restriction_type, restricts)
```

Bases: tuple

`__init__()`

Methods

<code>__init__()</code>	
<code>count(value, /)</code>	Return number of occurrences of value.
<code>index(value[, start, stop])</code>	Return first index of value.

Attributes

<code>import_options</code>	Alias for field number 3
<code>restriction_type</code>	Alias for field number 4
<code>restricts</code>	Alias for field number 5
<code>sdk</code>	Alias for field number 0
<code>server_options</code>	Alias for field number 2
<code>workspace</code>	Alias for field number 1

count(value, /)

Return number of occurrences of value.

property import_options

Alias for field number 3

index(value, start=0, stop=9223372036854775807, /)

Return first index of value.

Raises ValueError if the value is not present.

property restriction_type

Alias for field number 4

property restricts

Alias for field number 5

property sdk

Alias for field number 0

property server_options

Alias for field number 2

property workspace

Alias for field number 1

gooddata_fdw.import_workspace.InsightsWorkspaceImporter**class** gooddata_fdw.import_workspace.InsightsWorkspaceImporter(*data*: ImporterInitData)Bases: *WorkspaceImporter***__init__**(*data*: ImporterInitData) → None**Methods**

__init__(*data*)

import_tables()

support_object_type(*object_type*)

gooddata_fdw.import_workspace.SemanticLayerWorkspaceImporter**class** gooddata_fdw.import_workspace.SemanticLayerWorkspaceImporter(*data*: ImporterInitData)Bases: *WorkspaceImporter***__init__**(*data*: ImporterInitData) → None**Methods**

__init__(*data*)

import_tables()

support_object_type(*object_type*)

gooddata_fdw.import_workspace.WorkspaceImporter**class** gooddata_fdw.import_workspace.WorkspaceImporter(*data*: ImporterInitData)

Bases: object

__init__(*data*: ImporterInitData) → None

Methods

`__init__(data)`

`import_tables()`

`support_object_type(object_type)`

`gooddata_fdw.import_workspace.WorkspaceImportersLocator`

class `gooddata_fdw.import_workspace.WorkspaceImportersLocator`

Bases: `object`

`__init__()`

Methods

`__init__()`

`locate(object_type)`

`register(class_)`

4.1.8 `gooddata_fdw.naming`

Classes

`CatalogNamingStrategy()`

`DefaultCatalogNamingStrategy()`

`DefaultInsightColumnNaming()`

`DefaultInsightTableNameNaming()`

`InsightColumnNamingStrategy()`

`InsightTableNameNamingStrategy()`

gooddata_fdw.naming.CatalogNamingStrategy**class** gooddata_fdw.naming.CatalogNamingStrategy

Bases: object

__init__()**Methods****__init__**()

col_name_for_fact(attr)

col_name_for_label(attr)

col_name_for_metric(attr)

gooddata_fdw.naming.DefaultCatalogNamingStrategy**class** gooddata_fdw.naming.DefaultCatalogNamingStrategy

Bases: object

__init__() → None**Methods****__init__**()

col_name_for_fact(fact, dataset)

col_name_for_label(label, dataset)

col_name_for_metric(metric)

gooddata_fdw.naming.DefaultInsightColumnNaming**class** gooddata_fdw.naming.DefaultInsightColumnNamingBases: *InsightColumnNamingStrategy***__init__**() → None

Methods

`__init__()`

`col_name_for_attribute(attr)`

`col_name_for_metric(metric)`

`gooddata_fdw.naming.DefaultInsightTableNaming`

class `gooddata_fdw.naming.DefaultInsightTableNaming`

Bases: *InsightTableNamingStrategy*

`__init__()` → None

Methods

`__init__()`

`table_name_for_insight(insight)`

`gooddata_fdw.naming.InsightColumnNamingStrategy`

class `gooddata_fdw.naming.InsightColumnNamingStrategy`

Bases: `object`

`__init__()`

Methods

`__init__()`

`col_name_for_attribute(attr)`

`col_name_for_metric(attr)`

gooddata_fdw.naming.InsightTableNamingStrategy**class** gooddata_fdw.naming.InsightTableNamingStrategy

Bases: object

__init__()**Methods****__init__**()

table_name_for_insight(insight)

4.1.9 gooddata_fdw.options**Classes***BaseOptions*([validate, skip_attributes])*ImportSchemaOptions*(options)*ServerOptions*(options)*TableOptions*(options)**gooddata_fdw.options.BaseOptions****class** gooddata_fdw.options.BaseOptions(validate: bool = True, skip_attributes: Optional[list[str]] = None)

Bases: object

__init__(validate: bool = True, skip_attributes: Optional[list[str]] = None) → None**Methods****__init__**([validate, skip_attributes])

gooddata_fdw.options.ImportSchemaOptions

class gooddata_fdw.options.**ImportSchemaOptions**(options: dict[str, str])

Bases: *BaseOptions*

__init__(options: dict[str, str]) → None

Methods

__init__(options)

metric_data_type([precision])

Attributes

METRIC_DIGITS_AFTER_DEC_POINT_DEFAULT

METRIC_DIGITS_BEFORE_DEC_POINT_DEFAULT

numeric_max_size

object_type

gooddata_fdw.options.ServerOptions

class gooddata_fdw.options.**ServerOptions**(options: dict[str, str])

Bases: *BaseOptions*

__init__(options: dict[str, str]) → None

Methods

__init__(options)

Attributes

headers_host

host

token

gooddata_fdw.options.TableOptions

class gooddata_fdw.options.**TableOptions**(options: dict[str, str])

Bases: *BaseOptions*

__init__(options: dict[str, str]) → None

Methods

__init__(options)

Attributes

compute

insight

workspace

4.1.10 gooddata_fdw.pg_logging**4.1.11 gooddata_fdw.result_reader****Classes**

InsightTableResultReader(table_columns, ...)

TableResultReader(table_columns)

gooddata_fdw.result_reader.InsightTableResultReader

class gooddata_fdw.result_reader.**InsightTableResultReader**(table_columns: dict[str, ColumnDefinition], query_columns: list[str])

Bases: *TableResultReader*

__init__(table_columns: dict[str, ColumnDefinition], query_columns: list[str]) → None

Methods

`__init__(table_columns, query_columns)`

`read_all_rows(table)`

`gooddata_fdw.result_reader.TableResultReader`

class `gooddata_fdw.result_reader.TableResultReader`(*table_columns: dict[str, ColumnDefinition]*)

Bases: `object`

`__init__(table_columns: dict[str, ColumnDefinition])` → `None`

Methods

`__init__(table_columns)`

`read_all_rows(table)`

4.2 gooddata_sdk

The *gooddata-sdk* package aims to provide clean and convenient Python APIs to interact with GoodData.CN.

At the moment the SDK provides services to inspect and interact with the Semantic Model and consume analytics.

Modules

`gooddata_sdk.catalog`

<code>gooddata_sdk.client</code>	Module containing a class that provides access to meta-data and afm services.
----------------------------------	---

`gooddata_sdk.compute`

`gooddata_sdk.insight`

`gooddata_sdk.sdk`

`gooddata_sdk.support`

`gooddata_sdk.table`

`gooddata_sdk.type_converter`

`gooddata_sdk.utils`

4.2.1 gooddata_sdk.catalog

Modules

gooddata_sdk.catalog.base

gooddata_sdk.catalog.catalog_service_base

gooddata_sdk.catalog.data_source

gooddata_sdk.catalog.entity

gooddata_sdk.catalog.identifier

gooddata_sdk.catalog.organization

gooddata_sdk.catalog.permission

gooddata_sdk.catalog.setting

gooddata_sdk.catalog.types

gooddata_sdk.catalog.user

gooddata_sdk.catalog.workspace

gooddata_sdk.catalog.base

Functions

value_in_allowed(instance, attribute, value)

gooddata_sdk.catalog.base.value_in_allowed

`gooddata_sdk.catalog.base.value_in_allowed(instance: Type[Base], attribute: Attribute, value: str) → None`

Classes

`Base()`

gooddata_sdk.catalog.base.Base

class `gooddata_sdk.catalog.base.Base`

Bases: object

`__init__()` → None

Method generated by attrs for class Base.

Methods

<code>__init__()</code>	Method generated by attrs for class Base.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.catalog_service_base**Classes**

CatalogServiceBase(api_client)

gooddata_sdk.catalog.catalog_service_base.CatalogServiceBase

```
class gooddata_sdk.catalog.catalog_service_base.CatalogServiceBase(api_client:
                                                                    GoodDataApiClient)
```

Bases: object

__init__(*api_client*: GoodDataApiClient) → None

Methods

__init__(api_client)

get_organization()

layout_organization_folder(layout_root_path)

Attributes

organization_id

gooddata_sdk.catalog.data_source**Modules**

gooddata_sdk.catalog.data_source.
action_requests

gooddata_sdk.catalog.data_source.
declarative_model

gooddata_sdk.catalog.data_source.
entity_model

gooddata_sdk.catalog.data_source.service

gooddata_sdk.catalog.data_source.
validation

gooddata_sdk.catalog.data_source.action_requests

Modules

*gooddata_sdk.catalog.data_source.
action_requests.ldm_request*

*gooddata_sdk.catalog.data_source.
action_requests.scan_model_request*

gooddata_sdk.catalog.data_source.action_requests.ldm_request

Classes

CatalogGenerateLdmRequest([, separator, ...])*

gooddata_sdk.catalog.data_source.action_requests.ldm_request.CatalogGenerateLdmRequest


```

class gooddata_sdk.catalog.data_source.action_requests.ldm_request.CatalogGenerateLdmRequest(*,
    sep-
    a-
    ra-
    tor:
    str
    =
    '___',
    gen-
    er-
    ate_long_ids:
    Op-
    tional[bool]
    =
    None,
    ta-
    ble_prefix:
    Op-
    tional[str]
    =
    None,
    view_prefix:
    Op-
    tional[str]
    =
    None,
    pri-
    mary_label_prefix:
    Op-
    tional[str]
    =
    None,
    sec-
    ondary_label_prefix:
    Op-
    tional[str]
    =
    None,
    fact_prefix:
    Op-
    tional[str]
    =
    None,
    date_granularity:
    Op-
    tional[str]
    =
    None,
    grain_prefix:
    Op-
    tional[str]
    =
    None,
    ref-
    er-
    ence_prefix:
    Op-
    tional[str]
    =
    None,

```

Bases: [Base](#)

```
__init__(* , separator: str = '__', generate_long_ids: Optional[bool] = None, table_prefix: Optional[str] = None, view_prefix: Optional[str] = None, primary_label_prefix: Optional[str] = None, secondary_label_prefix: Optional[str] = None, fact_prefix: Optional[str] = None, date_granularities: Optional[str] = None, grain_prefix: Optional[str] = None, reference_prefix: Optional[str] = None, grain_reference_prefix: Optional[str] = None, denorm_prefix: Optional[str] = None, wdf_prefix: Optional[str] = None) → None
```

Method generated by attrs for class CatalogGenerateLdmRequest.

Methods

<code>__init__(*[, separator, generate_long_ids, ...])</code>	Method generated by attrs for class CatalogGenerateLdmRequest.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

separator

generate_long_ids

table_prefix

view_prefix

primary_label_prefix

secondary_label_prefix

fact_prefix

date_granularities

grain_prefix

reference_prefix

grain_reference_prefix

denorm_prefix

wdf_prefix

classmethod **from_api**(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod **from_dict**(*data: Dict[str, Any], camel_case: bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.data_source.action_requests.scan_model_request

Functions

one_scan_true(instance, *args)

`gooddata_sdk.catalog.data_source.action_requests.scan_model_request.one_scan_true`

`gooddata_sdk.catalog.data_source.action_requests.scan_model_request.one_scan_true`(instance: CatalogScanModelRequest, *args: Any) → None

Classes

`CatalogScanModelRequest`(*[, separator, ...])

`gooddata_sdk.catalog.data_source.action_requests.scan_model_request.CatalogScanModelRequest`

```
class gooddata_sdk.catalog.data_source.action_requests.scan_model_request.CatalogScanModelRequest(*,
    separator: str = '__', scan_tables: bool = True, scan_views: bool = False, table_prefix: Optional[str] = None,
    view_prefix: Optional[str] = None) → None
```

Bases: `Base`

`__init__`(*[, separator: str = '__', scan_tables: bool = True, scan_views: bool = False, table_prefix: Optional[str] = None, view_prefix: Optional[str] = None) → None

Method generated by attrs for class `CatalogScanModelRequest`.

Methods

<code>__init__(*[, separator, scan_tables, ...])</code>	Method generated by attrs for class <code>CatalogScan-ModelRequest</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>separator</code>
<code>scan_tables</code>
<code>scan_views</code>
<code>table_prefix</code>
<code>view_prefix</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.data_source.declarative_model`

Modules

<code>gooddata_sdk.catalog.data_source.declarative_model.data_source</code>
<code>gooddata_sdk.catalog.data_source.declarative_model.physical_model</code>

`gooddata_sdk.catalog.data_source.declarative_model.data_source`

Classes

`CatalogDeclarativeDataSource(, id, type, ...)`*

`CatalogDeclarativeDataSources(, data_sources)`*

`gooddata_sdk.catalog.data_source.declarative_model.data_source.CatalogDeclarativeDataSource`

```
class gooddata_sdk.catalog.data_source.declarative_model.data_source.CatalogDeclarativeDataSource(*,
                                                    id:
                                                    str,
                                                    type:
                                                    str,
                                                    name:
                                                    str,
                                                    url:
                                                    str,
                                                    schema:
                                                    str,
                                                    enable_cache:
                                                    Optional[bool] =
                                                    None,
                                                    pdm:
                                                    Optional[bool] =
                                                    None,
                                                    cache_url:
                                                    Optional[str] =
                                                    None,
                                                    user_name:
                                                    Optional[str] =
                                                    None,
                                                    permissions:
                                                    List[CatalogDeclarativeDataSourcePermission] =
                                                    NOTHING)
```

Bases: [Base](#)

```
__init__(*, id: str, type: str, name: str, url: str, schema: str, enable_caching: Optional[bool] = None, pdm:
Optional[CatalogDeclarativeTables] = None, cache_path: Optional[List[str]] = None, username:
Optional[str] = None, permissions: List[CatalogDeclarativeDataSourcePermission] = NOTHING)
→ None
```

Method generated by attrs for class CatalogDeclarativeDataSource.

Methods

<code>__init__(*, id, type, name, url, schema[, ...])</code>	Method generated by attrs for class CatalogDeclarativeDataSource.
<code>client_class()</code>	
<code>data_source_folder(data_sources_folder, ...)</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(data_sources_folder, ...)</code>	
<code>store_to_disk(data_sources_folder)</code>	
<code>to_api([password, token, ...])</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.
<code>to_test_request([password, token])</code>	

Attributes

id
type
name
url
schema
enable_caching
pdm
cache_path
username
permissions

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.data_source.declarative_model.data_source.CatalogDeclarativeDataSources`

class `gooddata_sdk.catalog.data_source.declarative_model.data_source.CatalogDeclarativeDataSources(*, data_sources: List[CatalogDeclarativeDataSource])`

Bases: `Base`

__init__(**, data_sources: List[CatalogDeclarativeDataSource]*) → None

Method generated by attrs for class `CatalogDeclarativeDataSources`.

Methods

<code>__init__(*, data_sources)</code>	Method generated by attrs for class CatalogDeclarativeDataSources.
<code>client_class()</code>	
<code>data_sources_folder(layout_organization_folder)</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(layout_organization_folder)</code>	
<code>store_to_disk(layout_organization_folder)</code>	
<code>to_api([credentials])</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>data_sources</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict `(camel_case: bool = True) → Dict[str, Any]`

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.data_source.declarative_model.physical_model

Modules

<code>gooddata_sdk.catalog.data_source.declarative_model.physical_model.column</code>
<code>gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm</code>
<code>gooddata_sdk.catalog.data_source.declarative_model.physical_model.table</code>

`gooddata_sdk.catalog.data_source.declarative_model.physical_model.column`

Classes

CatalogDeclarativeColumn(*, name, data_type)

`gooddata_sdk.catalog.data_source.declarative_model.physical_model.column.CatalogDeclarativeColumn`

`class gooddata_sdk.catalog.data_source.declarative_model.physical_model.column.CatalogDeclarativeColumn`

Bases: *Base*

`__init__`(*, name: str, data_type: str, is_primary_key: Optional[bool] = None, referenced_table_id: Optional[str] = None, referenced_table_column: Optional[str] = None) → None

Method generated by attrs for class CatalogDeclarativeColumn.

Methods

<code>__init__(*, name, data_type[, ...])</code>	Method generated by attrs for class CatalogDeclarativeColumn.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>name</code>
<code>data_type</code>
<code>is_primary_key</code>
<code>referenced_table_id</code>
<code>referenced_table_column</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm`

Functions

<code>get_pdm_folder(data_source_folder)</code>

gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm.get_pdm_folder

gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm.get_pdm_folder(*data_source_folder*: Path) → Path

Classes

CatalogDeclarativeTables(*[, tables])

CatalogScanResultPdm(*[, pdm, warnings])

gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm.CatalogDeclarativeTables

class gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm.CatalogDeclarativeTables(*, *ta-*
ble
Lis
=
NO
IN

Bases: *Base*

__init__(*[, tables: List[CatalogDeclarativeTable] = NOTHING) → None
Method generated by attrs for class CatalogDeclarativeTables.

Methods

<i>__init__</i> (*[, tables])	Method generated by attrs for class CatalogDeclarativeTables.
client_class()	
<i>from_api</i> (entity)	Creates object from entity passed by client class, which represents it as dictionary.
<i>from_dict</i> (data[, camel_case])	Creates object from dictionary.
load_from_disk(data_source_folder)	
store_to_disk(data_source_folder)	
to_api()	
<i>to_dict</i> ([camel_case])	Converts object into dictionary.

Attributes

tables

classmethod **from_api**(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod **from_dict**(*data: Dict[str, Any], camel_case: bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm.CatalogScanResultPdm

```
class gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm.CatalogScanResultPdm(*,
                                                                                               pdm:
                                                                                               Cat-
                                                                                               a-
                                                                                               logDecl
                                                                                               a-
                                                                                               tiveTa-
                                                                                               bles
                                                                                               =
                                                                                               Cat-
                                                                                               a-
                                                                                               logDecl
                                                                                               a-
                                                                                               tiveTa-
                                                                                               bles(tab
                                                                                               warn-
                                                                                               ings:
                                                                                               List[Dic
                                                                                               =
                                                                                               NOTH-
                                                                                               ING)
```

Bases: [Base](#)

__init__(**, pdm: CatalogDeclarativeTables = CatalogDeclarativeTables(tables=[]), warnings: List[Dict] = NOTHING*) → None

Method generated by attrs for class CatalogScanResultPdm.

Methods

<code>__init__(*[, pdm, warnings])</code>	Method generated by attrs for class <code>CatalogScanResultPdm</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>pdm</code>
<code>warnings</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.data_source.declarative_model.physical_model.table`

Classes

<code>CatalogDeclarativeTable(*, id, type, path, ...)</code>
--

`gooddata_sdk.catalog.data_source.declarative_model.physical_model.table.CatalogDeclarativeTable`

```
class gooddata_sdk.catalog.data_source.declarative_model.physical_model.table.CatalogDeclarativeTable(*,
```

Bases: [Base](#)

```
__init__(*, id: str, type: str, path: List[str], columns: List[CatalogDeclarativeColumn], name_prefix:
Optional[str] = None) → None
```

Method generated by attrs for class CatalogDeclarativeTable.

Methods

<code>__init__</code> (*, id, type, path, columns[, ...])	Method generated by attrs for class CatalogDeclarativeTable.
<code>client_class()</code>	
<code>from_api</code> (entity)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (data[, camel_case])	Creates object from dictionary.
<code>load_from_disk</code> (table_file_path)	
<code>store_to_disk</code> (pdm_folder)	
<code>to_api</code> ()	
<code>to_dict</code> ([camel_case])	Converts object into dictionary.

Attributes

<code>id</code>
<code>type</code>
<code>path</code>
<code>columns</code>
<code>name_prefix</code>

classmethod **from_api**(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod **from_dict**(*data: Dict[str, Any], camel_case: bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.data_source.entity_model

Modules

*gooddata_sdk.catalog.data_source.
entity_model.content_objects*

*gooddata_sdk.catalog.data_source.
entity_model.data_source*

gooddata_sdk.catalog.data_source.entity_model.content_objects

Modules

*gooddata_sdk.catalog.data_source.
entity_model.content_objects.table*

gooddata_sdk.catalog.data_source.entity_model.content_objects.table

Classes

CatalogDataSourceTable(*, id, type, attributes)

CatalogDataSourceTableAttributes(*, columns)

CatalogDataSourceTableColumn(*, name,
data_type)

gooddata_sdk.catalog.data_source.entity_model.content_objects.table.CatalogDataSourceTable


```
class gooddata_sdk.catalog.data_source.entity_model.content_objects.table.CatalogDataSourceTable(*,
id:
str,
type:
str,
attributes:
CatalogDataSourceTableAttributes)
```

Bases: *Base*

__init__(*, id: str, type: str, attributes: CatalogDataSourceTableAttributes) → None

Method generated by attrs for class CatalogDataSourceTable.

Methods

__init__ (*, id, type, attributes)	Method generated by attrs for class CatalogDataSourceTable.
client_class ()	
from_api (entity)	Creates object from entity passed by client class, which represents it as dictionary.
from_dict (data[, camel_case])	Creates object from dictionary.
to_api ()	
to_dict ([camel_case])	Converts object into dictionary.

Attributes

id
type
attributes

classmethod from_api(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod from_dict(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.data_source.entity_model.content_objects.table.CatalogDataSourceTableAttributes

```
class gooddata_sdk.catalog.data_source.entity_model.content_objects.table.CatalogDataSourceTableAttribu
```

Bases: *Base*

```
__init__(* , columns: List[CatalogDataSourceTableColumn], name_prefix: Optional[str] = None, path: Optional[List[str]] = None, type: Optional[str] = None) → None
```

Method generated by attrs for class CatalogDataSourceTableAttributes.

Methods

<code><i>__init__</i>(* , columns[, name_prefix, path, type])</code>	Method generated by attrs for class CatalogDataSourceTableAttributes.
<code>client_class()</code>	
<code><i>from_api</i>(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code><i>from_dict</i>(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code><i>to_dict</i>([camel_case])</code>	Converts object into dictionary.

Attributes

<code>columns</code>
<code>name_prefix</code>
<code>path</code>
<code>type</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.data_source.entity_model.content_objects.table.CatalogDataSourceTableColumn`

class `gooddata_sdk.catalog.data_source.entity_model.content_objects.table.CatalogDataSourceTableColumn`

Bases: `Base`

__init__(**, name: str, data_type: str, is_primary_key: Optional[bool] = None, referenced_table_column: Optional[str] = None, referenced_table_id: Optional[str] = None*) → None

Method generated by attrs for class `CatalogDataSourceTableColumn`.

Methods

<code>__init__(*, name, data_type[, ...])</code>	Method generated by attrs for class <code>CatalogData-SourceTableColumn</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>name</code>
<code>data_type</code>
<code>is_primary_key</code>
<code>referenced_table_column</code>
<code>referenced_table_id</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.data_source.entity_model.data_source**Classes**

BigQueryAttributes(project_id[, port])

CatalogDataSource(id, name, schema, credentials)

CatalogDataSourceBigQuery(id, name, schema, ...)

CatalogDataSourcePostgres(id, name, schema, ...)

CatalogDataSourceRedshift(id, name, schema, ...)

CatalogDataSourceSnowflake(id, name, schema,
...)

CatalogDataSourceVertica(id, name, schema, ...)

DatabaseAttributes()

PostgresAttributes(host, db_name[, port])

RedshiftAttributes(host, db_name[, port])

SnowflakeAttributes(account, warehouse,
db_name)

VerticaAttributes(host, db_name[, port])

gooddata_sdk.catalog.data_source.entity_model.data_source.BigQueryAttributes

```

class gooddata_sdk.catalog.data_source.entity_model.data_source.BigQueryAttributes(project_id:
                                                                                   str,
                                                                                   port: str
                                                                                   =
                                                                                   '443')

```

Bases: *DatabaseAttributes*

```
__init__(project_id: str, port: str = '443')
```

Methods

__init__(project_id[, port])

Attributes

str_attributes

gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSource

```
class gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSource(id: str,
                                                                                    name:
                                                                                    str,
                                                                                    schema:
                                                                                    str, credentials:
                                                                                    Credentials, url:
                                                                                    Optional[str]
                                                                                    = None,
                                                                                    data_source_type:
                                                                                    Optional[str]
                                                                                    = None,
                                                                                    db_specific_attributes:
                                                                                    Optional[DatabaseAttributes]
                                                                                    = None,
                                                                                    enable_caching:
                                                                                    Optional[bool]
                                                                                    = None,
                                                                                    cache_path:
                                                                                    Optional[list[str]]
                                                                                    = None,
                                                                                    url_params:
                                                                                    Optional[List[Tuple[str,
                                                                                    str]]] =
                                                                                    None)
```

Bases: [CatalogNameEntity](#)

```
__init__(id: str, name: str, schema: str, credentials: Credentials, url: Optional[str] = None,
          data_source_type: Optional[str] = None, db_specific_attributes: Optional[DatabaseAttributes] =
          None, enable_caching: Optional[bool] = None, cache_path: Optional[list[str]] = None,
          url_params: Optional[List[Tuple[str, str]]] = None)
```

Methods

`__init__(id, name, schema, credentials[, ...])`

`from_api(entity)`

`to_api()`

`to_api_patch(data_source_id, attributes)`

`gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSourceBigQuery`

```
class gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSourceBigQuery(id:
    str,
    name:
    str,
    schema:
    str,
    cre-
    den-
    tials:
    Credentials,
    url:
    Optional[str]
    =
    None,
    data_source_type:
    Optional[str]
    =
    None,
    db_specific_attrib
    Optional[DatabaseA
    =
    None,
    enable_caching:
    Optional[bool]
    =
    None,
    cache_path:
    Optional[list[str]]
    =
    None,
    url_params:
    Optional[List[Tuple
    str]])
    =
    None)
```

Bases: [CatalogDataSource](#)

```
__init__(id: str, name: str, schema: str, credentials: Credentials, url: Optional[str] = None,
    data_source_type: Optional[str] = None, db_specific_attributes: Optional[DatabaseAttributes] =
    None, enable_caching: Optional[bool] = None, cache_path: Optional[list[str]] = None,
    url_params: Optional[List[Tuple[str, str]]] = None)
```


Methods

`__init__(id, name, schema, credentials[, ...])`

`from_api(entity)`

`to_api()`

`to_api_patch(data_source_id, attributes)`

`gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSourcePostgres`

```
class gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSourcePostgres(id:
    str,
    name:
    str,
    schema:
    str,
    cre-
    den-
    tials:
    Credentials,
    url:
    Optional[str]
    =
    None,
    data_source_type:
    Optional[str]
    =
    None,
    db_specific_attrib
    Optional[DatabaseA
    =
    None,
    enable_caching:
    Optional[bool]
    =
    None,
    cache_path:
    Optional[list[str]]
    =
    None,
    url_params:
    Optional[List[Tuple
    str]]]
    =
    None)
```

Bases: [CatalogDataSource](#)

```
__init__(id: str, name: str, schema: str, credentials: Credentials, url: Optional[str] = None,
    data_source_type: Optional[str] = None, db_specific_attributes: Optional[DatabaseAttributes] =
    None, enable_caching: Optional[bool] = None, cache_path: Optional[list[str]] = None,
    url_params: Optional[List[Tuple[str, str]]] = None)
```

Methods

`__init__(id, name, schema, credentials[, ...])`

`from_api(entity)`

`to_api()`

`to_api_patch(data_source_id, attributes)`

`gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSourceRedshift`

```
class gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSourceRedshift(id:
                                                                                         str,
                                                                                         name:
                                                                                         str,
                                                                                         schema:
                                                                                         str,
                                                                                         cre-
                                                                                         den-
                                                                                         tials:
                                                                                         Credentials,
                                                                                         url:
                                                                                         Optional[str]
                                                                                         =
                                                                                         None,
                                                                                         data_source_type:
                                                                                         Optional[str]
                                                                                         =
                                                                                         None,
                                                                                         db_specific_attrib
                                                                                         Optional[DatabaseAttributes]
                                                                                         =
                                                                                         None,
                                                                                         en-
                                                                                         able_caching:
                                                                                         Optional[bool]
                                                                                         =
                                                                                         None,
                                                                                         cache_path:
                                                                                         Optional[list[str]]
                                                                                         =
                                                                                         None,
                                                                                         url_params:
                                                                                         Optional[List[Tuple[str, str]]]
                                                                                         =
                                                                                         None)
```

Bases: [CatalogDataSourcePostgres](#)

```
__init__(id: str, name: str, schema: str, credentials: Credentials, url: Optional[str] = None,
          data_source_type: Optional[str] = None, db_specific_attributes: Optional[DatabaseAttributes] =
          None, enable_caching: Optional[bool] = None, cache_path: Optional[list[str]] = None,
          url_params: Optional[List[Tuple[str, str]]] = None)
```

Methods

`__init__(id, name, schema, credentials[, ...])`

`from_api(entity)`

`to_api()`

`to_api_patch(data_source_id, attributes)`

`gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSourceSnowflake`

```

class gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSourceSnowflake(id:
    str,
    name:
    str,
    schema:
    str,
    cre-
    den-
    tials:
    Cre-
    den-
    tials,
    url:
    Op-
    tional[str]
    =
    None,
    data_source_type:
    Op-
    tional[str]
    =
    None,
    db_specific_attr
    Op-
    tional[Database
    =
    None,
    en-
    able_caching:
    Op-
    tional[bool]
    =
    None,
    cache_path:
    Op-
    tional[list[str]]
    =
    None,
    url_params:
    Op-
    tional[List[Tuple
    str]])
    =
    None)

```

Bases: `CatalogDataSource`

```

__init__(id: str, name: str, schema: str, credentials: Credentials, url: Optional[str] = None,
    data_source_type: Optional[str] = None, db_specific_attributes: Optional[DatabaseAttributes] =
    None, enable_caching: Optional[bool] = None, cache_path: Optional[list[str]] = None,
    url_params: Optional[List[Tuple[str, str]]] = None)

```

Methods

`__init__(id, name, schema, credentials[, ...])`

`from_api(entity)`

`to_api()`

`to_api_patch(data_source_id, attributes)`

`gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSourceVertica`

```
class gooddata_sdk.catalog.data_source.entity_model.data_source.CatalogDataSourceVertica(id:
                                                                    str,
                                                                    name:
                                                                    str,
                                                                    schema:
                                                                    str,
                                                                    cre-
                                                                    den-
                                                                    tials:
                                                                    Credentials,
                                                                    url:
                                                                    Optional[str]
                                                                    =
                                                                    None,
                                                                    data_source_type:
                                                                    Optional[str]
                                                                    =
                                                                    None,
                                                                    db_specific_attributes:
                                                                    Optional[DatabaseAttributes]
                                                                    =
                                                                    None,
                                                                    enable_caching:
                                                                    Optional[bool]
                                                                    =
                                                                    None,
                                                                    cache_path:
                                                                    Optional[list[str]]
                                                                    =
                                                                    None,
                                                                    url_params:
                                                                    Optional[List[Tuple[str, str]]]
                                                                    =
                                                                    None)
```

Bases: [CatalogDataSourcePostgres](#)

```
__init__(id: str, name: str, schema: str, credentials: Credentials, url: Optional[str] = None,
          data_source_type: Optional[str] = None, db_specific_attributes: Optional[DatabaseAttributes] =
          None, enable_caching: Optional[bool] = None, cache_path: Optional[list[str]] = None,
          url_params: Optional[List[Tuple[str, str]]] = None)
```


Methods

`__init__(id, name, schema, credentials[, ...])`

`from_api(entity)`

`to_api()`

`to_api_patch(data_source_id, attributes)`

`gooddata_sdk.catalog.data_source.entity_model.data_source.DatabaseAttributes`

```
class gooddata_sdk.catalog.data_source.entity_model.data_source.DatabaseAttributes
```

```
    Bases: object
```

```
    __init__()
```

Methods

`__init__()`

Attributes

`str_attributes`

`gooddata_sdk.catalog.data_source.entity_model.data_source.PostgresAttributes`

```
class gooddata_sdk.catalog.data_source.entity_model.data_source.PostgresAttributes(host:
                                                                                       str,
                                                                                       db_name:
                                                                                       str,
                                                                                       port: str
                                                                                       =
                                                                                       '5432')
```

```
    Bases: DatabaseAttributes
```

```
    __init__(host: str, db_name: str, port: str = '5432')
```

Methods

```
__init__(host, db_name[, port])
```

Attributes

```
str_attributes
```

gooddata_sdk.catalog.data_source.entity_model.data_source.RedshiftAttributes

```
class gooddata_sdk.catalog.data_source.entity_model.data_source.RedshiftAttributes(host:
                                                                                    str,
                                                                                    db_name:
                                                                                    str,
                                                                                    port: str
                                                                                    =
                                                                                    '5439')
```

Bases: *PostgresAttributes*

```
__init__(host: str, db_name: str, port: str = '5439')
```

Methods

```
__init__(host, db_name[, port])
```

Attributes

```
str_attributes
```

gooddata_sdk.catalog.data_source.entity_model.data_source.SnowflakeAttributes

```
class gooddata_sdk.catalog.data_source.entity_model.data_source.SnowflakeAttributes(account:
                                                                                    str,
                                                                                    ware-
                                                                                    house:
                                                                                    str,
                                                                                    db_name:
                                                                                    str,
                                                                                    port:
                                                                                    str =
                                                                                    '443')
```

Bases: *DatabaseAttributes*

```
__init__(account: str, warehouse: str, db_name: str, port: str = '443')
```

Methods

```
__init__(account, warehouse, db_name[, port])
```

Attributes

```
str_attributes
```

gooddata_sdk.catalog.data_source.entity_model.data_source.VerticaAttributes

```
class gooddata_sdk.catalog.data_source.entity_model.data_source.VerticaAttributes(host: str,  
                                                                                    db_name:  
                                                                                    str, port:  
                                                                                    str =  
                                                                                    '5433')
```

Bases: *PostgresAttributes*

```
__init__(host: str, db_name: str, port: str = '5433')
```

Methods

```
__init__(host, db_name[, port])
```

Attributes

```
str_attributes
```

gooddata_sdk.catalog.data_source.service

Classes

```
CatalogDataSourceService(api_client)
```

gooddata_sdk.catalog.data_source.service.CatalogDataSourceService

```
class gooddata_sdk.catalog.data_source.service.CatalogDataSourceService(api_client:  
                                                                           GoodDataApiClient)
```

Bases: *CatalogServiceBase*

__init__(*api_client:* GoodDataApiClient) → None

Methods

`__init__(api_client)`

`create_or_update_data_source(data_source)`

`data_source_folder(data_source_id, ...)`

`delete_data_source(data_source_id)`

`generate_logical_model(data_source_id[, ...])`

`get_data_source(data_source_id)`

`get_declarative_data_sources()`

`get_declarative_pdm(data_source_id)`

`get_organization()`

`layout_organization_folder(layout_root_path)`

`list_data_source_tables(data_source_id)`

`list_data_sources()`

`load_and_put_declarative_data_sources([...])`

`load_and_put_declarative_pdm(data_source_id)`

`load_declarative_data_sources([layout_root_path])`

`load_declarative_pdm(data_source_id[, ...])`

`patch_data_source_attributes(data_source_id,
...)`

`put_declarative_data_sources(...[, ...])`

`put_declarative_pdm(data_source_id, ...)`

`register_upload_notification(data_source_id)`

`report_warnings(warnings)`

`scan_and_put_pdm(data_source_id[,
scan_request])`

`scan_data_source(data_source_id[, ...])`

`scan_schemata(data_source_id)`

`store_declarative_data_sources([...])`

`store_declarative_pdm(data_source_id[, ...])`

Attributes

`organization_id`

`gooddata_sdk.catalog.data_source.validation`

Modules

`gooddata_sdk.catalog.data_source.
validation.data_source`

`gooddata_sdk.catalog.data_source.validation.data_source`

Classes

`DataSourceValidator(data_source_service)`

`gooddata_sdk.catalog.data_source.validation.data_source.DataSourceValidator`

`class gooddata_sdk.catalog.data_source.validation.data_source.DataSourceValidator`*(data_source_service: Catalog-Data-Source-Service)*

Bases: `object`

`__init__(data_source_service: CatalogDataSourceService)`

Methods

`__init__(data_source_service)`

`validate_data_source_ids(data_source_ids)`

`validate_ldm(model)`

gooddata_sdk.catalog.entity**Classes**

BasicCredentials(username, password)

CatalogEntity(entity)

CatalogNameEntity(id, name)

CatalogTitleEntity(id, title)

CatalogTypeEntity(id, type)

Credentials()

TokenCredentials(token)

TokenCredentialsFromFile(file_path)

gooddata_sdk.catalog.entity.BasicCredentials**class** gooddata_sdk.catalog.entity.**BasicCredentials**(username: str, password: str)Bases: *Credentials***__init__**(username: str, password: str)**Methods**

__init__(username, password)

create(creds_classes, entity)

from_api(attributes)

is_part_of_api(entity)

to_api_args()

validate_instance(creds_classes, instance)

Attributes

PASSWORD_KEY

USER_KEY

gooddata_sdk.catalog.entity.CatalogEntity

class gooddata_sdk.catalog.entity.CatalogEntity(entity: dict[str, Any])

Bases: object

__init__(entity: dict[str, Any]) → None

Methods

__init__(entity)

Attributes

description

id

obj_id

title

type

gooddata_sdk.catalog.entity.CatalogNameEntity

class gooddata_sdk.catalog.entity.CatalogNameEntity(id: str, name: str)

Bases: object

__init__(id: str, name: str)

Methods

```
__init__(id, name)
```

`gooddata_sdk.catalog.entity.CatalogTitleEntity`

```
class gooddata_sdk.catalog.entity.CatalogTitleEntity(id: str, title: str)
```

```
Bases: object
```

```
__init__(id: str, title: str)
```

Methods

```
__init__(id, title)
```

```
from_api(entity)
```

`gooddata_sdk.catalog.entity.CatalogTypeEntity`

```
class gooddata_sdk.catalog.entity.CatalogTypeEntity(id: str, type: str)
```

```
Bases: object
```

```
__init__(id: str, type: str)
```

Methods

```
__init__(id, type)
```

```
from_api(entity)
```

`gooddata_sdk.catalog.entity.Credentials`

```
class gooddata_sdk.catalog.entity.Credentials
```

```
Bases: object
```

```
__init__()
```

Methods

`__init__()`

`create(creds_classes, entity)`

`from_api(entity)`

`is_part_of_api(entity)`

`to_api_args()`

`validate_instance(creds_classes, instance)`

`gooddata_sdk.catalog.entity.TokenCredentials`

class `gooddata_sdk.catalog.entity.TokenCredentials`(*token: str*)

Bases: `Credentials`

`__init__`(*token: str*)

Methods

`__init__(token)`

`create(creds_classes, entity)`

`from_api(entity)`

`is_part_of_api(entity)`

`to_api_args()`

`validate_instance(creds_classes, instance)`

Attributes

`TOKEN_KEY`

`USER_KEY`

gooddata_sdk.catalog.entity.TokenCredentialsFromFile

```
class gooddata_sdk.catalog.entity.TokenCredentialsFromFile(file_path: Path)
```

Bases: *Credentials*

```
__init__(file_path: Path)
```

Methods

```
__init__(file_path)
```

```
create(creds_classes, entity)
```

```
from_api(entity)
```

```
is_part_of_api(entity)
```

```
to_api_args()
```

```
token_from_file(file_path)
```

```
validate_instance(creds_classes, instance)
```

Attributes

```
TOKEN_KEY
```

```
USER_KEY
```

gooddata_sdk.catalog.identifier**Classes**

```
CatalogAssigneeIdentifier(* , id, type)
```

```
CatalogGrainIdentifier(* , id, type)
```

```
CatalogLabelIdentifier(* , id, type)
```

```
CatalogReferenceIdentifier(* , id)
```

```
CatalogUserGroupIdentifier(* , id, type)
```

```
CatalogWorkspaceIdentifier(* , id)
```

gooddata_sdk.catalog.identifier.CatalogAssigneeIdentifier**class** gooddata_sdk.catalog.identifier.CatalogAssigneeIdentifier(*, id: str, type: str)Bases: *Base***__init__**(*, id: str, type: str) → None

Method generated by attrs for class CatalogAssigneeIdentifier.

Methods

<code>__init__</code> (*, id, type)	Method generated by attrs for class CatalogAssigneeIdentifier.
<code>client_class</code> ()	
<code>from_api</code> (entity)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (data[, camel_case])	Creates object from dictionary.
<code>to_api</code> ()	
<code>to_dict</code> ([camel_case])	Converts object into dictionary.

Attributes

<code>id</code>
<code>type</code>

classmethod `from_api`(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict`(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.identifier.CatalogGrainIdentifier**class** gooddata_sdk.catalog.identifier.CatalogGrainIdentifier(*, id: str, type: str)Bases: *Base***__init__**(*, id: str, type: str) → None

Method generated by attrs for class CatalogGrainIdentifier.

Methods

<code>__init__(*, id, type)</code>	Method generated by attrs for class Catalog-GrainIdentifier.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>type</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.identifier.CatalogLabelIdentifier`

class `gooddata_sdk.catalog.identifier.CatalogLabelIdentifier(*, id: str, type: str)`

Bases: `Base`

`__init__(*, id: str, type: str) → None`

Method generated by attrs for class CatalogLabelIdentifier.

Methods

<code>__init__(*, id, type)</code>	Method generated by attrs for class <code>CatalogLabelIdentifier</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>type</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.identifier.CatalogReferenceIdentifier`

class `gooddata_sdk.catalog.identifier.CatalogReferenceIdentifier(*, id: str)`

Bases: `Base`

`__init__(*, id: str) → None`

Method generated by attrs for class `CatalogReferenceIdentifier`.

Methods

<code>__init__(*, id)</code>	Method generated by attrs for class CatalogReferenceIdentifier.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.identifier.CatalogUserGroupIdentifier`

class `gooddata_sdk.catalog.identifier.CatalogUserGroupIdentifier(*, id: str, type: str)`

Bases: `Base`

`__init__(*, id: str, type: str) → None`

Method generated by attrs for class CatalogUserGroupIdentifier.

Methods

<code>__init__(*, id, type)</code>	Method generated by attrs for class CatalogUserGroupIdentifier.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

id

type

classmethod **from_api**(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod **from_dict**(*data: Dict[str, Any], camel_case: bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.identifier.CatalogWorkspaceIdentifier

class gooddata_sdk.catalog.identifier.CatalogWorkspaceIdentifier(*, *id: str*)

Bases: *Base*

__init__(*, *id: str*) → None

Method generated by attrs for class CatalogWorkspaceIdentifier.

Methods

__init__(*, *id*)

Method generated by attrs for class CatalogWorkspaceIdentifier.

client_class()

from_api(*entity*)

Creates object from entity passed by client class, which represents it as dictionary.

from_dict(*data[, camel_case]*)

Creates object from dictionary.

to_api()

to_dict(*[camel_case]*)

Converts object into dictionary.

Attributes

id

classmethod **from_api**(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod **from_dict**(*data: Dict[str, Any], camel_case: bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.organization

Modules

*gooddata_sdk.catalog.organization.
entity_model*

gooddata_sdk.catalog.organization.service

gooddata_sdk.catalog.organization.entity_model

Modules

*gooddata_sdk.catalog.organization.
entity_model.organization*

gooddata_sdk.catalog.organization.entity_model.organization

Classes

CatalogOrganization(*, id, attributes)

CatalogOrganizationAttributes(*[, name, ...])

CatalogOrganizationDocument(*, data)

gooddata_sdk.catalog.organization.entity_model.organization.CatalogOrganization

```
class gooddata_sdk.catalog.organization.entity_model.organization.CatalogOrganization(*,
                                                                                       id:
                                                                                       str,
                                                                                       at-
                                                                                       tributes:
                                                                                       Cat-
                                                                                       alo-
                                                                                       gOr-
                                                                                       ga-
                                                                                       ni-
                                                                                       za-
                                                                                       tion-
                                                                                       At-
                                                                                       tributes)
```

Bases: *Base*

__init__(**id: str, attributes: CatalogOrganizationAttributes*) → None

Method generated by attrs for class CatalogOrganization.

Methods

<code>__init__(*, id, attributes)</code>	Method generated by attrs for class CatalogOrganization.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>attributes</code>

classmethod from_api(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod from_dict(*data: Dict[str, Any], camel_case: bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.organization.entity_model.organization.CatalogOrganizationAttributes

```

class gooddata_sdk.catalog.organization.entity_model.organization.CatalogOrganizationAttributes(*,
                                                                                               name:
                                                                                               Op-
                                                                                               tional[str]
                                                                                               =
                                                                                               None,
                                                                                               host-
                                                                                               name:
                                                                                               Op-
                                                                                               tional[str]
                                                                                               =
                                                                                               None,
                                                                                               al-
                                                                                               lowed_or-
                                                                                               gins:
                                                                                               Op-
                                                                                               tional[List[str]]
                                                                                               =
                                                                                               None,
                                                                                               oauth_iss-
                                                                                               uer_location:
                                                                                               Op-
                                                                                               tional[str]
                                                                                               =
                                                                                               None,
                                                                                               oauth_cli-
                                                                                               ent_id:
                                                                                               Op-
                                                                                               tional[str]
                                                                                               =
                                                                                               None)

```

Bases: *Base*

```

__init__(*, name: Optional[str] = None, hostname: Optional[str] = None, allowed_origins:
Optional[List[str]] = None, oauth_issuer_location: Optional[str] = None, oauth_client_id:
Optional[str] = None) → None

```

Method generated by attrs for class CatalogOrganizationAttributes.

Methods

<code>__init__(*[, name, hostname, ...])</code>	Method generated by attrs for class CatalogOrganizationAttributes.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

`name`

`hostname`

`allowed_origins`

`oauth_issuer_location`

`oauth_client_id`

classmethod `from_api`(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict`(*data: Dict[str, Any]*, *camel_case: bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.organization.entity_model.organization.CatalogOrganizationDocument`

```
class gooddata_sdk.catalog.organization.entity_model.organization.CatalogOrganizationDocument(*,
                                                                                               data:
                                                                                               Cat-
                                                                                               a-
                                                                                               l-
                                                                                               o-
                                                                                               gOr-
                                                                                               ga-
                                                                                               ni-
                                                                                               za-
                                                                                               tion)
```

Bases: `Base`

__init__(*, *data: CatalogOrganization*) → None

Method generated by attrs for class CatalogOrganizationDocument.

Methods

<code>__init__(*, data)</code>	Method generated by attrs for class <code>CatalogOrganizationDocument</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api([oauth_client_secret])</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>data</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.organization.service

Classes

<code>CatalogOrganizationService(api_client)</code>

gooddata_sdk.catalog.organization.service.CatalogOrganizationService

class `gooddata_sdk.catalog.organization.service.CatalogOrganizationService(api_client: GoodDataApiClient)`

Bases: `CatalogServiceBase`

__init__(`api_client: GoodDataApiClient`) → None

Methods

`__init__(api_client)`

`get_organization()`

`layout_organization_folder(layout_root_path)`

`update_name(name)`

`update_oidc_parameters([...])`

Attributes

`organization_id`

`gooddata_sdk.catalog.permission`

Modules

`gooddata_sdk.catalog.permission.`

`declarative_model`

`gooddata_sdk.catalog.permission.service`

`gooddata_sdk.catalog.permission.declarative_model`

Modules

`gooddata_sdk.catalog.permission.`

`declarative_model.permission`

`gooddata_sdk.catalog.permission.declarative_model.permission`

Classes

`CatalogDeclarativeDataSourcePermission(*,`
`...)`

`CatalogDeclarativeSingleWorkspacePermission(*,`
`...)`

`CatalogDeclarativeWorkspaceHierarchyPermission(*,`
`...)`

`CatalogDeclarativeWorkspacePermissions(*[,`
`...])`

gooddata_sdk.catalog.permission.declarative_model.permission.CatalogDeclarativeDataSourcePermission

```
class gooddata_sdk.catalog.permission.declarative_model.permission.CatalogDeclarativeDataSourcePermission
```

Bases: *Base*

`__init__`(*, *name*: str, *assignee*: CatalogAssigneeIdentifier) → None

Method generated by attrs for class CatalogDeclarativeDataSourcePermission.

Methods

<code>__init__</code> (*, <i>name</i> , <i>assignee</i>)	Method generated by attrs for class CatalogDeclarativeDataSourcePermission.
<code>client_class</code> ()	
<code>from_api</code> (<i>entity</i>)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (<i>data</i> [, <i>camel_case</i>])	Creates object from dictionary.
<code>to_api</code> ()	
<code>to_dict</code> ([, <i>camel_case</i>])	Converts object into dictionary.

Attributes

<code>name</code>
<code>assignee</code>

classmethod `from_api`(*entity*: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict`(*data*: Dict[str, Any], *camel_case*: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

`to_dict`(*camel_case*: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.permission.declarative_model.permission.CatalogDeclarativeSingleWorkspacePermission``class gooddata_sdk.catalog.permission.declarative_model.permission.CatalogDeclarativeSingleWorkspacePerm`

Bases: *Base*

`__init__`(*, *name*: str, *assignee*: *CatalogAssigneeIdentifier*) → None

Method generated by attrs for class *CatalogDeclarativeSingleWorkspacePermission*.

Methods

<code>__init__</code> (*, <i>name</i> , <i>assignee</i>)	Method generated by attrs for class <i>CatalogDeclarativeSingleWorkspacePermission</i> .
<code>client_class</code> ()	
<code>from_api</code> (<i>entity</i>)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (<i>data</i> [, <i>camel_case</i>])	Creates object from dictionary.
<code>to_api</code> ()	
<code>to_dict</code> ([, <i>camel_case</i>])	Converts object into dictionary.

Attributes

<code>name</code>
<code>assignee</code>

classmethod `from_api`(*entity*: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict`(*data*: Dict[str, Any], *camel_case*: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case*: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.permission.declarative_model.permission.CatalogDeclarativeWorkspaceHierarchyPermission`

`class gooddata_sdk.catalog.permission.declarative_model.permission.CatalogDeclarativeWorkspaceHierarchyPermission`

Bases: *Base*

`__init__(*, name: str, assignee: CatalogAssigneeIdentifier) → None`

Method generated by attrs for class `CatalogDeclarativeWorkspaceHierarchyPermission`.

Methods

<code>__init__(*, name, assignee)</code>	Method generated by attrs for class <code>CatalogDeclarativeWorkspaceHierarchyPermission</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>name</code>
<code>assignee</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.permission.declarative_model.permission.CatalogDeclarativeWorkspacePermissions**class** gooddata_sdk.catalog.permission.declarative_model.permission.CatalogDeclarativeWorkspacePermissionsBases: *Base*

```
__init__(*[, permissions: List[CatalogDeclarativeSingleWorkspacePermission] = NOTHING,  
          hierarchy_permissions: List[CatalogDeclarativeWorkspaceHierarchyPermission] = NOTHING)  
    → None
```

Method generated by attrs for class CatalogDeclarativeWorkspacePermissions.

Methods

<code><i>__init__</i>(*[, permissions, hierarchy_permissions])</code>	Method generated by attrs for class CatalogDeclarativeWorkspacePermissions.
<code>client_class()</code>	
<code><i>from_api</i>(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code><i>from_dict</i>(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code><i>to_dict</i>([camel_case])</code>	Converts object into dictionary.

Attributes

<code>permissions</code>
<code>hierarchy_permissions</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.permission.service

Classes

CatalogPermissionService(api_client)

gooddata_sdk.catalog.permission.service.CatalogPermissionService

class gooddata_sdk.catalog.permission.service.CatalogPermissionService(*api_client:*
GoodDataApiClient)

Bases: *CatalogServiceBase*

__init__(*api_client: GoodDataApiClient*) → None

Methods

__init__(api_client)

get_declarative_permissions(workspace_id)

get_organization()

layout_organization_folder(layout_root_path)

put_declarative_permissions(workspace_id,
...)

Attributes

organization_id

gooddata_sdk.catalog.setting**Classes**

CatalogDeclarativeSetting(*, id[, content])

gooddata_sdk.catalog.setting.CatalogDeclarativeSetting**class** gooddata_sdk.catalog.setting.CatalogDeclarativeSetting(*, id: str, content: Optional[Dict[str, Any]] = None)Bases: *Base***__init__**(*, id: str, content: Optional[Dict[str, Any]] = None) → None

Method generated by attrs for class CatalogDeclarativeSetting.

Methods

<i>__init__</i> (*, id[, content])	Method generated by attrs for class CatalogDeclarativeSetting.
<i>client_class</i> ()	
<i>from_api</i> (entity)	Creates object from entity passed by client class, which represents it as dictionary.
<i>from_dict</i> (data[, camel_case])	Creates object from dictionary.
<i>to_api</i> ()	
<i>to_dict</i> ([camel_case])	Converts object into dictionary.

Attributes

<i>id</i>
<i>content</i>

classmethod *from_api*(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod *from_dict*(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.types**gooddata_sdk.catalog.user****Modules**

gooddata_sdk.catalog.user.declarative_model

gooddata_sdk.catalog.user.entity_model

gooddata_sdk.catalog.user.service

gooddata_sdk.catalog.user.declarative_model**Modules**

gooddata_sdk.catalog.user.declarative_model.user

gooddata_sdk.catalog.user.declarative_model.user_and_user_groups

gooddata_sdk.catalog.user.declarative_model.user_group

gooddata_sdk.catalog.user.declarative_model.user**Classes**

CatalogDeclarativeUser(, id[, auth_id, ...])*

CatalogDeclarativeUsers(, users)*

gooddata_sdk.catalog.user.declarative_model.user.CatalogDeclarativeUser

```
class gooddata_sdk.catalog.user.declarative_model.user.CatalogDeclarativeUser(*, id: str,
                                                                              auth_id:
                                                                              Optional[str]
                                                                              = None,
                                                                              user_groups:
                                                                              List[CatalogUserGroupIdentifier]
                                                                              = NOTHING,
                                                                              settings:
                                                                              List[CatalogDeclarativeSetting]
                                                                              = NOTHING)
```

Bases: *Base*

```
__init__(* , id: str, auth_id: Optional[str] = None, user_groups: List[CatalogUserGroupIdentifier] =  
        NOTHING, settings: List[CatalogDeclarativeSetting] = NOTHING) → None
```

Method generated by attrs for class CatalogDeclarativeUser.

Methods

<code>__init__(* , id[, auth_id, user_groups, settings])</code>	Method generated by attrs for class CatalogDeclarativeUser.
<code>client_class()</code>	
<code><i>from_api</i>(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code><i>from_dict</i>(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code><i>to_dict</i>([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>auth_id</code>
<code>user_groups</code>
<code>settings</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.user.declarative_model.user.CatalogDeclarativeUsers`

```
class gooddata_sdk.catalog.user.declarative_model.user.CatalogDeclarativeUsers(* , users:  
                                                                           List[CatalogDeclarativeUser])
```

Bases: `Base`

```
__init__(* , users: List[CatalogDeclarativeUser]) → None
```

Method generated by attrs for class CatalogDeclarativeUsers.

Methods

<code>__init__(*, users)</code>	Method generated by attrs for class <code>CatalogDeclarativeUsers</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(layout_organization_folder)</code>	
<code>store_to_disk(layout_organization_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>users</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.user.declarative_model.user_and_user_groups`

Classes

<code>CatalogDeclarativeUsersUserGroups(*, users, ...)</code>

`gooddata_sdk.catalog.user.declarative_model.user_and_user_groups.CatalogDeclarativeUsersUserGroups`

class `gooddata_sdk.catalog.user.declarative_model.user_and_user_groups.CatalogDeclarativeUsersUserGroups`

Bases: `Base`

```
__init__(*, users: List[CatalogDeclarativeUser], user_groups: List[CatalogDeclarativeUserGroup]) → None
```

Method generated by attrs for class CatalogDeclarativeUsersUserGroups.

Methods

<code>__init__(*, users, user_groups)</code>	Method generated by attrs for class CatalogDeclarativeUsersUserGroups.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(layout_organization_folder)</code>	
<code>store_to_disk(layout_organization_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>users</code>
<code>user_groups</code>

classmethod from_api(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod from_dict(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.user.declarative_model.user_group

Classes

<code>CatalogDeclarativeUserGroup(*, id[, parents])</code>
<code>CatalogDeclarativeUserGroups(*[, user_groups])</code>

gooddata_sdk.catalog.user.declarative_model.user_group.CatalogDeclarativeUserGroup

```
class gooddata_sdk.catalog.user.declarative_model.user_group.CatalogDeclarativeUserGroup(*,
                                                                                          id:
                                                                                          str,
                                                                                          par-
                                                                                          ents:
                                                                                          Op-
                                                                                          tional[List[Catalog
                                                                                          =
                                                                                          None])
```

Bases: *Base*

__init__(*, id: str, parents: Optional[List[CatalogUserGroupIdentifier]] = None) → None

Method generated by attrs for class CatalogDeclarativeUserGroup.

Methods

__init__ (*, id[, parents])	Method generated by attrs for class CatalogDeclarativeUserGroup.
client_class ()	
from_api (entity)	Creates object from entity passed by client class, which represents it as dictionary.
from_dict (data[, camel_case])	Creates object from dictionary.
to_api ()	
to_dict ([camel_case])	Converts object into dictionary.

Attributes

id
parents

classmethod from_api(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod from_dict(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.user.declarative_model.user_group.CatalogDeclarativeUserGroups

```
class gooddata_sdk.catalog.user.declarative_model.user_group.CatalogDeclarativeUserGroups(*,
                                                                 user_groups: List[CatalogDeclarativeUserGroup] = NOTHING) → None
```

Bases: *Base*

__init__(*[, user_groups: List[CatalogDeclarativeUserGroup] = NOTHING) → None

Method generated by attrs for class CatalogDeclarativeUserGroups.

Methods

<code>__init__(*[, user_groups])</code>	Method generated by attrs for class CatalogDeclarativeUserGroups.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(layout_organization_folder)</code>	
<code>store_to_disk(layout_organization_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>user_groups</code>

classmethod from_api(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod from_dict(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.user.entity_model**Modules**

```
gooddata_sdk.catalog.user.entity_model.  
user  
gooddata_sdk.catalog.user.entity_model.  
user_group
```

gooddata_sdk.catalog.user.entity_model.user**Classes**

```
CatalogUser(*, id[, attributes, relationships])  
  
CatalogUserAttributes(*[, authentication_id])  
  
CatalogUserDocument(*, data)  
  
CatalogUserGroupsData(*[, data])  
  
CatalogUserRelationships(*[, user_groups])
```

gooddata_sdk.catalog.user.entity_model.user.CatalogUser

```
class gooddata_sdk.catalog.user.entity_model.user.CatalogUser(*, id: str, attributes:  
                                                                Optional[CatalogUserAttributes] =  
                                                                None, relationships: Op-  
                                                                tional[CatalogUserRelationships]  
                                                                = None)
```

Bases: *Base*

```
__init__(*, id: str, attributes: Optional[CatalogUserAttributes] = None, relationships:  
        Optional[CatalogUserRelationships] = None) → None
```

Method generated by attrs for class CatalogUser.

Methods

<code>__init__(*, id[, attributes, relationships])</code>	Method generated by attrs for class CatalogUser.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>init(user_id[, authentication_id, ...])</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>get_user_groups</code>
<code>id</code>
<code>attributes</code>
<code>relationships</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.user.entity_model.user.CatalogUserAttributes`

class `gooddata_sdk.catalog.user.entity_model.user.CatalogUserAttributes(*, authentication_id: Optional[str] = None)`

Bases: `Base`

__init__(`*, authentication_id: Optional[str] = None`) → None

Method generated by attrs for class CatalogUserAttributes.

Methods

<code>__init__(*[, authentication_id])</code>	Method generated by attrs for class CatalogUserAttributes.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>authentication_id</code>	
--------------------------------	--

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict `(camel_case: bool = True) → Dict[str, Any]`

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.user.entity_model.user.CatalogUserDocument`

class `gooddata_sdk.catalog.user.entity_model.user.CatalogUserDocument(*, data: CatalogUser)`

Bases: `Base`

__init__ `(*, data: CatalogUser) → None`

Method generated by attrs for class CatalogUserDocument.

Methods

<code>__init__(*, data)</code>	Method generated by attrs for class CatalogUserDocument.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>init(user_id[, authentication_id, ...])</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.
<code>update_user([authentication_id, user_group_ids])</code>	

Attributes

<code>data</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.user.entity_model.user.CatalogUserGroupsData`

class `gooddata_sdk.catalog.user.entity_model.user.CatalogUserGroupsData(*, data: Optional[List[CatalogUserGroup]] = None)`

Bases: `Base`

__init__(`*, data: Optional[List[CatalogUserGroup]] = None`) → None

Method generated by attrs for class CatalogUserGroupsData.

Methods

<code>__init__(*[, data])</code>	Method generated by attrs for class <code>CatalogUserGroupsData</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>get_user_groups</code>	
<code>data</code>	

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.user.entity_model.user.CatalogUserRelationships`

```
class gooddata_sdk.catalog.user.entity_model.user.CatalogUserRelationships(*, user_groups:
    Optional[CatalogUserGroupsData] = None)
```

Bases: `Base`

__init__(*[, user_groups: Optional[CatalogUserGroupsData] = None) → None

Method generated by attrs for class `CatalogUserRelationships`.

Methods

<code>__init__(*[, user_groups])</code>	Method generated by attrs for class <code>CatalogUserRelationships</code> .
<code>client_class()</code>	
<code>create_user_relationships(user_group_ids)</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>get_user_groups</code>
<code>user_groups</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.user.entity_model.user_group`

Classes

<code>CatalogUserGroup(*, id[, relationships])</code>
<code>CatalogUserGroupDocument(*, data)</code>
<code>CatalogUserGroupParents(*[, data])</code>
<code>CatalogUserGroupRelationships(*[, parents])</code>

gooddata_sdk.catalog.user.entity_model.user_group.CatalogUserGroup

```
class gooddata_sdk.catalog.user.entity_model.user_group.CatalogUserGroup(*, id: str,
                                                                    relationships: Optional[CatalogUserGroupRelationships],
                                                                    = None)
```

Bases: *Base*

__init__(*, id: str, relationships: Optional[CatalogUserGroupRelationships] = None) → None

Method generated by attrs for class CatalogUserGroup.

Methods

__init__ (*, id[, relationships])	Method generated by attrs for class CatalogUserGroup.
client_class ()	
from_api (entity)	Creates object from entity passed by client class, which represents it as dictionary.
from_dict (data[, camel_case])	Creates object from dictionary.
init (user_group_id[, user_group_parent_ids])	
to_api ()	
to_dict ([camel_case])	Converts object into dictionary.

Attributes

get_parents
id
relationships

classmethod from_api(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod from_dict(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.user.entity_model.user_group.CatalogUserGroupDocument

```
class gooddata_sdk.catalog.user.entity_model.user_group.CatalogUserGroupDocument(*, data: CatalogUserGroup)
```

Bases: *Base*

__init__(*, data: *CatalogUserGroup*) → None

Method generated by attrs for class *CatalogUserGroupDocument*.

Methods

<code>__init__</code> (*, data)	Method generated by attrs for class <i>CatalogUserGroupDocument</i> .
<code>client_class</code> ()	
<code>from_api</code> (entity)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (data[, camel_case])	Creates object from dictionary.
<code>init</code> (user_group_id[, user_group_parent_ids])	
<code>to_api</code> ()	
<code>to_dict</code> ([camel_case])	Converts object into dictionary.
<code>update_user_group</code> ([user_group_parents_id])	

Attributes

<code>data</code>

classmethod `from_api`(entity: *Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict`(data: *Dict[str, Any]*, camel_case: *bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: *bool = True*) → *Dict[str, Any]*

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.user.entity_model.user_group.CatalogUserGroupParents

```
class gooddata_sdk.catalog.user.entity_model.user_group.CatalogUserGroupParents(*, data: Optional[List[CatalogUserGroupParents]] = None)
```

Bases: *Base*

__init__(*, data: Optional[List[CatalogUserGroupParents]] = None) → None

Method generated by attrs for class CatalogUserGroupParents.

Methods

<code>__init__(*[, data])</code>	Method generated by attrs for class CatalogUserGroupParents.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>get_parents</code>
<code>data</code>

classmethod from_api(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod from_dict(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.user.entity_model.user_group.CatalogUserGroupRelationships

```
class gooddata_sdk.catalog.user.entity_model.user_group.CatalogUserGroupRelationships(*,
                                                                                       parents: Optional[CatalogUserGroupRelationships] = None)
```

Bases: *Base*

__init__(*[, parents: *Optional[CatalogUserGroupParents]* = None) → None

Method generated by attrs for class CatalogUserGroupRelationships.

Methods

<code>__init__(*[, parents])</code>	Method generated by attrs for class CatalogUserGroupRelationships.
<code>client_class()</code>	
<code>create_user_group_relationships(...)</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>get_parents</code>
<code>parents</code>

classmethod from_api(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod from_dict(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.user.service

Classes

<code>CatalogUserService(api_client)</code>

gooddata_sdk.catalog.user.service.CatalogUserService

class gooddata_sdk.catalog.user.service.CatalogUserService(*api_client*: GoodDataApiClient)

Bases: *CatalogServiceBase*

__init__(*api_client*: GoodDataApiClient) → None

Methods

`__init__(api_client)`

`create_or_update_user(user)`

`create_or_update_user_group(user_group)`

`delete_user(user_id)`

`delete_user_group(user_group_id)`

`get_declarative_user_groups()`

`get_declarative_users()`

`get_declarative_users_user_groups()`

`get_organization()`

`get_user(user_id)`

`get_user_group(user_group_id)`

`layout_organization_folder(layout_root_path)`

`list_user_groups()`

`list_users()`

`load_and_put_declarative_user_groups([...])`

`load_and_put_declarative_users([...])`

`load_and_put_declarative_users_user_groups([...])`

`load_declarative_user_groups([layout_root_path])`

`load_declarative_users([layout_root_path])`

`load_declarative_users_user_groups([...])`

`put_declarative_user_groups(user_groups)`

`put_declarative_users(users)`

`put_declarative_users_user_groups(...)`

`store_declarative_user_groups([layout_root_path])`

`store_declarative_users([layout_root_path])`

`store_declarative_users_user_groups([...])`

Attributes

`organization_id`

`gooddata_sdk.catalog.workspace`

Modules

`gooddata_sdk.catalog.workspace.content_service`

`gooddata_sdk.catalog.workspace.declarative_model`

`gooddata_sdk.catalog.workspace.entity_model`

`gooddata_sdk.catalog.workspace.model_container`

`gooddata_sdk.catalog.workspace.service`

`gooddata_sdk.catalog.workspace.content_service`

Classes

`CatalogWorkspaceContentService(api_client)`

`gooddata_sdk.catalog.workspace.content_service.CatalogWorkspaceContentService`

```
class gooddata_sdk.catalog.workspace.content_service.CatalogWorkspaceContentService(api_client:
                                                                                       Good-
                                                                                       DataApi-
                                                                                       Client)
```

Bases: `CatalogServiceBase`

`__init__`(*api_client*: `GoodDataApiClient`) → None

Methods

<code>__init__(api_client)</code>	
<code>compute_valid_objects(workspace_id, ctx)</code>	Returns attributes, facts, and metrics which are valid to add to a context that already contains some entities from the semantic model.
<code>get_attributes_catalog(workspace_id)</code>	
<code>get_declarative_analytics_model(workspace_id)</code>	
<code>get_declarative_ldm(workspace_id)</code>	
<code>get_dependent_entities_graph(workspace_id)</code>	
<code>get_dependent_entities_graph_from_entry_points(...)</code>	
<code>get_facts_catalog(workspace_id)</code>	
<code>get_full_catalog(workspace_id)</code>	Retrieves catalog for a workspace.
<code>get_labels_catalog(workspace_id)</code>	
<code>get_metrics_catalog(workspace_id)</code>	
<code>get_organization()</code>	
<code>layout_organization_folder(layout_root_path)</code>	
<code>layout_workspace_folder(workspace_id, ...)</code>	
<code>load_and_put_declarative_analytics_model(...)</code>	
<code>load_and_put_declarative_ldm(workspace_id[, ...])</code>	
<code>load_declarative_analytics_model(workspace_id)</code>	
<code>load_declarative_ldm(workspace_id[, ...])</code>	
<code>put_declarative_analytics_model(...)</code>	
<code>put_declarative_ldm(workspace_id, ldm[, ...])</code>	
<code>store_declarative_analytics_model(workspace_id)</code>	
<code>store_declarative_ldm(workspace_id[, ...])</code>	

Attributes

`organization_id`

compute_valid_objects(*workspace_id*: str, *ctx*: Union[Attribute, Metric, Filter, CatalogLabel, CatalogFact, CatalogMetric, List[Union[Attribute, Metric, Filter, CatalogLabel, CatalogFact, CatalogMetric]], ExecutionDefinition]) → Dict[str, Set[str]]

Returns attributes, facts, and metrics which are valid to add to a context that already contains some entities from the semantic model. The entities are typically used to compute analytics and come from the execution definition. You may, however, specify the entities through different layers of convenience.

Parameters

- **workspace_id** – workspace identifier
- **ctx** – items already in context. you can specify context in one of the following ways:
 - single item or list of items from the execution model
 - single item or list of items from catalog model; catalog fact, label or metric may be added
 - the entire execution definition that is used to compute analytics

Returns

a dict of sets; type of available object is used as key in the dict, the value is a set containing id's of available items

get_full_catalog(*workspace_id*: str) → *CatalogWorkspaceContent*

Retrieves catalog for a workspace. Catalog contains all data sets and metrics defined in that workspace.

Parameters

workspace_id – workspace identifier

`gooddata_sdk.catalog.workspace.declarative_model`

Modules

`gooddata_sdk.catalog.workspace.
declarative_model.workspace`

`gooddata_sdk.catalog.workspace.declarative_model.workspace`

Modules

`gooddata_sdk.catalog.workspace.
declarative_model.workspace.
analytics_model`

`gooddata_sdk.catalog.workspace.
declarative_model.workspace.logical_model`

`gooddata_sdk.catalog.workspace.
declarative_model.workspace.workspace`

`gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model`

Modules

`gooddata_sdk.catalog.workspace.
declarative_model.workspace.
analytics_model.analytics_model`

`gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model`

Classes

`CatalogAnalyticsBase(*, id)`

`CatalogDeclarativeAnalyticalDashboard(*, id,
...)`

`CatalogDeclarativeAnalytics(*[, analytics])`

`CatalogDeclarativeAnalyticsLayer(*[, ...])`

`CatalogDeclarativeDashboardPlugin(*, id, ...)`

`CatalogDeclarativeFilterContext(*, id, ...)`

`CatalogDeclarativeMetric(*, id, title, content)`

`CatalogDeclarativeVisualizationObject(*, id,
...)`

`gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogAnalyticsBase`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogAnalyticsBase`

Bases: `Base`

`__init__(*, id: str) → None`

Method generated by attrs for class `CatalogAnalyticsBase`.

Methods

<code>__init__(*, id)</code>	Method generated by attrs for class CatalogAnalyticsBase.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(analytics_file)</code>	
<code>store_to_disk(analytics_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>	
-----------------	--

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeAnalyticalDashboard`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeAnalyticalDashboard`

Bases: `CatalogAnalyticsBase`

`__init__`(*, *id*: str, *title*: str, *content*: Dict[str, Any], *description*: Optional[str] = None, *tags*: Optional[List[str]] = None) → None

Method generated by attrs for class `CatalogDeclarativeAnalyticalDashboard`.

Methods

<code>__init__</code> (*, <i>id</i> , <i>title</i> , <i>content</i> [, ...])	Method generated by attrs for class <code>CatalogDeclarativeAnalyticalDashboard</code> .
<code>client_class</code> ()	
<code>from_api</code> (entity)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (data[, <i>camel_case</i>])	Creates object from dictionary.
<code>load_from_disk</code> (analytics_file)	
<code>store_to_disk</code> (analytics_folder)	
<code>to_api</code> ()	
<code>to_dict</code> ([<i>camel_case</i>])	Converts object into dictionary.

Attributes

`id`

`title`

`content`

`description`

`tags`

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeAnalyticsLayer`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeAnalyticsLayer`

Bases: `Base`

`__init__(*, analytics: Optional[CatalogDeclarativeAnalyticsLayer] = None) → None`

Method generated by attrs for class CatalogDeclarativeAnalytics.

Methods

<code>__init__</code> (*[, analytics])	Method generated by attrs for class CatalogDeclarativeAnalytics.
<code>client_class</code> ()	
<code>from_api</code> (entity)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (data[, camel_case])	Creates object from dictionary.
<code>load_from_disk</code> (workspace_folder)	
<code>store_to_disk</code> (workspace_folder)	
<code>to_api</code> ()	
<code>to_dict</code> ([camel_case])	Converts object into dictionary.

Attributes

<code>analytics</code>

classmethod `from_api`(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict`(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeAnalyticsLayer`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeAnalyticsLayer`

Bases: `Base`

```
__init__(*, analytical_dashboards: List[CatalogDeclarativeAnalyticalDashboard] = NOTHING,
          dashboard_plugins: List[CatalogDeclarativeDashboardPlugin] = NOTHING, filter_contexts:
          List[CatalogDeclarativeFilterContext] = NOTHING, metrics: List[CatalogDeclarativeMetric] =
          NOTHING, visualization_objects: List[CatalogDeclarativeVisualizationObject] = NOTHING) →
          None
```

Method generated by attrs for class `CatalogDeclarativeAnalyticsLayer`.

Methods

<code>__init__(*[, analytical_dashboards, ...])</code>	Method generated by attrs for class CatalogDeclarativeAnalyticsLayer.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>get_analytical_dashboards_folder(...)</code>	
<code>get_analytics_model_folder(workspace_folder)</code>	
<code>get_dashboard_plugins_folder(...)</code>	
<code>get_filter_contexts_folder(...)</code>	
<code>get_metrics_folder(analytics_model_folder)</code>	
<code>get_visualization_objects_folder(...)</code>	
<code>load_from_disk(workspace_folder)</code>	
<code>store_to_disk(workspace_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>analytical_dashboards</code>
<code>dashboard_plugins</code>
<code>filter_contexts</code>
<code>metrics</code>
<code>visualization_objects</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeDashboardPlugin`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeDashboardPlugin`

Bases: `CatalogAnalyticsBase`

`__init__`(**id*: *str*, *title*: *str*, *content*: *Dict*[*str*, *Any*], *description*: *Optional*[*str*] = *None*, *tags*: *Optional*[*List*[*str*]] = *None*) → *None*

Method generated by attrs for class `CatalogDeclarativeDashboardPlugin`.

Methods

<code>__init__</code> (* <i>id</i> , <i>title</i> , <i>content</i> [, ...])	Method generated by attrs for class <code>CatalogDeclarativeDashboardPlugin</code> .
<code>client_class</code> ()	
<code>from_api</code> (<i>entity</i>)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (<i>data</i> [, <i>camel_case</i>])	Creates object from dictionary.
<code>load_from_disk</code> (<i>analytics_file</i>)	
<code>store_to_disk</code> (<i>analytics_folder</i>)	
<code>to_api</code> ()	
<code>to_dict</code> ([<i>camel_case</i>])	Converts object into dictionary.

Attributes

id
title
content
description
tags

classmethod from_api(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod from_dict(*data: Dict[str, Any], camel_case: bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeModel

class gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeModel

Bases: *CatalogAnalyticsBase*

__init__(**, id: str, title: str, content: Dict[str, Any], description: Optional[str] = None, tags: Optional[List[str]] = None*) → None

Method generated by attrs for class CatalogDeclarativeFilterContext.

Methods

<code>__init__(*, id, title, content[, ...])</code>	Method generated by attrs for class CatalogDeclarativeFilterContext.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(analytics_file)</code>	
<code>store_to_disk(analytics_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>title</code>
<code>content</code>
<code>description</code>
<code>tags</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict `(camel_case: bool = True) → Dict[str, Any]`

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeFilterContext`

```
class gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.Catalog
```

Bases: *CatalogAnalyticsBase*

```
__init__(*, id: str, title: str, content: Dict[str, Any], description: Optional[str] = None, tags:
Optional[List[str]] = None) → None
```

Method generated by attrs for class CatalogDeclarativeMetric.

Methods

<code><i>__init__</i>(*, id, title, content[, ...])</code>	Method generated by attrs for class CatalogDeclarativeMetric.
<code>client_class()</code>	
<code><i>from_api</i>(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code><i>from_dict</i>(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(analytics_file)</code>	
<code>store_to_disk(analytics_folder)</code>	
<code>to_api()</code>	
<code><i>to_dict</i>([camel_case])</code>	Converts object into dictionary.

Attributes

id
title
content
description
tags

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeVisualizationObject`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model.CatalogDeclarativeVisualizationObject`

Bases: `CatalogAnalyticsBase`

__init__(`*, id: str, title: str, content: Dict[str, Any], description: Optional[str] = None, tags: Optional[List[str]] = None`) → None

Method generated by attrs for class CatalogDeclarativeVisualizationObject.

Methods

<code>__init__(*, id, title, content[, ...])</code>	Method generated by attrs for class CatalogDeclarativeVisualizationObject.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(analytics_file)</code>	
<code>store_to_disk(analytics_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>title</code>
<code>content</code>
<code>description</code>
<code>tags</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict `(camel_case: bool = True) → Dict[str, Any]`

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model`

Modules

```
gooddata_sdk.catalog.workspace.
declarative_model.workspace.logical_model.
dataset
```

```
gooddata_sdk.catalog.workspace.
declarative_model.workspace.logical_model.
date_dataset
```

```
gooddata_sdk.catalog.workspace.
declarative_model.workspace.logical_model.
ldm
```

gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset

Modules

```
gooddata_sdk.catalog.workspace.
declarative_model.workspace.logical_model.
dataset.dataset
```

gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset

Classes

```
CatalogDataSourceTableIdentifier(*, id, ...)
```

```
CatalogDeclarativeAttribute(*, id, title, ...)
```

```
CatalogDeclarativeDataset(*, id, title, ...)
```

```
CatalogDeclarativeFact(*, id, title, ..., ...)
```

```
CatalogDeclarativeLabel(*, id, title, ..., ...)
```

```
CatalogDeclarativeReference(*, identifier, ...)
```

gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogDataSourceTableIdentifier

class gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogDataSourceTableIdentifier

Bases: *Base*

__init__(*, id: str, data_source_id: str) → None

Method generated by attrs for class CatalogDataSourceTableIdentifier.

Methods

<code>__init__(*, id, data_source_id)</code>	Method generated by attrs for class <code>CatalogData-SourceTableIdentifier</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>data_source_id</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogDeclarative`


```
class gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogD
```

Bases: [Base](#)

```
__init__(*, id: str, title: str, source_column: str, labels: List[CatalogDeclarativeLabel], default_view:
    Optional[CatalogLabelIdentifier] = None, sort_column: Optional[str] = None, sort_direction:
    Optional[str] = None, description: Optional[str] = None, tags: Optional[List[str]] = None) →
    None
```

Method generated by attrs for class CatalogDeclarativeAttribute.

Methods

<code>__init__(*, id, title, source_column, labels)</code>	Method generated by attrs for class <code>CatalogDeclarativeAttribute</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>title</code>
<code>source_column</code>
<code>labels</code>
<code>default_view</code>
<code>sort_column</code>
<code>sort_direction</code>
<code>description</code>
<code>tags</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogDeclarative`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogD`

Bases: [Base](#)

```
__init__(*, id: str, title: str, grain: List[CatalogGrainIdentifier], references:
    List[CatalogDeclarativeReference], description: Optional[str] = None, attributes:
    Optional[List[CatalogDeclarativeAttribute]] = None, facts:
    Optional[List[CatalogDeclarativeFact]] = None, data_source_table_id:
    Optional[CatalogDataSourceTableIdentifier] = None, tags: Optional[List[str]] = None) → None
```

Method generated by attrs for class CatalogDeclarativeDataset.

Methods

<code>__init__(*, id, title, grain, references[, ...])</code>	Method generated by attrs for class CatalogDeclarativeDataset.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(dataset_file)</code>	
<code>store_to_disk(datasets_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>title</code>
<code>grain</code>
<code>references</code>
<code>description</code>
<code>attributes</code>
<code>facts</code>
<code>data_source_table_id</code>
<code>tags</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogDeclarative`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogD`

Bases: `Base`

`__init__`(*, *id*: str, *title*: str, *source_column*: str, *description*: Optional[str] = None, *tags*: Optional[List[str]] = None) → None

Method generated by attrs for class CatalogDeclarativeFact.

Methods

<code>__init__</code> (*, <i>id</i> , <i>title</i> , <i>source_column</i> [, ...])	Method generated by attrs for class CatalogDeclarativeFact.
<code>client_class</code> ()	
<code>from_api</code> (entity)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (data[, <i>camel_case</i>])	Creates object from dictionary.
<code>to_api</code> ()	
<code>to_dict</code> ([<i>camel_case</i>])	Converts object into dictionary.

Attributes

`id`

`title`

`source_column`

`description`

`tags`

classmethod `from_api`(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict`(*data: Dict[str, Any], camel_case: bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogDeclarative`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogD`

Bases: [Base](#)

```
__init__(*, id: str, title: str, source_column: str, description: Optional[str] = None, tags:
Optional[List[str]] = None, value_type: Optional[str] = None) → None
```

Method generated by attrs for class CatalogDeclarativeLabel.

Methods

<code>__init__(*, id, title, source_column[, ...])</code>	Method generated by attrs for class CatalogDeclarativeLabel.
<code>client_class()</code>	
<code><i>from_api</i>(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code><i>from_dict</i>(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code><i>to_dict</i>([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>title</code>
<code>source_column</code>
<code>description</code>
<code>tags</code>
<code>value_type</code>

classmethod from_api(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod from_dict(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogDeclarative`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset.CatalogD`

Bases: `Base`

`__init__`(*, *identifier*: `CatalogReferenceIdentifier`, *multivalue*: `bool`, *source_columns*: `List[str]`) → None
Method generated by attrs for class `CatalogDeclarativeReference`.

Methods

<code>__init__</code> (*, <i>identifier</i> , <i>multivalue</i> , ...)	Method generated by attrs for class <code>CatalogDeclarativeReference</code> .
<code>client_class</code> ()	
<code>from_api</code> (<i>entity</i>)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (<i>data</i> [, <i>camel_case</i>])	Creates object from dictionary.
<code>to_api</code> ()	
<code>to_dict</code> ([<i>camel_case</i>])	Converts object into dictionary.

Attributes

<code>identifier</code>
<code>multivalue</code>
<code>source_columns</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.date_dataset

Modules

*gooddata_sdk.catalog.workspace.
declarative_model.workspace.logical_model.
date_dataset.date_dataset*

gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.date_dataset.date_dataset

Classes

CatalogDeclarativeDateDataset(, id, title, ...)*

CatalogGranularitiesFormatting(, ...)*

gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.date_dataset.date_dataset.Catalog

```
class gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.date_dataset.date_dataset
```

Bases: [Base](#)

```
__init__(*, id: str, title: str, granularities_formatting: CatalogGranularitiesFormatting, granularities:
    List[str], description: Optional[str] = None, tags: Optional[List[str]] = None) → None
```

Method generated by attrs for class CatalogDeclarativeDateDataset.

Methods

<code>__init__(*, id, title, ...[, description, tags])</code>	Method generated by attrs for class CatalogDeclarativeDateDataset.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(date_instance_file)</code>	
<code>store_to_disk(date_instances_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>title</code>
<code>granularities_formatting</code>
<code>granularities</code>
<code>description</code>
<code>tags</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.date_dataset.date_dataset.Catalog`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.date_dataset.date_dataset`

Bases: *Base*

`__init__(*, title_base: str, title_pattern: str) → None`

Method generated by attrs for class CatalogGranularitiesFormatting.

Methods

<code>__init__(*, title_base, title_pattern)</code>	Method generated by attrs for class CatalogGranularitiesFormatting.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>title_base</code>
<code>title_pattern</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.ldm**Classes**

`CatalogDeclarativeLdm(*[, datasets, ...])`

`CatalogDeclarativeModel(*[, ldm])`

gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.ldm.CatalogDeclarativeLdm
class gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.ldm.CatalogDeclarativeLdm
Bases: *Base*
__init__(*[, datasets: List[CatalogDeclarativeDataset] = NOTHING, date_instances:
List[CatalogDeclarativeDateDataset] = NOTHING) → None

Method generated by attrs for class CatalogDeclarativeLdm.

Methods

<code>__init__(*[, datasets, date_instances])</code>	Method generated by attrs for class CatalogDeclarativeLdm.
--	--

<code>client_class()</code>	
-----------------------------	--

<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
-------------------------------	---

<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
--	---------------------------------

<code>get_datasets_folder(ldm_folder)</code>	
--	--

<code>get_date_instances_folder(ldm_folder)</code>	
--	--

<code>get_ldm_folder(workspace_folder)</code>	
---	--

<code>load_from_disk(workspace_folder)</code>	
---	--

<code>store_to_disk(workspace_folder)</code>	
--	--

<code>to_api()</code>	
-----------------------	--

<code>to_dict([camel_case])</code>	Converts object into dictionary.
------------------------------------	----------------------------------

Attributes

`datasets`

`date_instances`

classmethod `from_api`(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict`(*data: Dict[str, Any]*, *camel_case: bool = True*) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.ldm.CatalogDeclarativeModel`

class `gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.ldm.CatalogDeclarativeModel`

Bases: `Base`

__init__(*, *ldm: Optional[CatalogDeclarativeLdm] = None*) → None

Method generated by attrs for class CatalogDeclarativeModel.

Methods

<code>__init__</code> (*, ldm)	Method generated by attrs for class CatalogDeclarativeModel.
--------------------------------	--

`client_class()`

<code>from_api</code> (entity)	Creates object from entity passed by client class, which represents it as dictionary.
--------------------------------	---

<code>from_dict</code> (data[, camel_case])	Creates object from dictionary.
---	---------------------------------

`load_from_disk`(workspace_folder)

`modify_mapped_data_source`(data_source_mapping)

`store_to_disk`(workspace_folder)

`to_api`()

<code>to_dict</code> ([camel_case])	Converts object into dictionary.
-------------------------------------	----------------------------------

Attributes

`ldm`

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace`

Functions

`get_workspace_folder(workspace_id, ...)`

`gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.get_workspace_folder`

`gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.get_workspace_folder(workspace_id: str, layout_organization: Path) → Path`

Classes

`CatalogDeclarativeWorkspace(*, id, name[, ...])`

`CatalogDeclarativeWorkspaceDataFilter(*, id, ...)`

`CatalogDeclarativeWorkspaceDataFilterSetting(*, ...)`

`CatalogDeclarativeWorkspaceDataFilters(*, ...)`

`CatalogDeclarativeWorkspaceModel(*[, ldm, ...])`

`CatalogDeclarativeWorkspaces(*, workspaces, ...)`

`gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspace`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspace(`

Bases: [Base](#)

```
__init__(*, id: str, name: str, model: Optional[CatalogDeclarativeWorkspaceModel] = None, parent:
    Optional[CatalogWorkspaceIdentifier] = None, permissions:
    List[CatalogDeclarativeSingleWorkspacePermission] = NOTHING, hierarchy_permissions:
    List[CatalogDeclarativeWorkspaceHierarchyPermission] = NOTHING, early_access:
    Optional[str] = None, settings: List[CatalogDeclarativeSetting] = NOTHING) → None
```

Method generated by attrs for class CatalogDeclarativeWorkspace.

Methods

<code>__init__(*, id, name[, model, parent, ...])</code>	Method generated by attrs for class CatalogDeclarativeWorkspace.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(workspaces_folder, workspace_id)</code>	
<code>store_to_disk(workspaces_folder)</code>	
<code>to_api([include_nested_structures])</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>name</code>
<code>model</code>
<code>parent</code>
<code>permissions</code>
<code>hierarchy_permissions</code>
<code>early_access</code>
<code>settings</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspaceDataFilter`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspaceDataFilter`

Bases: `Base`

`__init__`(**id*: *str*, *title*: *str*, *column_name*: *str*, *workspace_data_filter_settings*: *List*[`CatalogDeclarativeWorkspaceDataFilterSetting`], *description*: *Optional*[*str*] = *None*, *workspace*: *Optional*[`CatalogWorkspaceIdentifier`] = *None*) → *None*

Method generated by attrs for class `CatalogDeclarativeWorkspaceDataFilter`.

Methods

<code>__init__</code> (* <i>id</i> , <i>title</i> , <i>column_name</i> , ...[, ...])	Method generated by attrs for class <code>CatalogDeclarativeWorkspaceDataFilter</code> .
<code>client_class</code> ()	
<code>from_api</code> (<i>entity</i>)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (<i>data</i> [, <i>camel_case</i>])	
	param data Data loaded for example from the file.
<code>load_from_disk</code> (<i>workspaces_data_filter_file</i>)	
<code>store_to_disk</code> (<i>workspaces_data_filters_folder</i>)	
<code>to_api</code> ()	
<code>to_dict</code> ([<i>camel_case</i>])	Converts object into dictionary.

Attributes

id
title
column_name
workspace_data_filter_settings
description
workspace

classmethod `from_api`(*entity: Dict[str, Any]*) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict`(*data: dict[str, Any], camel_case: bool = True*) → *CatalogDeclarativeWorkspaceDataFilter*

Parameters

- **data** – Data loaded for example from the file.
- **camel_case** – True if the variable names in the input data are serialized names as specified in the OpenAPI document. False if the variables names in the input data are python variable names in PEP-8 snake case.

Returns

CatalogDeclarativeWorkspaceDataFilter object.

to_dict(*camel_case: bool = True*) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspaceDataFilterSetting`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspaceDataFilterSetting`

Bases: `Base`

`__init__`(**id*: *str*, *title*: *str*, *filter_values*: *List[str]*, *workspace*: `CatalogWorkspaceIdentifier`, *description*: *Optional[str]* = *None*) → *None*

Method generated by attrs for class `CatalogDeclarativeWorkspaceDataFilterSetting`.

Methods

<code>__init__</code> (* <i>id</i> , <i>title</i> , <i>filter_values</i> , <i>workspace</i>)	Method generated by attrs for class <code>CatalogDeclarativeWorkspaceDataFilterSetting</code> .
<code>client_class</code> ()	
<code>from_api</code> (<i>entity</i>)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (<i>data</i> [, <i>camel_case</i>])	Creates object from dictionary.
<code>to_api</code> ()	
<code>to_dict</code> ([<i>camel_case</i>])	Converts object into dictionary.

Attributes

<code>id</code>
<code>title</code>
<code>filter_values</code>
<code>workspace</code>
<code>description</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspaceDataFilter`

`class gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspaceDataFilters`

Bases: `Base`

__init__(`*`, `workspace_data_filters: List[CatalogDeclarativeWorkspaceDataFilter]`) → None

Method generated by attrs for class CatalogDeclarativeWorkspaceDataFilters.

Methods

<code>__init__</code> (<code>*</code> , <code>workspace_data_filters</code>)	Method generated by attrs for class CatalogDeclarativeWorkspaceDataFilters.
<code>client_class</code> ()	
<code>from_api</code> (<code>entity</code>)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (<code>data</code> [, <code>camel_case</code>])	Creates object from dictionary.
<code>load_from_disk</code> (<code>layout_organization_folder</code>)	
<code>store_to_disk</code> (<code>layout_organization_folder</code>)	
<code>to_api</code> ()	
<code>to_dict</code> ([<code>camel_case</code>])	Converts object into dictionary.

Attributes

`workspace_data_filters`

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(`camel_case: bool = True`) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspaceModel

class `gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspaceModel`

Bases: `Base`

__init__(`*, ldm: Optional[CatalogDeclarativeLdm] = None, analytics: Optional[CatalogDeclarativeAnalyticsLayer] = None`) → None

Method generated by attrs for class CatalogDeclarativeWorkspaceModel.

Methods

<code>__init__(*[, ldm, analytics])</code>	Method generated by attrs for class CatalogDeclarativeWorkspaceModel.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(workspace_folder)</code>	
<code>store_to_disk(workspace_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>ldm</code>
<code>analytics</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict `(camel_case: bool = True) → Dict[str, Any]`

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspaces`

class `gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace.CatalogDeclarativeWorkspaces`

Bases: `Base`

__init__ `(*[, workspaces: List[CatalogDeclarativeWorkspace], workspace_data_filters: List[CatalogDeclarativeWorkspaceDataFilter]) → None`

Method generated by attrs for class CatalogDeclarativeWorkspaces.

Methods

<code>__init__(*, workspaces, workspace_data_filters)</code>	Method generated by attrs for class CatalogDeclarativeWorkspaces.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>load_from_disk(layout_organization_folder)</code>	
<code>store_to_disk(layout_organization_folder)</code>	
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.
<code>workspace_data_filters_folder(...)</code>	
<code>workspaces_folder(layout_organization_folder)</code>	

Attributes

<code>workspaces</code>
<code>workspace_data_filters</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.entity_model`

Modules

<code>gooddata_sdk.catalog.workspace.entity_model.content_objects</code>
<code>gooddata_sdk.catalog.workspace.entity_model.graph_objects</code>
<code>gooddata_sdk.catalog.workspace.entity_model.workspace</code>

gooddata_sdk.catalog.workspace.entity_model.content_objects**Modules**

```
gooddata_sdk.catalog.workspace.  
entity_model.content_objects.dataset
```

```
gooddata_sdk.catalog.workspace.  
entity_model.content_objects.metric
```

gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset**Classes**

```
CatalogAttribute(entity, labels)
```

```
CatalogDataset(entity, attributes, facts)
```

```
CatalogFact(entity)
```

```
CatalogLabel(entity)
```

gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset.CatalogAttribute

```
class gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset.CatalogAttribute(entity: dict[str, Any], labels: list[CatalogLabel])
```

Bases: *CatalogEntity*

```
__init__(entity: dict[str, Any], labels: list[CatalogLabel]) → None
```

Methods

```
__init__(entity, labels)
```

```
as_computable()
```

```
find_label(id_obj)
```

```
primary_label()
```

Attributes

dataset
description
granularity
id
labels
obj_id
title
type

`gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset.CatalogDataset`

```
class gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset.CatalogDataset(entity: dict[str, Any], attributes: list[CatalogAttribute], facts: list[CatalogFact])
```

Bases: `CatalogEntity`

`__init__(entity: dict[str, Any], attributes: list[CatalogAttribute], facts: list[CatalogFact])` → None

Methods

<code>__init__(entity, attributes, facts)</code>	
<code>filter_dataset(valid_objects)</code>	Filters dataset so that it contains only attributes and facts that are part of the provided valid objects structure.
<code>find_label_attribute(id_obj)</code>	

Attributes

attributes
data_type
description
facts
id
obj_id
title
type

filter_dataset(*valid_objects: Dict[str, Set[str]]*) → Optional[*CatalogDataset*]

Filters dataset so that it contains only attributes and facts that are part of the provided valid objects structure.

Parameters

valid_objects – mapping of object type to a set of valid object ids

Returns

CatalogDataset containing only valid attributes and facts; None if all of the attributes and facts were filtered out

gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset.CatalogFact

```
class gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset.CatalogFact(entity:
dict[str,
Any])
```

Bases: *CatalogEntity*

__init__(*entity: dict[str, Any]*) → None

Methods

<code>__init__</code> (entity)
as_computable()

Attributes

description

id

obj_id

title

type

gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset.CatalogLabel

```
class gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset.CatalogLabel(entity: dict[str, Any])
```

Bases: *CatalogEntity*

`__init__(entity: dict[str, Any]) → None`

Methods

`__init__(entity)`

`as_computable()`

Attributes

description

id

obj_id

primary

title

type

`gooddata_sdk.catalog.workspace.entity_model.content_objects.metric`

Classes

`CatalogMetric(entity)`

`gooddata_sdk.catalog.workspace.entity_model.content_objects.metric.CatalogMetric`

class `gooddata_sdk.catalog.workspace.entity_model.content_objects.metric.CatalogMetric`(*entity*: *dict[str, Any]*)

Bases: `CatalogEntity`

`__init__`(*entity*: *dict[str, Any]*) → None

Methods

`__init__`(*entity*)

`as_computable`()

Attributes

`description`

`format`

`id`

`obj_id`

`title`

`type`

gooddata_sdk.catalog.workspace.entity_model.graph_objects

Modules

*gooddata_sdk.catalog.workspace.
entity_model.graph_objects.graph*

gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph

Classes

CatalogDependentEntitiesGraph([, nodes,
edges])*

CatalogDependentEntitiesNode(, id, type[, ...])*

CatalogDependentEntitiesRequest([, identi-
fiers])*

CatalogDependentEntitiesResponse(, graph)*

CatalogEntityIdentifier(, id, type)*

gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph.CatalogDependentEntitiesGraph

```
class gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph.CatalogDependentEntitiesGraph(*,
node: CatalogEntityIdentifier, edges: List[List[CatalogEntityIdentifier]] = NOTHING) → None
    """
    A graph object representing a catalog dependent entities graph.
    The graph is composed of nodes and edges.
    The nodes are represented by CatalogEntityIdentifier objects.
    The edges are represented by List[CatalogEntityIdentifier] objects.
    The graph is initialized with a list of nodes and a list of edges.
    The graph is mutable and can be modified after creation.
    The graph is immutable and cannot be modified after creation.
    """
```

Bases: *Base*

```
__init__(*, nodes: List[CatalogDependentEntitiesNode] = NOTHING, edges:
List[List[CatalogEntityIdentifier]] = NOTHING) → None
```

Method generated by attrs for class CatalogDependentEntitiesGraph.

Methods

<code>__init__</code> (*[, nodes, edges])	Method generated by attrs for class CatalogDependentEntitiesGraph.
<code>client_class</code> ()	
<code>from_api</code> (entity)	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict</code> (data[, camel_case])	Creates object from dictionary.
<code>to_api</code> ()	
<code>to_dict</code> ([camel_case])	Converts object into dictionary.

Attributes

<code>nodes</code>
<code>edges</code>

classmethod `from_api`(entity: Dict[str, Any]) → T

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict`(data: Dict[str, Any], camel_case: bool = True) → T

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph.CatalogDependentEntitiesNode`

```
class gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph.CatalogDependentEntitiesNode(*,
id: str,
type: str,
title: Optional[str] = None)
```

Bases: `Base`

__init__(*[, id: str, type: str, title: Optional[str] = None) → None

Method generated by attrs for class CatalogDependentEntitiesNode.

Methods

<code>__init__(*, id, type[, title])</code>	Method generated by attrs for class <code>CatalogDependentEntitiesNode</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>type</code>
<code>title</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph.CatalogDependentEntitiesRequest`

class `gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph.CatalogDependentEntitiesRequest(*,`

Bases: `Base`

__init__(`*, identifiers: List[CatalogEntityIdentifier] = NOTHING`) → None

Method generated by attrs for class `CatalogDependentEntitiesRequest`.

Methods

<code>__init__(*[, identifiers])</code>	Method generated by attrs for class <code>CatalogDependentEntitiesRequest</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>identifiers</code>	
--------------------------	--

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph.CatalogDependentEntitiesResponse`

class `gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph.CatalogDependentEntitiesResponse(`

Bases: `Base`

__init__(`*`, `graph: CatalogDependentEntitiesGraph`) → None

Method generated by attrs for class `CatalogDependentEntitiesResponse`.

Methods

<code>__init__(*, graph)</code>	Method generated by attrs for class <code>CatalogDependentEntitiesResponse</code> .
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>graph</code>	
--------------------	--

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

`gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph.CatalogEntityIdentifier`

```
class gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph.CatalogEntityIdentifier(*,
                                                                                               id:
                                                                                               str,
                                                                                               type:
                                                                                               str)
```

Bases: `Base`

__init__(*, id: str, type: str) → None

Method generated by attrs for class `CatalogEntityIdentifier`.

Methods

<code>__init__(*, id, type)</code>	Method generated by attrs for class CatalogEntityIdentifier.
<code>client_class()</code>	
<code>from_api(entity)</code>	Creates object from entity passed by client class, which represents it as dictionary.
<code>from_dict(data[, camel_case])</code>	Creates object from dictionary.
<code>to_api()</code>	
<code>to_dict([camel_case])</code>	Converts object into dictionary.

Attributes

<code>id</code>
<code>type</code>

classmethod `from_api(entity: Dict[str, Any]) → T`

Creates object from entity passed by client class, which represents it as dictionary.

classmethod `from_dict(data: Dict[str, Any], camel_case: bool = True) → T`

Creates object from dictionary. It needs to be specified if the dictionary is in camelCase or snake_case.

to_dict(camel_case: bool = True) → Dict[str, Any]

Converts object into dictionary. Optional argument if the dictionary should be camelCase or snake_case can be specified.

gooddata_sdk.catalog.workspace.entity_model.workspace

Classes

<code>CatalogWorkspace(workspace_id, name[, parent_id])</code>
--

gooddata_sdk.catalog.workspace.entity_model.workspace.CatalogWorkspace

class `gooddata_sdk.catalog.workspace.entity_model.workspace.CatalogWorkspace(workspace_id: str, name: str, parent_id: Optional[str] = None)`

Bases: `CatalogNameEntity`

`__init__(workspace_id: str, name: str, parent_id: Optional[str] = None)`

Methods

`__init__(workspace_id, name[, parent_id])`

`from_api(entity)`

`to_api()`

`gooddata_sdk.catalog.workspace.model_container`

Classes

`CatalogWorkspaceContent(valid_obj_fun, ...)`

`gooddata_sdk.catalog.workspace.model_container.CatalogWorkspaceContent`

```
class gooddata_sdk.catalog.workspace.model_container.CatalogWorkspaceContent(valid_obj_fun:
                                                                              func-
                                                                              tools.partial[dict[str,
                                                                              set[str]]],
                                                                              datasets:
                                                                              list[CatalogDataset],
                                                                              metrics:
                                                                              list[CatalogMetric])
```

Bases: `object`

```
__init__(valid_obj_fun: functools.partial[dict[str, set[str]]], datasets: list[CatalogDataset], metrics:
          list[CatalogMetric]) → None
```

Methods

`__init__(valid_obj_fun, datasets, metrics)`

<code>catalog_with_valid_objects(ctx)</code>	Returns a new instance of catalog which contains only those datasets (attributes and facts) that are valid in the provided context.
--	---

`create_workspace_content_catalog(...)`

<code>find_label_attribute(id_obj)</code>	Get attribute by label id.
---	----------------------------

<code>get_dataset(dataset_id)</code>	Gets dataset by id.
--------------------------------------	---------------------

<code>get_metric(metric_id)</code>	Gets metric by id.
------------------------------------	--------------------

Attributes

datasets

metrics

catalog_with_valid_objects(*ctx*: Union[Attribute, Metric, Filter, CatalogLabel, CatalogFact, CatalogMetric, List[Union[Attribute, Metric, Filter, CatalogLabel, CatalogFact, CatalogMetric]], ExecutionDefinition]) → CatalogWorkspaceContent

Returns a new instance of catalog which contains only those datasets (attributes and facts) that are valid in the provided context. The context is composed of one more more entities of the semantic model and the filtered catalog will contain only those entities that can be safely added on top of that existing context.

Parameters

ctx – existing context. you can specify context in one of the following ways:

- single item or list of items from the execution model
- single item or list of items from catalog model; catalog fact, label or metric may be added
- the entire execution definition that is used to compute analytics

find_label_attribute(*id_obj*: Union[str, ObjId, Dict[str, Dict[str, str]], Dict[str, str]]) → Optional[CatalogAttribute]

Get attribute by label id.

get_dataset(*dataset_id*: Union[str, ObjId]) → Optional[CatalogDataset]

Gets dataset by id. The id can be either an instance of ObjId or string containing serialized ObjId ('dataset/some.dataset.id') or contain just the id part (some.dataset.id).

Parameters

dataset_id – fully qualified dataset entity id (type/id) or just the identifier of dataset entity

Returns

instance of CatalogDataset or None if no such dataset in catalog

Return type

CatalogDataset

get_metric(*metric_id*: Union[str, ObjId]) → Optional[CatalogMetric]

Gets metric by id. The id can be either an instance of ObjId or string containing serialized ObjId ('metric/some.metric.id') or contain just the id part ('some.metric.id').

Parameters

metric_id – fully qualified metric entity id (type/id) or just the identifier of metric entity

Returns

instance of CatalogMetric or None if no such metric in catalog

Return type

CatalogMetric

`gooddata_sdk.catalog.workspace.service`

Classes

`CatalogWorkspaceService(api_client)`

`gooddata_sdk.catalog.workspace.service.CatalogWorkspaceService`

```
class gooddata_sdk.catalog.workspace.service.CatalogWorkspaceService(api_client:  
                                                                    GoodDataApiClient)
```

Bases: `CatalogServiceBase`

`__init__`(*api_client: GoodDataApiClient*) → None

Methods

<code>__init__(api_client)</code>	
<code>create_or_update(workspace)</code>	
<code>delete_workspace(workspace_id)</code>	This method is implemented according to our implementation of delete workspace, which returns HTTP 204 no matter if the workspace_id exists.
<code>get_declarative_workspace(workspace_id)</code>	
<code>get_declarative_workspace_data_filters()</code>	
<code>get_declarative_workspaces()</code>	
<code>get_organization()</code>	
<code>get_workspace(workspace_id)</code>	Gets workspace content and returns it as Catalog-Workspace object.
<code>layout_organization_folder(layout_root_path)</code>	
<code>list_workspaces()</code>	
<code>load_and_put_declarative_workspace(workspace_id)</code>	
<code>load_and_put_declarative_workspace_data_filters([...])</code>	
<code>load_and_put_declarative_workspaces([...])</code>	
<code>load_declarative_workspace(workspace_id[, ...])</code>	
<code>load_declarative_workspace_data_filters([...])</code>	
<code>load_declarative_workspaces([layout_root_path])</code>	
<code>put_declarative_workspace(workspace_id, ...)</code>	
<code>put_declarative_workspace_data_filters(...)</code>	
<code>put_declarative_workspaces(workspace)</code>	
<code>store_declarative_workspace(workspace_id[, ...])</code>	
<code>store_declarative_workspace_data_filters([...])</code>	
<code>store_declarative_workspaces([layout_root_path])</code>	

Attributes

`organization_id`

delete_workspace(*workspace_id: str*) → None

This method is implemented according to our implementation of delete workspace, which returns HTTP 204 no matter if the `workspace_id` exists.

get_workspace(*workspace_id: str*) → *CatalogWorkspace*

Gets workspace content and returns it as `CatalogWorkspace` object.

Parameters

workspace_id – An input string parameter of workspace id.

Returns

`CatalogWorkspace` object containing structure of workspace.

4.2.2 gooddata_sdk.client

Module containing a class that provides access to metadata and afm services.

Classes

<i>GoodDataApiClient</i> (<i>host, token[, ...]</i>)	Provide access to metadata and afm services.
--	--

gooddata_sdk.client.GoodDataApiClient

class `gooddata_sdk.client.GoodDataApiClient`(*host: str, token: str, custom_headers: Optional[dict[str, str]] = None, extra_user_agent: Optional[str] = None*)

Bases: `object`

Provide access to metadata and afm services.

__init__(*host: str, token: str, custom_headers: Optional[dict[str, str]] = None, extra_user_agent: Optional[str] = None*) → None

Take url, token for connecting to GoodData.CN.

HTTP requests made by this class may be enriched by *custom_headers* dict containing header names as keys and header values as dict values.

extra_user_agent is optional string to be added to default http User-Agent header. This takes precedence over *custom_headers* setting.

Methods

<code>__init__(host, token[, custom_headers, ...])</code>	Take url, token for connecting to GoodData.CN.
---	--

Attributes

<code>afm_client</code>

<code>metadata_client</code>

<code>scan_client</code>

4.2.3 gooddata_sdk.compute

Modules

<code>gooddata_sdk.compute.model</code>

<code>gooddata_sdk.compute.service</code>

gooddata_sdk.compute.model

Modules

<code>gooddata_sdk.compute.model.attribute</code>

<code>gooddata_sdk.compute.model.base</code>
--

<code>gooddata_sdk.compute.model.execution</code>

<code>gooddata_sdk.compute.model.filter</code>
--

<code>gooddata_sdk.compute.model.metric</code>
--

gooddata_sdk.compute.model.attribute**Classes**

Attribute(local_id, label)

gooddata_sdk.compute.model.attribute.Attribute**class** gooddata_sdk.compute.model.attribute.**Attribute**(local_id: str, label: Union[ObjId, str])Bases: *ExecModelEntity***__init__**(local_id: str, label: Union[ObjId, str]) → None

Creates new attribute that can be used to slice or dice metric values during computation.

Parameters

- **local_id** – identifier of the attribute within the execution
- **label** – identifier of the label to use for slicing or dicing; specified either as ObjId or str containing the label id

Methods

__init__(local_id, label) Creates new attribute that can be used to slice or dice metric values during computation.

as_api_model()

has_same_label(other)

Attributes

label

local_id

gooddata_sdk.compute.model.base**Classes**

ExecModelEntity()

Filter()

ObjId(id, type)

gooddata_sdk.compute.model.base.ExecModelEntity**class** gooddata_sdk.compute.model.base.**ExecModelEntity**

Bases: object

__init__() → None**Methods****__init__**()

as_api_model()

gooddata_sdk.compute.model.base.Filter**class** gooddata_sdk.compute.model.base.**Filter**Bases: *ExecModelEntity***__init__**() → None**Methods****__init__**()

as_api_model()

is_noop()

Attributes

apply_on_result

gooddata_sdk.compute.model.base.ObjId**class** gooddata_sdk.compute.model.base.**ObjId**(*id: str, type: str*)

Bases: object

__init__(*id: str, type: str*) → None

Methods

`__init__(id, type)`

`as_afm_id()`

`as_afm_id_attribute()`

`as_afm_id_dataset()`

`as_afm_id_label()`

`as_identifier()`

Attributes

`id`

`type`

gooddata_sdk.compute.model.execution

Functions

<code>compute_model_to_api_model([attributes, ...])</code>	Transforms categorized execution model entities (attributes, metrics, facts) into an API model that can be used for computations of data results or computations of object availability.
--	--

gooddata_sdk.compute.model.execution.compute_model_to_api_model

`gooddata_sdk.compute.model.execution.compute_model_to_api_model(attributes: Optional[list[Attribute]] = None, metrics: Optional[list[Metric]] = None, filters: Optional[list[Filter]] = None) → models.AFM`

Transforms categorized execution model entities (attributes, metrics, facts) into an API model that can be used for computations of data results or computations of object availability.

Parameters

- **attributes** – optionally specify list of attributes
- **metrics** – optionally specify list of metrics
- **filters** – optionally specify list of filters

Classes

<i>BareExecutionResponse</i> (actions_api, ...)	Holds ExecutionResponse from triggered report computation and allows reading report's results.
<i>Execution</i> (actions_api, workspace_id, ...)	An envelope class holding execution related classes:
<i>ExecutionDefinition</i> (attributes, metrics, ...)	
<i>ExecutionResponse</i>	alias of <i>Execution</i>
<i>ExecutionResult</i> (result)	
<i>TotalDefinition</i> (local_id, aggregation, ...)	
<i>TotalDimension</i> (idx[, items])	

gooddata_sdk.compute.model.execution.BareExecutionResponse

```
class gooddata_sdk.compute.model.execution.BareExecutionResponse(actions_api: ActionsApi,
                                                                    workspace_id: str, response:
                                                                    AfmExecutionResponse)
```

Bases: object

Holds ExecutionResponse from triggered report computation and allows reading report's results.

```
__init__(actions_api: ActionsApi, workspace_id: str, response: AfmExecutionResponse)
```

Methods

<i>__init__</i> (actions_api, workspace_id, response)	
<i>read_result</i> (limit[, offset])	Reads from the execution result.

Attributes

dimensions	
result_id	
workspace_id	

```
read_result(limit: Union[int, list[int]], offset: Union[None, int, list[int]] = None) → ExecutionResult
```

Reads from the execution result.

gooddata_sdk.compute.model.execution.Execution

```
class gooddata_sdk.compute.model.execution.Execution(actions_api: ActionsApi, workspace_id: str,
                                                    exec_def: ExecutionDefinition, response:
                                                    AfmExecutionResponse)
```

Bases: object

An envelope class holding execution related classes:

- exec_def ExecutionDefinition
- bare_exec_response BareExecutionResponse

```
__init__(actions_api: ActionsApi, workspace_id: str, exec_def: ExecutionDefinition, response:
         AfmExecutionResponse)
```

Methods

```
__init__(actions_api, workspace_id, ...)
```

```
read_result(limit[, offset])
```

Attributes

```
bare_exec_response
```

```
dimensions
```

```
exec_def
```

```
result_id
```

```
workspace_id
```

gooddata_sdk.compute.model.execution.ExecutionDefinition

```
class gooddata_sdk.compute.model.execution.ExecutionDefinition(attributes:
                                                                Optional[list[Attribute]], metrics:
                                                                Optional[list[Metric]], filters:
                                                                Optional[list[Filter]], dimensions:
                                                                list[Optional[list[str]]], totals:
                                                                Optional[list[TotalDefinition]] =
                                                                None)
```

Bases: object

```
__init__(attributes: Optional[list[Attribute]], metrics: Optional[list[Metric]], filters: Optional[list[Filter]],
         dimensions: list[Optional[list[str]]], totals: Optional[list[TotalDefinition]] = None) → None
```

Methods

`__init__(attributes, metrics, filters, ...)`

`as_api_model()`

`has_attributes()`

`has_filters()`

`has_metrics()`

`is_one_dim()`

`is_two_dim()`

Attributes

`attributes`

`dimensions`

`filters`

`metrics`

`gooddata_sdk.compute.model.execution.ExecutionResponse`

`gooddata_sdk.compute.model.execution.ExecutionResponse`

alias of *Execution*

`gooddata_sdk.compute.model.execution.ExecutionResult`

class `gooddata_sdk.compute.model.execution.ExecutionResult`(*result: ExecutionResult*)

Bases: `object`

__init__(*result: ExecutionResult*)

Methods

`__init__(result)`

`check_size_limits(result_size_limits)`

`get_all_header_values(dim, header_idx)`

`get_all_headers(dim)`

`is_complete([dim])`

`next_page_start([dim])`

Attributes

`data`

`grand_totals`

`headers`

`paging`

`paging_count`

`paging_offset`

`paging_total`

`gooddata_sdk.compute.model.execution.TotalDefinition`

```
class gooddata_sdk.compute.model.execution.TotalDefinition(local_id: str, aggregation: str,  
                                                         metric_local_id: str, total_dims:  
                                                         list[TotalDimension])
```

Bases: `object`

```
__init__(local_id: str, aggregation: str, metric_local_id: str, total_dims: list[TotalDimension]) → None
```

Method generated by attrs for class `TotalDefinition`.

Methods

<code>__init__(local_id, aggregation, ...)</code>	Method generated by attrs for class TotalDefinition.
---	--

Attributes

<code>local_id</code>	total's local identifier
<code>aggregation</code>	aggregation function; case insensitive; one of SUM, MIN, MAX, MED, AVG
<code>metric_local_id</code>	local identifier of the measure to calculate total for
<code>total_dims</code>	

aggregation: str

aggregation function; case insensitive; one of SUM, MIN, MAX, MED, AVG

local_id: str

total's local identifier

metric_local_id: str

local identifier of the measure to calculate total for

`gooddata_sdk.compute.model.execution.TotalDimension`

class `gooddata_sdk.compute.model.execution.TotalDimension`(*idx: int, items: list[str] = NOTHING*)

Bases: `object`

__init__(*idx: int, items: list[str] = NOTHING*) → `None`

Method generated by attrs for class TotalDimension.

Methods

<code>__init__(idx[, items])</code>	Method generated by attrs for class TotalDimension.
-------------------------------------	---

Attributes

<code>idx</code>	index of dimension in which to calculate the total
<code>items</code>	items to use during total calculation

idx: int

index of dimension in which to calculate the total

items: list[str]

items to use during total calculation

Exceptions

ResultSizeLimitsExceeded(result_size_limits, ...)

gooddata_sdk.compute.model.execution.ResultSizeLimitsExceeded

exception gooddata_sdk.compute.model.execution.ResultSizeLimitsExceeded(*result_size_limits*:
Tuple[Optional[int],
...],
actual_result_size:
Tuple[Optional[int],
...],
first_violating_index:
int)

gooddata_sdk.compute.model.filter

Classes

AbsoluteDateFilter(dataset, from_date, to_date)

AllTimeFilter() Filter that is semantically equivalent to absent filter.

AttributeFilter(label[, values])

MetricValueFilter(metric, operator, values)

NegativeAttributeFilter(label[, values])

PositiveAttributeFilter(label[, values])

RankingFilter(metrics, operator, value, ...)

RelativeDateFilter(dataset, granularity, ...)

gooddata_sdk.compute.model.filter.AbsoluteDateFilter

class gooddata_sdk.compute.model.filter.AbsoluteDateFilter(*dataset*: [ObjId](#), *from_date*: str, *to_date*:
str)

Bases: [Filter](#)

__init__(*dataset*: [ObjId](#), *from_date*: str, *to_date*: str) → None

Methods

`__init__(dataset, from_date, to_date)`

`as_api_model()`

`is_noop()`

Attributes

`apply_on_result`

`dataset`

`from_date`

`to_date`

`gooddata_sdk.compute.model.filter.AllTimeFilter`

class `gooddata_sdk.compute.model.filter.AllTimeFilter`

Bases: `Filter`

Filter that is semantically equivalent to absent filter.

This filter exists because ‘All time filter’ retrieved from GoodData.CN is non-standard as it does not have *from* and *to* fields; this is also the reason why `as_api_model` method is not implemented - it would lead to invalid object.

The main feature of this filter is noop.

`__init__()` → None

Methods

`__init__()`

`as_api_model()`

`is_noop()`

Attributes

`apply_on_result`

`gooddata_sdk.compute.model.filter.AttributeFilter`

class `gooddata_sdk.compute.model.filter.AttributeFilter`(*label: Union[ObjId, str, Attribute], values: list[str] = None*)

Bases: `Filter`

__init__(*label: Union[ObjId, str, Attribute], values: list[str] = None*) → None

Methods

`__init__`(*label[, values]*)

`as_api_model`()

`is_noop`()

Attributes

`apply_on_result`

`label`

`values`

`gooddata_sdk.compute.model.filter.MetricValueFilter`

class `gooddata_sdk.compute.model.filter.MetricValueFilter`(*metric: Union[ObjId, str, Metric], operator: str, values: Union[float, int, tuple[float, float]], treat_nulls_as: Union[float, None] = None*)

Bases: `Filter`

__init__(*metric: Union[ObjId, str, Metric], operator: str, values: Union[float, int, tuple[float, float]], treat_nulls_as: Union[float, None] = None*) → None

Methods

`__init__(metric, operator, values[, ...])`

`as_api_model()`

`is_noop()`

Attributes

`apply_on_result`

`metric`

`operator`

`treat_nulls_as`

`values`

`gooddata_sdk.compute.model.filter.NegativeAttributeFilter`

```
class gooddata_sdk.compute.model.filter.NegativeAttributeFilter(label: Union[ObjId, str,
                                                                    Attribute], values: list[str] =
                                                                    None)
```

Bases: `AttributeFilter`

```
__init__(label: Union[ObjId, str, Attribute], values: list[str] = None) → None
```

Methods

`__init__(label[, values])`

`as_api_model()`

`is_noop()`

Attributes

apply_on_result

label

values

gooddata_sdk.compute.model.filter.PositiveAttributeFilter

```
class gooddata_sdk.compute.model.filter.PositiveAttributeFilter(label: Union[ObjId, str,
                                                                    Attribute], values: list[str] =
                                                                    None)
```

Bases: *AttributeFilter*

```
__init__(label: Union[ObjId, str, Attribute], values: list[str] = None) → None
```

Methods

```
__init__(label[, values])
```

```
as_api_model()
```

```
is_noop()
```

Attributes

apply_on_result

label

values

gooddata_sdk.compute.model.filter.RankingFilter

```
class gooddata_sdk.compute.model.filter.RankingFilter(metrics: list[Union[ObjId, Metric, str]],
                                                       operator: str, value: int, dimensionality:
                                                       Optional[list[Union[str, ObjId, Attribute,
                                                       Metric]]])
```

Bases: *Filter*

```
__init__(metrics: list[Union[ObjId, Metric, str]], operator: str, value: int, dimensionality:
         Optional[list[Union[str, ObjId, Attribute, Metric]]]) → None
```

Methods

`__init__(metrics, operator, value, ...)`

`as_api_model()`

`is_noop()`

Attributes

`apply_on_result`

`dimensionality`

`metrics`

`operator`

`value`

gooddata_sdk.compute.model.filter.RelativeDateFilter

class gooddata_sdk.compute.model.filter.RelativeDateFilter(*dataset: [ObjId](#), granularity: str, from_shift: int, to_shift: int*)

Bases: [Filter](#)

__init__(*dataset: [ObjId](#), granularity: str, from_shift: int, to_shift: int*) → None

Methods

`__init__(dataset, granularity, from_shift, ...)`

`as_api_model()`

`is_noop()`

Attributes

`apply_on_result`

`dataset`

`from_shift`

`granularity`

`to_shift`

`gooddata_sdk.compute.model.metric`

Classes

`ArithmeticMetric`(local_id, operator, operands)

`Metric`(local_id)

`PopDate`(attribute, periods_ago)

`PopDateDataset`(dataset, periods_ago)

`PopDateMetric`(local_id, metric, date_attributes)

`PopDateSetMetric`(local_id, metric, date_datasets)

`SimpleMetric`(local_id, item[, aggregation, ...])

`gooddata_sdk.compute.model.metric.ArithmeticMetric`

```
class gooddata_sdk.compute.model.metric.ArithmeticMetric(local_id: str, operator: str, operands:
                                                         list[Union[str, Metric]])
```

Bases: `Metric`

```
__init__(local_id: str, operator: str, operands: list[Union[str, Metric]]) → None
```


Methods

`__init__(local_id, operator, operands)`

`as_api_model()`

Attributes

`local_id`

`operand_local_ids`

`operator`

gooddata_sdk.compute.model.metric.Metric
class gooddata_sdk.compute.model.metric.**Metric**(*local_id: str*)

 Bases: [ExecModelEntity](#)
`__init__(local_id: str) → None`
Methods

`__init__(local_id)`

`as_api_model()`

Attributes

`local_id`

gooddata_sdk.compute.model.metric.PopDate
class gooddata_sdk.compute.model.metric.**PopDate**(*attribute: Union[ObjId, Attribute], periods_ago: int*)

 Bases: `object`
`__init__(attribute: Union[ObjId, Attribute], periods_ago: int) → None`

Methods

`__init__(attribute, periods_ago)`

`as_api_model()`

Attributes

`attribute`

`periods_ago`

`gooddata_sdk.compute.model.metric.PopDateDataset`

class `gooddata_sdk.compute.model.metric.PopDateDataset`(*dataset: Union[ObjId, str], periods_ago: int*)

Bases: `object`

`__init__(dataset: Union[ObjId, str], periods_ago: int) → None`

Methods

`__init__(dataset, periods_ago)`

`as_api_model()`

Attributes

`dataset`

`periods_ago`

`gooddata_sdk.compute.model.metric.PopDateMetric`

class `gooddata_sdk.compute.model.metric.PopDateMetric`(*local_id: str, metric: Union[str, Metric], date_attributes: list[PopDate]*)

Bases: `Metric`

`__init__(local_id: str, metric: Union[str, Metric], date_attributes: list[PopDate]) → None`

Methods

`__init__(local_id, metric, date_attributes)`

`as_api_model()`

Attributes

`date_attributes`

`local_id`

`metric_local_id`

`gooddata_sdk.compute.model.metric.PopDatasetMetric`

class `gooddata_sdk.compute.model.metric.PopDatasetMetric`(*local_id: str, metric: Union[str, Metric], date_datasets: list[PopDateDataset]*)

Bases: `Metric`

`__init__(local_id: str, metric: Union[str, Metric], date_datasets: list[PopDateDataset])` → None

Methods

`__init__(local_id, metric, date_datasets)`

`as_api_model()`

Attributes

`date_datasets`

`local_id`

`metric_local_id`

gooddata_sdk.compute.model.metric.SimpleMetric

```
class gooddata_sdk.compute.model.metric.SimpleMetric(local_id: str, item: ObjId, aggregation: Optional[str] = None, compute_ratio: bool = False, filters: list[Filter] = None)
```

Bases: [Metric](#)

```
__init__(local_id: str, item: ObjId, aggregation: Optional[str] = None, compute_ratio: bool = False, filters: list[Filter] = None) → None
```

Methods

```
__init__(local_id, item[, aggregation, ...])
```

```
as_api_model()
```

Attributes

```
aggregation
```

```
compute_ratio
```

```
filters
```

```
item
```

```
local_id
```

gooddata_sdk.compute.service**Classes**

ComputeService (api_client)	Compute service drives computation of analytics for a GoodData.CN workspaces.
---	---

gooddata_sdk.compute.service.ComputeService

```
class gooddata_sdk.compute.service.ComputeService(api_client: GoodDataApiClient)
```

Bases: `object`

Compute service drives computation of analytics for a GoodData.CN workspaces. The prescription of what to compute is encapsulated by the ExecutionDefinition which consists of attributes, metrics, filters and definition of dimensions that influence how to organize the data in the result.

```
__init__(api_client: GoodDataApiClient)
```

Methods

<code>__init__(api_client)</code>	
<code>for_exec_def(workspace_id, exec_def)</code>	Starts computation in GoodData.CN workspace, using the provided execution definition.
<code>get_exec_metadata(workspace_id, result_id)</code>	Gets execution result's metadata from GoodData.CN workspace for given execution result ID.

for_exec_def(workspace_id: str, exec_def: ExecutionDefinition) → Execution

Starts computation in GoodData.CN workspace, using the provided execution definition.

Parameters

- **workspace_id** – workspace identifier
- **exec_def** – execution definition - this prescribes what to calculate, how to place labels and metric values into dimensions

get_exec_metadata(workspace_id: str, result_id: str) → ResultCacheMetadata

Gets execution result's metadata from GoodData.CN workspace for given execution result ID.

Parameters

- **workspace_id** – workspace identifier
- **result_id** – execution result ID

Returns

execution result's metadata

4.2.4 gooddata_sdk.insight

Classes

<code>Insight(from_vis_obj[, side_loads])</code>	
<code>InsightAttribute(attribute)</code>	
<code>InsightBucket(bucket)</code>	
<code>InsightFilter(f)</code>	
<code>InsightMetric(metric)</code>	Represents metric placed on an insight.
<code>InsightService(api_client)</code>	Insight Service allows retrieval of insights from a GD.CN workspace.

gooddata_sdk.insight.Insight

```
class gooddata_sdk.insight.Insight(from_vis_obj: dict[str, Any], side_loads: Optional[SideLoads] = None)
```

Bases: object

```
__init__(from_vis_obj: dict[str, Any], side_loads: Optional[SideLoads] = None) → None
```

Methods

```
__init__(from_vis_obj[, side_loads])
```

```
get_metadata(id_obj)
```

Attributes

```
are_relations_valid
```

```
attributes
```

```
buckets
```

```
description
```

```
filters
```

```
id
```

```
metrics
```

```
properties
```

```
side_loads
```

```
sorts
```

```
title
```

```
vis_url
```

gooddata_sdk.insight.InsightAttribute**class** gooddata_sdk.insight.InsightAttribute(*attribute: dict[str, Any]*)

Bases: object

__init__(*attribute: dict[str, Any]*) → None**Methods****__init__**(attribute)

as_computable()

Attributes

alias

label

label_id

local_id

gooddata_sdk.insight.InsightBucket**class** gooddata_sdk.insight.InsightBucket(*bucket: dict[str, Any]*)

Bases: object

__init__(*bucket: dict[str, Any]*) → None**Methods****__init__**(bucket)**Attributes**

attributes

items

local_id

metrics

gooddata_sdk.insight.InsightFilter**class** gooddata_sdk.insight.InsightFilter(*f: dict[str, Any]*)

Bases: object

__init__(*f: dict[str, Any]*) → None**Methods**

__init__(*f*)

as_computable()

gooddata_sdk.insight.InsightMetric**class** gooddata_sdk.insight.InsightMetric(*metric: dict[str, Any]*)

Bases: object

Represents metric placed on an insight.

Note: this has different shape than object passed to execution.

__init__(*metric: dict[str, Any]*) → None**Methods**

__init__(*metric*)

as_computable()

Attributes

alias

format

is_time_comparison

item

item_id

local_id

time_comparison_master

If this is a time comparison metric, return local_id of the master metric from which it is derived.

title

property time_comparison_master: Optional[str]

If this is a time comparison metric, return local_id of the master metric from which it is derived.

Returns

local_id of master metric, None if not a time comparison metric

gooddata_sdk.insight.InsightService

class gooddata_sdk.insight.InsightService(*api_client*: GoodDataApiClient)

Bases: object

Insight Service allows retrieval of insights from a GD.CN workspace. The insights are returned as instances of Insight which allows convenient introspection and necessary functions to convert the insight into a form where it can be sent for computation.

Note: the insights are created using GD.CN Analytical Designer or using GoodData.UI SDK. They are stored as visualization objects with a free-form body. This body is specific for AD & SDK. The Insight wrapper exists to take care of these discrepancies.

__init__(*api_client*: GoodDataApiClient) → None

Methods

__init__(*api_client*)

get_insight (<i>workspace_id</i> , <i>insight_id</i>)	Gets a single insight from a workspace.
--	---

get_insights (<i>workspace_id</i>)	Gets all insights for a workspace.
---	------------------------------------

get_insight(*workspace_id*: str, *insight_id*: str) → *Insight*

Gets a single insight from a workspace.

Parameters

- **workspace_id** – identifier of workspace to load insight from
- **insight_id** – identifier of the insight

Returns

single insight; the insight will contain sideloaded metadata about the entities it references

Return type

Insight

get_insights(*workspace_id*: str) → list[*Insight*]

Gets all insights for a workspace. The insights will contain side loaded metadata for all execution entities that they reference.

Parameters

workspace_id – identifier of workspace to load insights from

Returns

all available insights, each insight will contain side loaded metadata about the entities it references

4.2.5 gooddata_sdk.sdk

Classes

<code>GoodDataSdk(client)</code>	Top-level class that wraps all the functionality together.
----------------------------------	--

gooddata_sdk.sdk.GoodDataSdk

class gooddata_sdk.sdk.GoodDataSdk(*client*: GoodDataApiClient)

Bases: object

Top-level class that wraps all the functionality together.

__init__(*client*: GoodDataApiClient) → None

Take instance of GoodDataApiClient and return new GoodDataSdk instance.

Useful when customized GoodDataApiClient is needed. Usually users should use *GoodDataSdk.create* classmethod.

Methods

<code>__init__(client)</code>	Take instance of GoodDataApiClient and return new GoodDataSdk instance.
<code>create(host_, token_[, extra_user_agent_])</code>	Create common GoodDataApiClient and return new GoodDataSdk instance.

Attributes

<code>catalog_data_source</code>
<code>catalog_organization</code>
<code>catalog_permission</code>
<code>catalog_user</code>
<code>catalog_workspace</code>
<code>catalog_workspace_content</code>
<code>client</code>
<code>compute</code>
<code>insights</code>
<code>support</code>
<code>tables</code>

```
classmethod create(host_: str, token_: str, extra_user_agent_: Optional[str] = None,
                    **custom_headers_: Optional[str]) → GoodDataSdk
```

Create common GoodDataApiClient and return new GoodDataSdk instance. Custom headers are filtered. Headers with None value are removed. It simplifies usage because headers can be created directly from optional values.

This is preferred way of creating GoodDataSdk, when no tweaks are needed.

4.2.6 gooddata_sdk.support

Classes

```
SupportService(api_client)
```

gooddata_sdk.support.SupportService

```
class gooddata_sdk.support.SupportService(api_client: GoodDataApiClient)
```

Bases: object

```
__init__(api_client: GoodDataApiClient) → None
```

Methods

```
__init__(api_client)
```

```
wait_till_available(timeout[, sleep_time])      Wait till GD.CN service is available.
```

Attributes

```
is_available      Checks if GD.CN is available.
```

```
property is_available: bool
```

Checks if GD.CN is available. Can raise exceptions in case of authentication or authorization failure.

Returns

True - available, False - not available

```
wait_till_available(timeout: int, sleep_time: float = 2.0) → None
```

Wait till GD.CN service is available. When timeout is:

- > 0 exception is raised after given number of seconds.
- = 0 exception is raised whe service is not available immediately
- < 0 no timeout

Method propagates is_available exceptions.

Parameters

- **timeout** – seconds to wait to service to be available (see method description for details)
- **sleep_time** – seconds to wait between GD.CN availability tests

4.2.7 gooddata_sdk.table

Classes

<code>ExecutionTable(response, first_page)</code>	Represents execution result as a table.
<code>TableService(api_client)</code>	The TableService provides a convenient way to drive computations and access the results in a tabular fashion.

gooddata_sdk.table.ExecutionTable

class gooddata_sdk.table.**ExecutionTable**(response: [Execution](#), first_page: [ExecutionResult](#))

Bases: object

Represents execution result as a table. This is a convenience wrapper for executions constructed using the following convention:

- all attributes are in the first dimension
- all metrics are in the second dimension
- if the execution is attribute- or metric-less, then there is always single dimension

The mapping to rows is then as follows:

- both attributes + metrics are on the execution = iteration over first dimension; as many rows as total records in the first dimension (paging.total[0])
- just attributes = iteration over just headers in first dimension; as many rows as total records in the first dimension (paging.total[0])
- just metrics = single row, all metrics values returned in one row

__init__(response: [Execution](#), first_page: [ExecutionResult](#)) → None

Methods

<code>__init__(response, first_page)</code>	
<code>read_all()</code>	Returns a generator that will be yielding execution result as rows.

Attributes

<hr/>	
attributes	
<hr/>	
<code>column_ids</code>	Returns column identifiers.
<code>column_metadata</code>	Returns mapping of column identifier to definition of either attribute whose elements will be in that column or metric whose value will be calculated in that column.
<hr/>	
metrics	
<hr/>	

property `column_ids`: `list[str]`

Returns column identifiers. Each row will be a mapping of column identifier to column data.

property `column_metadata`: `dict[str, Union[Attribute, Metric]]`

Returns mapping of column identifier to definition of either attribute whose elements will be in that column or metric whose value will be calculated in that column.

read_all() → `Generator[dict[str, Any], None, None]`

Returns a generator that will be yielding execution result as rows. Each row is a dict() mapping column identifier to value of that column.

Returns

generator yielding dict() representing rows of the table

gooddata_sdk.table.TableService

class `gooddata_sdk.table.TableService`(*api_client*: `GoodDataApiClient`)

Bases: `object`

The TableService provides a convenient way to drive computations and access the results in a tabular fashion.

Compared to the ComputeService, with this one here you do not have to worry about the layout of the result and do not have to have to work with execution response, access the data using paging.

The ExecutionTable returned by the TableService allows you to iterate over the rows of the calculated data.

__init__(*api_client*: `GoodDataApiClient`) → `None`

Methods

<hr/>	
<code>__init__(api_client)</code>	
<hr/>	
<code>for_insight(workspace_id, insight)</code>	
<hr/>	
<code>for_items(workspace_id, items[, filters])</code>	
<hr/>	

4.2.8 gooddata_sdk.type_converter

Functions

<i>build_stores()</i>	Initialize both AttributeConverterStore and DBTypeConverterStore with Convertors.
-----------------------	---

gooddata_sdk.type_converter.build_stores

gooddata_sdk.type_converter.build_stores() → None

Initialize both AttributeConverterStore and DBTypeConverterStore with Convertors.

Classes

<i>AttributeConverterStore()</i>	Store for conversion of attributes
<i>Converter()</i>	Base Converter class.
<i>ConverterRegistryStore()</i>	Class store TypeConverterRegistry instances for each registered type.
<i>DBTypeConverterStore()</i>	Store for conversion of database types
<i>DateConverter()</i>	
<i>DatetimeConverter()</i>	
<i>IntegerConverter()</i>	
<i>StringConverter()</i>	
<i>TypeConverterRegistry(type_name)</i>	Class stores converters for given type with ability to distinguish converters based on sub-type granularity.

gooddata_sdk.type_converter.AttributeConverterStore

class gooddata_sdk.type_converter.AttributeConverterStore

Bases: *ConverterRegistryStore*

Store for conversion of attributes

__init__()

Methods

__init__()	
<i>find_converter(type_name[, sub_type])</i>	Find Converter for given type and sub type.
<i>register(type_name, class_converter[, sub_types])</i>	Register Converter instance created from provided Converter class to given type and list of sub types.
<i>reset()</i>	Reset converters setup

classmethod `find_converter`(*type_name: str, sub_type: Optional[str] = None*) → *Converter*

Find Converter for given type and sub type.

Parameters

- **type_name** – type name
- **sub_type** – sub type name

classmethod `register`(*type_name: str, class_converter: Type[Converter], sub_types: Optional[list[str]] = None*) → None

Register Converter instance created from provided Converter class to given type and list of sub types. When sub types are not provided, converter is registered as the default one for given type.

Parameters

- **type_name** – type name
- **class_converter** – Converter class
- **sub_types** – list of sub types or None (default type Converter)

classmethod `reset`() → None

Reset converters setup

`gooddata_sdk.type_converter.Converter`

class `gooddata_sdk.type_converter.Converter`

Bases: object

Base Converter class. It defines Converter API and implements support for external type conversion. External type conversion provides ability to plug-in conversion function to Converter

`__init__()`

Methods

`__init__()`

`db_data_type()`

`set_external_fnc(fnc)`

`to_external_type(value)`

`to_type(value)`

Attributes

DEFAULT_DB_DATA_TYPE

gooddata_sdk.type_converter.ConverterRegistryStore

class gooddata_sdk.type_converter.ConverterRegistryStore

Bases: object

Class store TypeConverterRegistry instances for each registered type. It provides interface to register converters with type and sub-type and to find converter. The class is not meant to be used directly but as base class for child classes

__init__()

Methods

__init__()

find_converter (type_name[, sub_type])	Find Converter for given type and sub type.
register (type_name, class_converter[, sub_types])	Register Converter instance created from provided Converter class to given type and list of sub types.
reset ()	Reset converters setup

classmethod find_converter(type_name: str, sub_type: Optional[str] = None) → Converter

Find Converter for given type and sub type.

Parameters

- **type_name** – type name
- **sub_type** – sub type name

classmethod register(type_name: str, class_converter: Type[Converter], sub_types: Optional[list[str]] = None) → None

Register Converter instance created from provided Converter class to given type and list of sub types. When sub types are not provided, converter is registered as the default one for given type.

Parameters

- **type_name** – type name
- **class_converter** – Converter class
- **sub_types** – list of sub types or None (default type Converter)

classmethod reset() → None

Reset converters setup

gooddata_sdk.type_converter.DBTypeConverterStore**class** gooddata_sdk.type_converter.DBTypeConverterStoreBases: [ConverterRegistryStore](#)

Store for conversion of database types

__init__()**Methods**

__init__ ()	
find_converter (type_name[, sub_type])	Find Converter for given type and sub type.
register (type_name, class_converter[, sub_types])	Register Converter instance created from provided Converter class to given type and list of sub types.
reset ()	Reset converters setup

classmethod **find_converter**(type_name: str, sub_type: Optional[str] = None) → [Converter](#)

Find Converter for given type and sub type.

Parameters

- **type_name** – type name
- **sub_type** – sub type name

classmethod **register**(type_name: str, class_converter: Type[[Converter](#)], sub_types: Optional[list[str]] = None) → None

Register Converter instance created from provided Converter class to given type and list of sub types. When sub types are not provided, converter is registered as the default one for given type.

Parameters

- **type_name** – type name
- **class_converter** – Converter class
- **sub_types** – list of sub types or None (default type Converter)

classmethod **reset**() → None

Reset converters setup

gooddata_sdk.type_converter.DateConverter**class** gooddata_sdk.type_converter.DateConverterBases: [Converter](#)**__init__**()

Methods

<code>__init__()</code>	
<code>db_data_type()</code>	
<code>set_external_fnc(fnc)</code>	
<code>to_date(value)</code>	Add first month and first date to incomplete iso date string.
<code>to_external_type(value)</code>	
<code>to_type(value)</code>	

Attributes

<code>DEFAULT_DB_DATA_TYPE</code>

classmethod `to_date(value: str) → date`

Add first month and first date to incomplete iso date string.

```
>>> assert DateConverter.to_date("2021-01") == date(2021, 1, 1)
>>> assert DateConverter.to_date("1992") == date(1992, 1, 1)
```

`gooddata_sdk.type_converter.DatetimeConverter`

class `gooddata_sdk.type_converter.DatetimeConverter`

Bases: `Converter`

`__init__()`

Methods

<code>__init__()</code>	
<code>db_data_type()</code>	
<code>set_external_fnc(fnc)</code>	
<code>to_datetime(value)</code>	Append minutes to incomplete datetime string.
<code>to_external_type(value)</code>	
<code>to_type(value)</code>	

Attributes

DEFAULT_DB_DATA_TYPE

classmethod `to_datetime(value: str) → datetime`

Append minutes to incomplete datetime string.

```
>>> from datetime import datetime
>>> assert DatetimeConverter.to_datetime("2021-01-01 02") == datetime(2021, 1, 1, 2, 0)
>>> assert DatetimeConverter.to_datetime("2021-01-01 12:34") == datetime(2021, 1, 1, 12, 34)
```

`gooddata_sdk.type_converter.IntegerConverter`

class `gooddata_sdk.type_converter.IntegerConverter`

Bases: `Converter`

`__init__()`

Methods

`__init__()`

`db_data_type()`

`set_external_fnc(fnc)`

`to_external_type(value)`

`to_type(value)`

Attributes

DEFAULT_DB_DATA_TYPE

gooddata_sdk.type_converter.StringConverter**class** gooddata_sdk.type_converter.StringConverterBases: *Converter***__init__**()**Methods**

__init__()

db_data_type()

set_external_fnc(fnc)

to_external_type(value)

to_type(value)

Attributes

DEFAULT_DB_DATA_TYPE

gooddata_sdk.type_converter.TypeConverterRegistry**class** gooddata_sdk.type_converter.TypeConverterRegistry(*type_name: str*)

Bases: object

Class stores converters for given type with ability to distinguish converters based on sub-type granularity.

__init__(*type_name: str*)

Initialize instance with type for which instance is going to be responsible

Parameters**type_name** – type name**Methods**

__init__(type_name)

Initialize instance with type for which instance is going to be responsible

converter(sub_type)

Find and return converter instance for a given sub-type.

register(converter, sub_type)Register converter instance for given sub-type (granularity).

converter(*sub_type*: *Optional[str]*) → *Converter*

Find and return converter instance for a given sub-type. Default converter instance is returned if the sub-type is not found or not provided. When a default converter is not registered, `ValueError` exception is raised.

Parameters

sub_type – sub-type name

Returns

Converter instance

register(*converter*: *Converter*, *sub_type*: *Optional[str]*) → *None*

Register converter instance for given sub-type (granularity). If sub-type is not specified, converter is registered as the default one for the whole type. Default converter can be registered only once.

Parameters

- **converter** – converter instance
- **sub_type** – sub-type name

4.2.9 gooddata_sdk.utils

Functions

camel_to_snake(*camel_case_str*)

change_case(*dictionary*, *case*)

change_case_helper(*value*, *case*)

create_directory(*path*)

get_sorted_yaml_files(*folder*)

<i>id_obj_to_key</i> (<i>id_obj</i>)	Given an object containing an id+type pair, this function will return a string key.
--	---

<i>load_all_entities</i> (<i>get_page_func</i> [, <i>page_size</i>])	Loads all entities from a paged resource.
--	---

load_all_entities_dict(*get_page_func*[, ...])

read_layout_from_file(*path*)

snake_to_camel(*snake_case_str*)

write_layout_to_file(*path*, *content*)

`gooddata_sdk.utils.camel_to_snake`

`gooddata_sdk.utils.camel_to_snake(camel_case_str: str) → str`

`gooddata_sdk.utils.change_case`

`gooddata_sdk.utils.change_case(dictionary: dict, case: Callable[[str], str]) → dict`

`gooddata_sdk.utils.change_case_helper`

`gooddata_sdk.utils.change_case_helper(value: Union[list, dict, str], case: Callable[[str], str]) → Union[list, dict, str]`

`gooddata_sdk.utils.create_directory`

`gooddata_sdk.utils.create_directory(path: Path) → None`

`gooddata_sdk.utils.get_sorted_yaml_files`

`gooddata_sdk.utils.get_sorted_yaml_files(folder: Path) → list[Path]`

`gooddata_sdk.utils.id_obj_to_key`

`gooddata_sdk.utils.id_obj_to_key(id_obj: Union[str, ObjId, Dict[str, Dict[str, str]], Dict[str, str]]) → str`

Given an object containing an id+type pair, this function will return a string key.

For convenience, this also recognizes the *ref* format used by GoodData.UI SDK. In that format, the id+type are wrapped in ‘identifier’.

Parameters

id_obj – id object

Returns

string that can be used as key

`gooddata_sdk.utils.load_all_entities`

`gooddata_sdk.utils.load_all_entities(get_page_func: functools.partial[Any], page_size: int = 500) → AllPagedEntities`

Loads all entities from a paged resource. The primary input to this function is a partial function that is setup with all the fixed parameters. Given this the function will get entities page-by-page and merge them into a single ‘pseudo-response’ containing data and included attributes.

An example usage:

```
>>> import functools
>>> import gooddata_metadata_client as metadata_client
>>> import gooddata_metadata_client.apis as metadata_apis
>>> api = metadata_apis.EntitiesApi(metadata_client.ApiClient())
```

(continues on next page)

(continued from previous page)

```
>>> get_func = functools.partial(api.get_all_entities_visualization_objects, 'some-
↳workspace-id',
>>>                                     include=["ALL"], _check_return_type=False)
>>> vis_objects = load_all_entities(get_func)
```

Parameters

- **get_page_func** – an API controller from the metadata client
- **page_size** – optionally specify page length, default is 500

gooddata_sdk.utils.load_all_entities_dict

```
gooddata_sdk.utils.load_all_entities_dict(get_page_func: functools.partial[Any], page_size: int = 500,
                                          camel_case: bool = False) → dict[str, Any]
```

gooddata_sdk.utils.read_layout_from_file

```
gooddata_sdk.utils.read_layout_from_file(path: Path) → Any
```

gooddata_sdk.utils.snake_to_camel

```
gooddata_sdk.utils.snake_to_camel(snake_case_str: str) → str
```

gooddata_sdk.utils.write_layout_to_file

```
gooddata_sdk.utils.write_layout_to_file(path: Path, content: Union[dict[str, Any], list[dict]]) → None
```

Classes

```
AllPagedEntities(data, included)
```

```
IndentDumper(stream[, default_style, ...])
```

```
SideLoads(objs)
```

gooddata_sdk.utils.AllPagedEntities

```
class gooddata_sdk.utils.AllPagedEntities(data, included)
```

```
    Bases: tuple
```

```
    __init__()
```

Methods

<code>__init__()</code>	
<code>count(value, /)</code>	Return number of occurrences of value.
<code>index(value[, start, stop])</code>	Return first index of value.

Attributes

<code>data</code>	Alias for field number 0
<code>included</code>	Alias for field number 1

count(*value*, /)

Return number of occurrences of value.

property data

Alias for field number 0

property included

Alias for field number 1

index(*value*, *start*=0, *stop*=9223372036854775807, /)

Return first index of value.

Raises ValueError if the value is not present.

gooddata_sdk.utils.IndentDumper

```
class gooddata_sdk.utils.IndentDumper(stream, default_style=None, default_flow_style=False,
                                       canonical=None, indent=None, width=None,
                                       allow_unicode=None, line_break=None, encoding=None,
                                       explicit_start=None, explicit_end=None, version=None,
                                       tags=None, sort_keys=True)
```

Bases: SafeDumper

```
__init__(stream, default_style=None, default_flow_style=False, canonical=None, indent=None,
         width=None, allow_unicode=None, line_break=None, encoding=None, explicit_start=None,
         explicit_end=None, version=None, tags=None, sort_keys=True)
```

Methods

<code>__init__(stream[, default_style, ...])</code>
<code>add_implicit_resolver(tag, regexp, first)</code>
<code>add_multi_representer(data_type, representer)</code>
<code>add_path_resolver(tag, path[, kind])</code>

continues on next page

Table 1 – continued from previous page

<code>add_representer(data_type, representer)</code>
<code>analyze_scalar(scalar)</code>
<code>anchor_node(node)</code>
<code>ascend_resolver()</code>
<code>check_empty_document()</code>
<code>check_empty_mapping()</code>
<code>check_empty_sequence()</code>
<code>check_resolver_prefix(depth, path, kind, ...)</code>
<code>check_simple_key()</code>
<code>choose_scalar_style()</code>
<code>close()</code>
<code>descend_resolver(current_node, current_index)</code>
<code>determine_block_hints(text)</code>
<code>dispose()</code>
<code>emit(event)</code>
<code>expect_alias()</code>
<code>expect_block_mapping()</code>
<code>expect_block_mapping_key([first])</code>
<code>expect_block_mapping_simple_value()</code>
<code>expect_block_mapping_value()</code>
<code>expect_block_sequence()</code>
<code>expect_block_sequence_item([first])</code>
<code>expect_document_end()</code>
<code>expect_document_root()</code>
<code>expect_document_start([first])</code>

continues on next page

Table 1 – continued from previous page

<code>expect_first_block_mapping_key()</code>
<code>expect_first_block_sequence_item()</code>
<code>expect_first_document_start()</code>
<code>expect_first_flow_mapping_key()</code>
<code>expect_first_flow_sequence_item()</code>
<code>expect_flow_mapping()</code>
<code>expect_flow_mapping_key()</code>
<code>expect_flow_mapping_simple_value()</code>
<code>expect_flow_mapping_value()</code>
<code>expect_flow_sequence()</code>
<code>expect_flow_sequence_item()</code>
<code>expect_node([root, sequence, mapping, ...])</code>
<code>expect_nothing()</code>
<code>expect_scalar()</code>
<code>expect_stream_start()</code>
<code>flush_stream()</code>
<code>generate_anchor(node)</code>
<code>ignore_aliases(data)</code>
<code>increase_indent([flow, indentless])</code>
<code>need_events(count)</code>
<code>need_more_events()</code>
<code>open()</code>
<code>prepare_anchor(anchor)</code>
<code>prepare_tag(tag)</code>
<code>prepare_tag_handle(handle)</code>

continues on next page

Table 1 – continued from previous page

<code>prepare_tag_prefix(prefix)</code>
<code>prepare_version(version)</code>
<code>process_anchor(indicator)</code>
<code>process_scalar()</code>
<code>process_tag()</code>
<code>represent(data)</code>
<code>represent_binary(data)</code>
<code>represent_bool(data)</code>
<code>represent_data(data)</code>
<code>represent_date(data)</code>
<code>represent_datetime(data)</code>
<code>represent_dict(data)</code>
<code>represent_float(data)</code>
<code>represent_int(data)</code>
<code>represent_list(data)</code>
<code>represent_mapping(tag, mapping[, flow_style])</code>
<code>represent_none(data)</code>
<code>represent_scalar(tag, value[, style])</code>
<code>represent_sequence(tag, sequence[, flow_style])</code>
<code>represent_set(data)</code>
<code>represent_str(data)</code>
<code>represent_undefined(data)</code>
<code>represent_yaml_object(tag, data, cls[, ...])</code>
<code>resolve(kind, value, implicit)</code>
<code>serialize(node)</code>

continues on next page

Table 1 – continued from previous page

<code>serialize_node(node, parent, index)</code>
<code>write_double_quoted(text[, split])</code>
<code>write_folded(text)</code>
<code>write_indent()</code>
<code>write_indicator(indicator, need_whitespace)</code>
<code>write_line_break([data])</code>
<code>write_literal(text)</code>
<code>write_plain(text[, split])</code>
<code>write_single_quoted(text[, split])</code>
<code>write_stream_end()</code>
<code>write_stream_start()</code>
<code>write_tag_directive(handle_text, prefix_text)</code>
<code>write_version_directive(version_text)</code>

Attributes

<code>ANCHOR_TEMPLATE</code>
<code>DEFAULT_MAPPING_TAG</code>
<code>DEFAULT_SCALAR_TAG</code>
<code>DEFAULT_SEQUENCE_TAG</code>
<code>DEFAULT_TAG_PREFIXES</code>
<code>ESCAPE_REPLACEMENTS</code>
<code>inf_value</code>
<code>yaml_implicit_resolvers</code>
<code>yaml_multi_representers</code>
<code>yaml_path_resolvers</code>
<code>yaml_representers</code>

gooddata_sdk.utils.SideLoads**class** gooddata_sdk.utils.**SideLoads**(*objs: list[Any]*)

Bases: object

__init__(*objs: list[Any]*) → None**Methods**

__init__(objs)

all_for_type(obj_type)

find(id_obj)

PYTHON MODULE INDEX

g

[gooddata_fdw](#), 13
[gooddata_fdw.column_utils](#), 14
[gooddata_fdw.column_validation](#), 14
[gooddata_fdw.environment](#), 16
[gooddata_fdw.executor](#), 18
[gooddata_fdw.fdw](#), 22
[gooddata_fdw.filter](#), 23
[gooddata_fdw.import_workspace](#), 23
[gooddata_fdw.naming](#), 26
[gooddata_fdw.options](#), 29
[gooddata_fdw.pg_logging](#), 31
[gooddata_fdw.result_reader](#), 31
[gooddata_sdk](#), 32
[gooddata_sdk.catalog](#), 33
[gooddata_sdk.catalog.base](#), 33
[gooddata_sdk.catalog.catalog_service_base](#), 35
[gooddata_sdk.catalog.data_source](#), 35
[gooddata_sdk.catalog.data_source.action_requests](#), 36
[gooddata_sdk.catalog.data_source.action_requests.idm_request](#), 36
[gooddata_sdk.catalog.data_source.action_requests.scan_model_request](#), 39
[gooddata_sdk.catalog.data_source.declarative_model](#), 41
[gooddata_sdk.catalog.data_source.declarative_model.data_source](#), 42
[gooddata_sdk.catalog.data_source.declarative_model.physical_model](#), 45
[gooddata_sdk.catalog.data_source.declarative_model.physical_model.column](#), 46
[gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm](#), 47
[gooddata_sdk.catalog.data_source.declarative_model.physical_model.table](#), 50
[gooddata_sdk.catalog.data_source.entity_model](#), 52
[gooddata_sdk.catalog.data_source.entity_model.content_objects](#), 52
[gooddata_sdk.catalog.data_source.entity_model.content_objects.table](#), 52
[gooddata_sdk.catalog.data_source.entity_model.data_source](#), 57
[gooddata_sdk.catalog.data_source.service](#), 71
[gooddata_sdk.catalog.data_source.validation](#), 74
[gooddata_sdk.catalog.data_source.validation.data_source](#), 74
[gooddata_sdk.catalog.entity](#), 75
[gooddata_sdk.catalog.identifier](#), 79
[gooddata_sdk.catalog.organization](#), 85
[gooddata_sdk.catalog.organization.entity_model](#), 85
[gooddata_sdk.catalog.organization.entity_model.organization](#), 85
[gooddata_sdk.catalog.organization.service](#), 89
[gooddata_sdk.catalog.permission](#), 90
[gooddata_sdk.catalog.permission.declarative_model](#), 90
[gooddata_sdk.catalog.permission.declarative_model.permission](#), 90
[gooddata_sdk.catalog.permission.service](#), 95
[gooddata_sdk.catalog.setting](#), 96
[gooddata_sdk.catalog.types](#), 97
[gooddata_sdk.catalog.user](#), 97
[gooddata_sdk.catalog.user.declarative_model](#), 97
[gooddata_sdk.catalog.user.declarative_model.user](#), 97
[gooddata_sdk.catalog.user.declarative_model.user_and_user_group](#), 99
[gooddata_sdk.catalog.user.declarative_model.user_group](#), 100
[gooddata_sdk.catalog.user.entity_model](#), 103
[gooddata_sdk.catalog.user.entity_model.user](#), 103
[gooddata_sdk.catalog.user.entity_model.user_group](#), 108
[gooddata_sdk.catalog.user.service](#), 112
[gooddata_sdk.catalog.workspace](#), 115
[gooddata_sdk.catalog.workspace.content_service](#), 115
[gooddata_sdk.catalog.workspace.declarative_model](#), 115

117
gooddata_sdk.catalog.workspace.declarative_model.workspace,
117
gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model,
118
gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model,
118
gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model,
130
gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset,
131
gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset,
131
gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.date_dataset,
141
gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.date_dataset.date_dataset,
141
gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.ldm,
145
gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace,
147
gooddata_sdk.catalog.workspace.entity_model,
156
gooddata_sdk.catalog.workspace.entity_model.content_objects,
157
gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset,
157
gooddata_sdk.catalog.workspace.entity_model.content_objects.metric,
161
gooddata_sdk.catalog.workspace.entity_model.graph_objects,
162
gooddata_sdk.catalog.workspace.entity_model.graph_objects.graph,
162
gooddata_sdk.catalog.workspace.entity_model.workspace,
167
gooddata_sdk.catalog.workspace.model_container,
168
gooddata_sdk.catalog.workspace.service, 170
gooddata_sdk.client, 172
gooddata_sdk.compute, 173
gooddata_sdk.compute.model, 173
gooddata_sdk.compute.model.attribute, 174
gooddata_sdk.compute.model.base, 174
gooddata_sdk.compute.model.execution, 176
gooddata_sdk.compute.model.filter, 182
gooddata_sdk.compute.model.metric, 188
gooddata_sdk.compute.service, 192
gooddata_sdk.insight, 193
gooddata_sdk.sdk, 198
gooddata_sdk.support, 199
gooddata_sdk.table, 200
gooddata_sdk.type_converter, 202
gooddata_sdk.utils, 209

INDEX

Symbols

[__init__\(\) \(gooddata_fdw.column_validation.ColumnValidator method\), 15](#)
[__init__\(\) \(gooddata_fdw.column_validation.IdOptionValidator method\), 15](#)
[__init__\(\) \(gooddata_fdw.column_validation.LocalIdOptionValidator method\), 15](#)
[__init__\(\) \(gooddata_fdw.environment.ColumnDefinitionStub method\), 17](#)
[__init__\(\) \(gooddata_fdw.environment.ForeignDataWrapperStub method\), 17](#)
[__init__\(\) \(gooddata_fdw.environment.QualStub method\), 18](#)
[__init__\(\) \(gooddata_fdw.environment.TableDefinitionStub method\), 18](#)
[__init__\(\) \(gooddata_fdw.executor.ComputeExecutor method\), 19](#)
[__init__\(\) \(gooddata_fdw.executor.CustomExecutor method\), 19](#)
[__init__\(\) \(gooddata_fdw.executor.Executor method\), 19](#)
[__init__\(\) \(gooddata_fdw.executor.ExecutorFactory method\), 20](#)
[__init__\(\) \(gooddata_fdw.executor.InitData method\), 20](#)
[__init__\(\) \(gooddata_fdw.executor.InsightExecutor method\), 21](#)
[__init__\(\) \(gooddata_fdw.fdw.GoodDataForeignDataWrapper method\), 22](#)
[__init__\(\) \(gooddata_fdw.import_workspace.ImporterInitData method\), 24](#)
[__init__\(\) \(gooddata_fdw.import_workspace.InsightsWorkspaceImporter method\), 25](#)
[__init__\(\) \(gooddata_fdw.import_workspace.SemanticLayerWorkspaceImporter method\), 25](#)
[__init__\(\) \(gooddata_fdw.import_workspace.WorkspaceImporter method\), 25](#)
[__init__\(\) \(gooddata_fdw.import_workspace.WorkspaceImporterLocator method\), 26](#)
[__init__\(\) \(gooddata_fdw.naming.CatalogNamingStrategy method\), 27](#)
[__init__\(\) \(gooddata_fdw.naming.DefaultCatalogNamingStrategy method\), 27](#)
[__init__\(\) \(gooddata_fdw.naming.DefaultInsightColumnNaming method\), 27](#)
[__init__\(\) \(gooddata_fdw.naming.DefaultInsightTableNameNaming method\), 28](#)
[__init__\(\) \(gooddata_fdw.naming.InsightColumnNamingStrategy method\), 28](#)
[__init__\(\) \(gooddata_fdw.naming.InsightTableNameNamingStrategy method\), 29](#)
[__init__\(\) \(gooddata_fdw.options.BaseOptions method\), 29](#)
[__init__\(\) \(gooddata_fdw.options.ImportSchemaOptions method\), 30](#)
[__init__\(\) \(gooddata_fdw.options.ServerOptions method\), 30](#)
[__init__\(\) \(gooddata_fdw.options.TableOptions method\), 31](#)
[__init__\(\) \(gooddata_fdw.result_reader.InsightTableResultReader method\), 31](#)
[__init__\(\) \(gooddata_fdw.result_reader.TableResultReader method\), 32](#)
[__init__\(\) \(gooddata_sdk.catalog.base.Base method\), 34](#)
[__init__\(\) \(gooddata_sdk.catalog.catalog_service_base.CatalogService method\), 35](#)
[__init__\(\) \(gooddata_sdk.catalog.data_source.action_requests.ldm_request method\), 38](#)
[__init__\(\) \(gooddata_sdk.catalog.data_source.action_requests.scan_model method\), 40](#)
[__init__\(\) \(gooddata_sdk.catalog.data_source.declarative_model.data_source method\), 43](#)
[__init__\(\) \(gooddata_sdk.catalog.data_source.declarative_model.data_source method\), 44](#)
[__init__\(\) \(gooddata_sdk.catalog.data_source.declarative_model.physical_model method\), 46](#)
[__init__\(\) \(gooddata_sdk.catalog.data_source.declarative_model.physical_model method\), 48](#)
[__init__\(\) \(gooddata_sdk.catalog.data_source.declarative_model.physical_model method\), 49](#)
[__init__\(\) \(gooddata_sdk.catalog.data_source.declarative_model.physical_model method\), 51](#)
[__init__\(\) \(gooddata_sdk.catalog.data_source.entity_model.content_object method\), 51](#)

Index	223
--------------	------------

method), 184
__init__() (gooddata_sdk.compute.model.filter.NegativeAttributeFilter method), 206
method), 185
__init__() (gooddata_sdk.compute.model.filter.PositiveAttributeFilter method), 207
method), 186
__init__() (gooddata_sdk.compute.model.filter.RankingFilter method), 208
method), 186
__init__() (gooddata_sdk.compute.model.filter.RelativeDateFilter method), 208
method), 187
__init__() (gooddata_sdk.compute.model.metric.ArithmeticMetric method), 211
method), 188
__init__() (gooddata_sdk.compute.model.metric.Metric method), 212
method), 189
__init__() (gooddata_sdk.compute.model.metric.PopDate method), 217
method), 189
__init__() (gooddata_sdk.compute.model.metric.PopDateDataset method), 190
__init__() (gooddata_sdk.compute.model.metric.PopDateMetric data_sdk.compute.model.filter), 182
method), 190
__init__() (gooddata_sdk.compute.model.metric.PopDateSetMetric attribute), 181
method), 191
__init__() (gooddata_sdk.compute.model.metric.SimpleMetric data_sdk.compute.model.filter), 183
method), 192
__init__() (gooddata_sdk.compute.service.ComputeService method), 192
__init__() (gooddata_sdk.insight.Insight method), 194
__init__() (gooddata_sdk.insight.InsightAttribute method), 195
__init__() (gooddata_sdk.insight.InsightBucket method), 195
__init__() (gooddata_sdk.insight.InsightFilter method), 196
__init__() (gooddata_sdk.insight.InsightMetric method), 196
__init__() (gooddata_sdk.insight.InsightService method), 197
__init__() (gooddata_sdk.sdk.GoodDataSdk method), 198
__init__() (gooddata_sdk.support.SupportService method), 199
__init__() (gooddata_sdk.table.ExecutionTable method), 200
__init__() (gooddata_sdk.table.TableService method), 201
__init__() (gooddata_sdk.type_converter.AttributeConverterStore method), 202
__init__() (gooddata_sdk.type_converter.Converter method), 203
__init__() (gooddata_sdk.type_converter.ConverterRegistryStore method), 204
__init__() (gooddata_sdk.type_converter.DBTypeConverterStore method), 205
__init__() (gooddata_sdk.type_converter.DateConverter method), 205
__init__() (gooddata_sdk.type_converter.DatetimeConverter method), 206
__init__() (gooddata_sdk.type_converter.IntegerConverter method), 207
__init__() (gooddata_sdk.type_converter.StringConverter method), 208
__init__() (gooddata_sdk.type_converter.TypeConverterRegistry method), 208
__init__() (gooddata_sdk.utils.AllPagedEntities method), 211
__init__() (gooddata_sdk.utils.IndentDumper method), 212
__init__() (gooddata_sdk.utils.SideLoads method), 217

A

AggregateDateFilter (class in gooddata_sdk.compute.model.filter), 182
aggregation (gooddata_sdk.compute.model.execution.TotalDefinition attribute), 181
AllPagedEntities (class in gooddata_sdk.utils), 211
AbsoluteDateFilter (class in gooddata_sdk.compute.model.filter), 183
ArithmeticMetric (class in gooddata_sdk.compute.model.metric), 188
Attribute (class in gooddata_sdk.compute.model.attribute), 174
AttributeConverterStore (class in gooddata_sdk.type_converter), 202
AttributeFilter (class in gooddata_sdk.compute.model.filter), 184

B

BareExecutionResponse (class in gooddata_sdk.compute.model.execution), 177
Base (class in gooddata_sdk.catalog.base), 34
BaseOptions (class in gooddata_sdk.options), 29
BasicCredentials (class in gooddata_sdk.catalog.entity), 75
BigQueryAttributes (class in gooddata_sdk.catalog.data_source.entity_model.data_source), 57
build_stores() (in module gooddata_sdk.type_converter), 202

C

camel_to_snake() (in module gooddata_sdk.utils), 210
catalog_with_valid_objects() (gooddata_sdk.catalog.workspace.model_container.CatalogWorkspace method), 169
CatalogAnalyticsBase (class in gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics), 118

CatalogAssigneeIdentifier	(class in good-data_sdk.catalog.identifier), 80	data_sdk.catalog.data_source.declarative_model.physical_model	46
CatalogAttribute	(class in good-data_sdk.catalog.workspace.entity_model.content_objects.dataset), 157	CatalogDeclarativeDashboardPlugin	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model), 125
CatalogDataset	(class in good-data_sdk.catalog.workspace.entity_model.content_objects.dataset), 158	CatalogDeclarativeDataset	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset), 135
CatalogDataSource	(class in good-data_sdk.catalog.data_source.entity_model.data_source), 58	CatalogDeclarativeDataSource	(class in good-data_sdk.catalog.data_source.declarative_model.data_source), 42
CatalogDataSourceBigQuery	(class in good-data_sdk.catalog.data_source.entity_model.data_source), 59	CatalogDeclarativeDataSourcePermission	(class in good-data_sdk.catalog.permission.declarative_model.permission), 91
CatalogDataSourcePostgres	(class in good-data_sdk.catalog.data_source.entity_model.data_source), 61	CatalogDeclarativeDataSources	(class in good-data_sdk.catalog.data_source.declarative_model.data_source), 44
CatalogDataSourceRedshift	(class in good-data_sdk.catalog.data_source.entity_model.data_source), 63	CatalogDeclarativeDateDataset	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset), 141
CatalogDataSourceService	(class in good-data_sdk.catalog.data_source.service), 72	CatalogDeclarativeFact	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset), 137
CatalogDataSourceSnowflake	(class in good-data_sdk.catalog.data_source.entity_model.data_source), 65	CatalogDeclarativeFilterContext	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model), 126
CatalogDataSourceTable	(class in good-data_sdk.catalog.data_source.entity_model.content_objects.table), 52	CatalogDeclarativeLabel	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset), 138
CatalogDataSourceTableAttributes	(class in good-data_sdk.catalog.data_source.entity_model.content_objects.table), 54	CatalogDeclarativeLdm	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset), 145
CatalogDataSourceTableColumn	(class in good-data_sdk.catalog.data_source.entity_model.content_objects.table), 55	CatalogDeclarativeMetric	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model), 127
CatalogDataSourceTableIdentifier	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset), 131	CatalogDeclarativeModel	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset), 146
CatalogDataSourceVertica	(class in good-data_sdk.catalog.data_source.entity_model.data_source), 67	CatalogDeclarativeReference	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset), 120
CatalogDeclarativeAnalyticalDashboard	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model), 120	CatalogDeclarativeSetting	(class in good-data_sdk.catalog.setting), 96
CatalogDeclarativeAnalytics	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model), 121	CatalogDeclarativeSingleWorkspacePermission	(class in good-data_sdk.catalog.permission.declarative_model.permission), 91
CatalogDeclarativeAnalyticsLayer	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.analytics_model.analytics_model), 123	CatalogDeclarativeTable	(class in good-data_sdk.catalog.data_source.declarative_model.physical_model.physical_model), 158
CatalogDeclarativeAttribute	(class in good-data_sdk.catalog.workspace.declarative_model.workspace.logical_model.dataset.dataset), 132	CatalogDeclarativeTables	(class in good-data_sdk.catalog.data_source.declarative_model.physical_model.physical_model), 158
CatalogDeclarativeColumn	(class in good-		

48	CatalogDeclarativeUser (class in good- data_sdk.catalog.user.declarative_model.user), 97	data_sdk.catalog.workspace.entity_model.graph_objects.graph), 163
CatalogDeclarativeUserGroup (class in good- data_sdk.catalog.user.declarative_model.user_group), 101	CatalogDependentEntitiesRequest (class in good- data_sdk.catalog.workspace.entity_model.graph_objects.graph), 164	CatalogDependentEntitiesResponse (class in good- data_sdk.catalog.workspace.entity_model.graph_objects.graph), 165
CatalogDeclarativeUserGroups (class in good- data_sdk.catalog.user.declarative_model.user_group), 102	CatalogEntity (class in good- data_sdk.catalog.entity), 76	CatalogEntityIdentifier (class in good- data_sdk.catalog.workspace.entity_model.graph_objects.graph), 166
CatalogDeclarativeUsers (class in good- data_sdk.catalog.user.declarative_model.user), 98	CatalogFact (class in good- data_sdk.catalog.workspace.entity_model.content_objects.database), 36	CatalogGenerateLdmRequest (class in good- data_sdk.catalog.data_source.action_requests.ldm_request), 80
CatalogDeclarativeUsersUserGroups (class in good- data_sdk.catalog.user.declarative_model.user_and_user_group), 99	CatalogGranularityIdentifier (class in good- data_sdk.catalog.identifier), 144	CatalogGranularitiesFormatting (class in good- data_sdk.catalog.workspace.declarative_model.workspace.logical_model), 166
CatalogDeclarativeVisualizationObject (class in good- data_sdk.catalog.workspace.declarative_model.workspace.declarative_model), 129	CatalogLabel (class in good- data_sdk.catalog.workspace.entity_model.content_objects.database), 150	CatalogLabelIdentifier (class in good- data_sdk.catalog.identifier), 81
CatalogDeclarativeWorkspace (class in good- data_sdk.catalog.workspace.declarative_model.workspace.declarative_model), 148	CatalogMetric (class in good- data_sdk.catalog.workspace.entity_model.content_objects.metric), 161	CatalogNameEntity (class in good- data_sdk.catalog.entity), 76
CatalogDeclarativeWorkspaceDataFilter (class in good- data_sdk.catalog.workspace.declarative_model.workspace.declarative_model), 150	CatalogNamingStrategy (class in good- data_sdk.fdw.naming), 27	CatalogOrganization (class in good- data_sdk.catalog.organization.entity_model.organization), 85
CatalogDeclarativeWorkspaceDataFilters (class in good- data_sdk.catalog.workspace.declarative_model.workspace.declarative_model), 153	CatalogOrganizationAttributes (class in good- data_sdk.catalog.organization.entity_model.organization), 86	CatalogOrganizationDocument (class in good- data_sdk.catalog.organization.entity_model.organization), 88
CatalogDeclarativeWorkspaceDataFilterSetting (class in good- data_sdk.catalog.workspace.declarative_model.workspace.declarative_model), 152	CatalogOrganizationService (class in good- data_sdk.catalog.organization.service), 89	CatalogPermissionService (class in good- data_sdk.catalog.permission.service), 95
CatalogDeclarativeWorkspaceHierarchyPermission (class in good- data_sdk.catalog.permission.declarative_model.permission), 93	CatalogReferenceIdentifier (class in good- data_sdk.catalog.identifier), 82	CatalogScanModelRequest (class in good- data_sdk.catalog.data_source.action_requests.scan_model_request), 40
CatalogDeclarativeWorkspaceModel (class in good- data_sdk.catalog.workspace.declarative_model.workspace.declarative_model), 154		
CatalogDeclarativeWorkspacePermissions (class in good- data_sdk.catalog.permission.declarative_model.permission), 94		
CatalogDeclarativeWorkspaces (class in good- data_sdk.catalog.workspace.declarative_model.workspace.declarative_model), 155		
CatalogDependentEntitiesGraph (class in good- data_sdk.catalog.workspace.entity_model.graph_objects.graph), 162		
CatalogDependentEntitiesNode (class in good- data_sdk.catalog.workspace.entity_model.graph_objects.graph), 162		

CatalogScanResultPdm (class in good-
 data_sdk.catalog.data_source.declarative_model.
 49
 CatalogServiceBase (class in good-
 data_sdk.catalog.catalog_service_base),
 35
 CatalogTitleEntity (class in good-
 data_sdk.catalog.entity), 77
 CatalogTypeEntity (class in good-
 data_sdk.catalog.entity), 77
 CatalogUser (class in good-
 data_sdk.catalog.user.entity_model.user),
 103
 CatalogUserAttributes (class in good-
 data_sdk.catalog.user.entity_model.user),
 104
 CatalogUserDocument (class in good-
 data_sdk.catalog.user.entity_model.user),
 105
 CatalogUserGroup (class in good-
 data_sdk.catalog.user.entity_model.user_group),
 109
 CatalogUserGroupDocument (class in good-
 data_sdk.catalog.user.entity_model.user_group),
 110
 CatalogUserGroupIdentifier (class in good-
 data_sdk.catalog.identifier), 83
 CatalogUserGroupParents (class in good-
 data_sdk.catalog.user.entity_model.user_group),
 111
 CatalogUserGroupRelationships (class in good-
 data_sdk.catalog.user.entity_model.user_group),
 111
 CatalogUserGroupsData (class in good-
 data_sdk.catalog.user.entity_model.user),
 106
 CatalogUserRelationships (class in good-
 data_sdk.catalog.user.entity_model.user),
 107
 CatalogUserService (class in good-
 data_sdk.catalog.user.service), 113
 CatalogWorkspace (class in good-
 data_sdk.catalog.workspace.entity_model.workspace),
 167
 CatalogWorkspaceContent (class in good-
 data_sdk.catalog.workspace.model_container),
 168
 CatalogWorkspaceContentService (class in good-
 data_sdk.catalog.workspace.content_service),
 115
 CatalogWorkspaceIdentifier (class in good-
 data_sdk.catalog.identifier), 84
 CatalogWorkspaceService (class in good-
 data_sdk.catalog.workspace.service), 170
 change_case() (in module gooddata_sdk.utils), 210
 change_case_helper() (in module good-
 data_sdk.utils), 210
 column_data_type_for() (in module good-
 data_fdw.column_utils), 14
 column_ids (gooddata_sdk.table.ExecutionTable prop-
 erty), 201
 column_metadata (gooddata_sdk.table.ExecutionTable
 property), 201
 ColumnDefinition (in module good-
 data_fdw.environment), 16
 ColumnDefinitionStub (class in good-
 data_fdw.environment), 17
 columns (gooddata_fdw.executor.InitData property), 21
 ColumnValidator (class in good-
 data_fdw.column_validation), 15
 compute_model_to_api_model() (in module good-
 data_sdk.compute.model.execution), 176
 compute_valid_objects() (good-
 data_sdk.catalog.workspace.content_service.CatalogWorkspaceC
 method), 117
 ComputeExecutor (class in gooddata_fdw.executor), 19
 ComputeService (class in good-
 data_sdk.compute.service), 192
 Converter (class in gooddata_sdk.type_converter), 203
 converter() (gooddata_sdk.type_converter.TypeConverterRegistry
 method), 208
 ConverterRegistryStore (class in good-
 data_sdk.type_converter), 204
 count() (gooddata_fdw.executor.InitData method), 21
 count() (gooddata_fdw.import_workspace.ImporterInitData
 method), 24
 count() (gooddata_sdk.utils.AllPagedEntities method),
 212
 create() (gooddata_sdk.sdk.GoodDataSdk class
 method), 199
 create_directory() (in module gooddata_sdk.utils),
 210
 Credentials (class in gooddata_sdk.catalog.entity), 77
 CustomExecutor (class in gooddata_fdw.executor), 19
D
 data (gooddata_sdk.utils.AllPagedEntities property),
 212
 DatabaseAttributes (class in good-
 data_sdk.catalog.data_source.entity_model.data_source),
 69
 DataSourceValidator (class in good-
 data_sdk.catalog.data_source.validation.data_source),
 74
 DateConverter (class in gooddata_sdk.type_converter),
 205
 DatetimeConverter (class in good-
 data_sdk.type_converter), 206

DBTypeConverterStore (class in gooddata_sdk.type_converter), 205

DefaultCatalogNamingStrategy (class in gooddata_sdk.fdw.naming), 27

DefaultInsightColumnNaming (class in gooddata_sdk.fdw.naming), 27

DefaultInsightTableNameNaming (class in gooddata_sdk.fdw.naming), 28

delete_workspace() (gooddata_sdk.catalog.workspace.service.CatalogWorkspaceService class method), 172

E

ExecModelEntity (class in gooddata_sdk.compute.model.base), 175

Execution (class in gooddata_sdk.compute.model.execution), 178

ExecutionDefinition (class in gooddata_sdk.compute.model.execution), 178

ExecutionResponse (in module gooddata_sdk.compute.model.execution), 179

ExecutionResult (class in gooddata_sdk.compute.model.execution), 179

ExecutionTable (class in gooddata_sdk.table), 200

Executor (class in gooddata_sdk.fdw.executor), 19

ExecutorFactory (class in gooddata_sdk.fdw.executor), 20

extract_filters_from_qual() (in module gooddata_sdk.fdw.filter), 23

F

Filter (class in gooddata_sdk.compute.model.base), 175

filter_dataset() (gooddata_sdk.catalog.workspace.entity_model.content_objects.dataset.CatalogDataset class method), 159

find_converter() (gooddata_sdk.type_converter.AttributeConverterStore class method), 202

find_converter() (gooddata_sdk.type_converter.ConverterRegistryStore class method), 204

find_converter() (gooddata_sdk.type_converter.DBTypeConverterStore class method), 205

find_label_attribute() (gooddata_sdk.catalog.workspace.model_container.CatalogWorkspaceContent class method), 169

for_exec_def() (gooddata_sdk.compute.service.ComputeService class method), 193

ForeignDataWrapper (in module gooddata_sdk.fdw.environment), 17

ForeignDataWrapperStub (class in gooddata_sdk.fdw.environment), 17

from_api() (gooddata_sdk.catalog.base.Base class method), 34

from_api() (gooddata_sdk.catalog.data_source.action_requests.ldm_request class method), 39

from_api() (gooddata_sdk.catalog.data_source.action_requests.scan_model class method), 41

from_api() (gooddata_sdk.catalog.data_source.declarative_model.data_source class method), 44

from_api() (gooddata_sdk.catalog.data_source.declarative_model.data_source class method), 45

from_api() (gooddata_sdk.catalog.data_source.declarative_model.physical_model class method), 47

from_api() (gooddata_sdk.catalog.data_source.declarative_model.physical_model class method), 49

from_api() (gooddata_sdk.catalog.data_source.declarative_model.physical_model class method), 50

from_api() (gooddata_sdk.catalog.data_source.declarative_model.physical_model class method), 51

from_api() (gooddata_sdk.catalog.data_source.entity_model.content_object class method), 53

from_api() (gooddata_sdk.catalog.data_source.entity_model.content_object class method), 54

from_api() (gooddata_sdk.catalog.data_source.entity_model.content_object class method), 56

from_api() (gooddata_sdk.catalog.identifier.CatalogAssigneeIdentifier class method), 80

from_api() (gooddata_sdk.catalog.identifier.CatalogGrainIdentifier class method), 81

from_api() (gooddata_sdk.catalog.identifier.CatalogLabelIdentifier class method), 82

from_api() (gooddata_sdk.catalog.identifier.CatalogReferenceIdentifier class method), 83

from_api() (gooddata_sdk.catalog.identifier.CatalogUserGroupIdentifier class method), 84

from_api() (gooddata_sdk.catalog.identifier.CatalogWorkspaceIdentifier class method), 84

from_api() (gooddata_sdk.catalog.organization.entity_model.organization class method), 86

from_api() (gooddata_sdk.catalog.organization.entity_model.organization class method), 88

from_api() (gooddata_sdk.catalog.organization.entity_model.organization class method), 89

from_api() (gooddata_sdk.catalog.permission.declarative_model.permission class method), 91

from_api() (gooddata_sdk.catalog.permission.declarative_model.permission class method), 92

from_api() (gooddata_sdk.catalog.permission.declarative_model.permission class method), 93

from_api() (gooddata_sdk.catalog.permission.declarative_model.permission class method), 94

from_api() (gooddata_sdk.catalog.setting.CatalogDeclarativeSetting class method), 96

from_api() (gooddata_sdk.catalog.user.declarative_model.user.CatalogUser class method), 98

229

`from_dict()` (`gooddata_sdk.catalog.workspace.entity_model` class method), 165
`from_dict()` (`gooddata_sdk.catalog.workspace.entity_model` class method), 166
`from_dict()` (`gooddata_sdk.catalog.workspace.entity_model` class method), 167
G
`get_dataset()` (`gooddata_sdk.catalog.workspace.model_container.CatalogWorkspaceContent` method), 169
`get_exec_metadata()` (`gooddata_sdk.compute.service.ComputeService` method), 193
`get_full_catalog()` (`gooddata_sdk.catalog.workspace.content_service.CatalogWorkspaceContentService` method), 117
`get_insight()` (`gooddata_sdk.insight.InsightService` method), 197
`get_insights()` (`gooddata_sdk.insight.InsightService` method), 197
`get_metric()` (`gooddata_sdk.catalog.workspace.model_container.CatalogWorkspaceContent` method), 169
`get_pdm_folder()` (in module `gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm`), 48
`get_sorted_yaml_files()` (in module `gooddata_sdk.utils`), 210
`get_workspace()` (`gooddata_sdk.catalog.workspace.service.CatalogWorkspaceService` method), 172
`get_workspace_folder()` (in module `gooddata_sdk.catalog.workspace.declarative_model.workspace.workspace`), 147
`gooddata_fdw` module, 13
`gooddata_fdw.column_utils` module, 14
`gooddata_fdw.column_validation` module, 14
`gooddata_fdw.environment` module, 16
`gooddata_fdw.executor` module, 18
`gooddata_fdw.fdw` module, 22
`gooddata_fdw.filter` module, 23
`gooddata_fdw.import_workspace` module, 23
`gooddata_fdw.naming` module, 26
`gooddata_fdw.options` module, 29
`gooddata_sdk.catalog` module, 33
`gooddata_sdk.catalog.base` module, 32
`gooddata_sdk.catalog.catalog_service_base` module, 35
`gooddata_sdk.catalog.data_source` module, 35
`gooddata_sdk.catalog.data_source.action_requests` module, 36
`gooddata_sdk.catalog.data_source.action_requests.ldm_request` module, 36
`gooddata_sdk.catalog.data_source.action_requests.scan_model` module, 39
`gooddata_sdk.catalog.data_source.declarative_model` module, 41
`gooddata_sdk.catalog.data_source.declarative_model.data_source` module, 42
`gooddata_sdk.catalog.data_source.declarative_model.physical_model` module, 45
`gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm` module, 46
`gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm.pdm` module, 47
`gooddata_sdk.catalog.data_source.declarative_model.physical_model.pdm.pdm.pdm` module, 50
`gooddata_sdk.catalog.data_source.entity_model` module, 52
`gooddata_sdk.catalog.data_source.entity_model.content_object` module, 52
`gooddata_sdk.catalog.data_source.entity_model.content_object` module, 52
`gooddata_sdk.catalog.data_source.entity_model.data_source` module, 57
`gooddata_sdk.catalog.data_source.service` module, 71
`gooddata_sdk.catalog.data_source.validation` module, 74
`gooddata_sdk.catalog.data_source.validation.data_source` module, 74
`gooddata_sdk.catalog.entity` module, 75
`gooddata_sdk.catalog.identifier` module, 79
`gooddata_sdk.catalog.organization` module, 85
`gooddata_sdk.catalog.organization.entity_model` module, 85

gooddata_sdk.catalog.organization.entity_model	gooddata_sdk.catalog.organization.entity_model
module, 85	module, 141
gooddata_sdk.catalog.organization.service	gooddata_sdk.catalog.workspace.declarative_model.workspace
module, 89	module, 145
gooddata_sdk.catalog.permission	gooddata_sdk.catalog.workspace.declarative_model.workspace
module, 90	module, 147
gooddata_sdk.catalog.permission.declarative_model	gooddata_sdk.catalog.workspace.entity_model
module, 90	module, 156
gooddata_sdk.catalog.permission.declarative_model.permission	gooddata_sdk.catalog.workspace.entity_model.content_object
module, 90	module, 157
gooddata_sdk.catalog.permission.service	gooddata_sdk.catalog.workspace.entity_model.content_object
module, 95	module, 157
gooddata_sdk.catalog.setting	gooddata_sdk.catalog.workspace.entity_model.content_object
module, 96	module, 161
gooddata_sdk.catalog.types	gooddata_sdk.catalog.workspace.entity_model.graph_objects
module, 97	module, 162
gooddata_sdk.catalog.user	gooddata_sdk.catalog.workspace.entity_model.graph_objects
module, 97	module, 162
gooddata_sdk.catalog.user.declarative_model	gooddata_sdk.catalog.workspace.entity_model.workspace
module, 97	module, 167
gooddata_sdk.catalog.user.declarative_model.user	gooddata_sdk.catalog.workspace.model_container
module, 97	module, 168
gooddata_sdk.catalog.user.declarative_model.user_group	gooddata_sdk.catalog.workspace.service
module, 99	module, 170
gooddata_sdk.catalog.user.declarative_model.user_group_group	gooddata_sdk.client
module, 100	module, 172
gooddata_sdk.catalog.user.entity_model	gooddata_sdk.compute
module, 103	module, 173
gooddata_sdk.catalog.user.entity_model.user	gooddata_sdk.compute.model
module, 103	module, 173
gooddata_sdk.catalog.user.entity_model.user_group	gooddata_sdk.compute.model.attribute
module, 108	module, 174
gooddata_sdk.catalog.user.service	gooddata_sdk.compute.model.base
module, 112	module, 174
gooddata_sdk.catalog.workspace	gooddata_sdk.compute.model.execution
module, 115	module, 176
gooddata_sdk.catalog.workspace.content_service	gooddata_sdk.compute.model.filter
module, 115	module, 182
gooddata_sdk.catalog.workspace.declarative_model	gooddata_sdk.compute.model.metric
module, 117	module, 188
gooddata_sdk.catalog.workspace.declarative_model.workspace	gooddata_sdk.compute.service
module, 117	module, 192
gooddata_sdk.catalog.workspace.declarative_model.insight	gooddata_sdk.insights_model
module, 118	module, 193
gooddata_sdk.catalog.workspace.declarative_model.snapshot	gooddata_sdk.analytics_model.analytics_model
module, 118	module, 198
gooddata_sdk.catalog.workspace.declarative_model.snapshot_model	gooddata_sdk.snapshot_model
module, 130	module, 199
gooddata_sdk.catalog.workspace.declarative_model.logical_model	gooddata_sdk.logical_model.dataset
module, 131	module, 200
gooddata_sdk.catalog.workspace.declarative_model.logical_model.type_converter	gooddata_sdk.logical_model.type_converter
module, 131	module, 202
gooddata_sdk.catalog.workspace.declarative_model.logical_model.date_dataset	gooddata_sdk.logical_model.date_dataset
module, 141	module, 209

- GoodDataApiClient (class in *gooddata_sdk.client*), 172
- GoodDataForeignDataWrapper (class in *gooddata_fdw.fdw*), 22
- GoodDataSdk (class in *gooddata_sdk.sdk*), 198
- I**
- id_obj_to_key() (in module *gooddata_sdk.utils*), 210
- IdOptionValidator (class in *gooddata_fdw.column_validation*), 15
- idx (*gooddata_sdk.compute.model.execution.TotalDimension* attribute), 181
- import_options (gooddata_fdw.import_workspace.ImporterInitData property), 24
- ImporterInitData (class in *gooddata_fdw.import_workspace*), 24
- ImportSchemaOptions (class in *gooddata_fdw.options*), 30
- included (*gooddata_sdk.utils.AllPagedEntities* property), 212
- IndentDumper (class in *gooddata_sdk.utils*), 212
- index() (*gooddata_fdw.executor.InitData* method), 21
- index() (*gooddata_fdw.import_workspace.ImporterInitData* method), 24
- index() (*gooddata_sdk.utils.AllPagedEntities* method), 212
- InitData (class in *gooddata_fdw.executor*), 20
- Insight (class in *gooddata_sdk.insight*), 194
- InsightAttribute (class in *gooddata_sdk.insight*), 195
- InsightBucket (class in *gooddata_sdk.insight*), 195
- InsightColumnNamingStrategy (class in *gooddata_fdw.naming*), 28
- InsightExecutor (class in *gooddata_fdw.executor*), 21
- InsightFilter (class in *gooddata_sdk.insight*), 196
- InsightMetric (class in *gooddata_sdk.insight*), 196
- InsightService (class in *gooddata_sdk.insight*), 197
- InsightsWorkspaceImporter (class in *gooddata_fdw.import_workspace*), 25
- InsightTableNameStrategy (class in *gooddata_fdw.naming*), 29
- InsightTableResultReader (class in *gooddata_fdw.result_reader*), 31
- IntegerConverter (class in *gooddata_sdk.type_converter*), 207
- is_available (*gooddata_sdk.support.SupportService* property), 199
- items (*gooddata_sdk.compute.model.execution.TotalDimension* attribute), 181
- L**
- load_all_entities() (in module *gooddata_sdk.utils*), 210
- load_all_entities_dict() (in module *gooddata_sdk.utils*), 211
- local_id (*gooddata_sdk.compute.model.execution.TotalDefinition* attribute), 181
- LocalIdOptionValidator (class in *gooddata_fdw.column_validation*), 15
- log_to_postgres() (in module *gooddata_fdw.environment*), 16
- M**
- Metric (class in *gooddata_sdk.compute.model.metric*), 189
- metric_local_id (gooddata_sdk.compute.model.execution.TotalDefinition attribute), 181
- MetricValueFilter (class in *gooddata_sdk.compute.model.filter*), 184
- module**
- gooddata_fdw, 13
- gooddata_fdw.column_utils, 14
- gooddata_fdw.column_validation, 14
- gooddata_fdw.environment, 16
- gooddata_fdw.executor, 18
- gooddata_fdw.fdw, 22
- gooddata_fdw.filter, 23
- gooddata_fdw.import_workspace, 23
- gooddata_fdw.naming, 26
- gooddata_fdw.options, 29
- gooddata_fdw.pg_logging, 31
- gooddata_fdw.result_reader, 31
- gooddata_sdk, 32
- gooddata_sdk.catalog, 33
- gooddata_sdk.catalog.base, 33
- gooddata_sdk.catalog.catalog_service_base, 35
- gooddata_sdk.catalog.data_source, 35
- gooddata_sdk.catalog.data_source.action_requests, 36
- gooddata_sdk.catalog.data_source.action_requests.ldm_r, 36
- gooddata_sdk.catalog.data_source.action_requests.scan, 39
- gooddata_sdk.catalog.data_source.declarative_model, 41
- gooddata_sdk.catalog.data_source.declarative_model.dat, 42
- gooddata_sdk.catalog.data_source.declarative_model.phy, 45
- gooddata_sdk.catalog.data_source.declarative_model.phy, 46
- gooddata_sdk.catalog.data_source.declarative_model.phy, 47
- gooddata_sdk.catalog.data_source.declarative_model.phy, 50
- gooddata_sdk.catalog.data_source.entity_model, 52

[gooddata_sdk.catalog.data_source.entity_model.content_objects,](#)
[52](#) [gooddata_sdk.catalog.workspace.declarative_model.works](#)
[gooddata_sdk.catalog.data_source.entity_model.content_objects.table,](#)
[52](#) [gooddata_sdk.catalog.workspace.declarative_model.works](#)
[gooddata_sdk.catalog.data_source.entity_model.data_source,](#)
[57](#) [gooddata_sdk.catalog.workspace.declarative_model.works](#)
[gooddata_sdk.catalog.data_source.service,](#) [131](#)
[gooddata_sdk.catalog.data_source.validation,](#) [131](#)
[74](#) [gooddata_sdk.catalog.workspace.declarative_model.works](#)
[gooddata_sdk.catalog.data_source.validation.data_source,](#)
[74](#) [gooddata_sdk.catalog.workspace.declarative_model.works](#)
[gooddata_sdk.catalog.entity,](#) [75](#) [141](#)
[gooddata_sdk.catalog.identifier,](#) [79](#) [gooddata_sdk.catalog.workspace.declarative_model.works](#)
[gooddata_sdk.catalog.organization,](#) [85](#) [145](#)
[gooddata_sdk.catalog.organization.entity_model,](#) [gooddata_sdk.catalog.workspace.declarative_model.works](#)
[85](#) [147](#)
[gooddata_sdk.catalog.organization.entity_model.organization,](#) [gooddata_sdk.catalog.workspace.declarative_model.works](#)
[85](#) [156](#)
[gooddata_sdk.catalog.organization.service,](#) [gooddata_sdk.catalog.workspace.entity_model,](#)
[89](#) [157](#)
[gooddata_sdk.catalog.permission,](#) [90](#) [gooddata_sdk.catalog.workspace.entity_model.content_ob](#)
[gooddata_sdk.catalog.permission.declarative_model,](#) [157](#)
[90](#) [gooddata_sdk.catalog.workspace.entity_model.content_ob](#)
[gooddata_sdk.catalog.permission.declarative_model.permission,](#)
[90](#) [gooddata_sdk.catalog.workspace.entity_model.graph_obje](#)
[gooddata_sdk.catalog.permission.service,](#) [162](#)
[95](#) [gooddata_sdk.catalog.workspace.entity_model.graph_obje](#)
[gooddata_sdk.catalog.setting,](#) [96](#) [162](#)
[gooddata_sdk.catalog.types,](#) [97](#) [gooddata_sdk.catalog.workspace.entity_model.workspace,](#)
[gooddata_sdk.catalog.user,](#) [97](#) [167](#)
[gooddata_sdk.catalog.user.declarative_model,](#) [gooddata_sdk.catalog.workspace.model_container,](#)
[97](#) [168](#)
[gooddata_sdk.catalog.user.declarative_model.user,](#) [gooddata_sdk.catalog.workspace.service,](#)
[97](#) [170](#)
[gooddata_sdk.catalog.user.declarative_model.user_group,](#) [gooddata_sdk.workspace,](#) [172](#)
[99](#) [gooddata_sdk.compute,](#) [173](#)
[gooddata_sdk.catalog.user.declarative_model.user_group,](#) [gooddata_sdk.compute.model,](#) [173](#)
[100](#) [gooddata_sdk.compute.model.attribute,](#) [174](#)
[gooddata_sdk.catalog.user.entity_model,](#) [gooddata_sdk.compute.model.base,](#) [174](#)
[103](#) [gooddata_sdk.compute.model.execution,](#) [176](#)
[gooddata_sdk.catalog.user.entity_model.user,](#) [gooddata_sdk.compute.model.filter,](#) [182](#)
[103](#) [gooddata_sdk.compute.model.metric,](#) [188](#)
[gooddata_sdk.catalog.user.entity_model.user_group,](#) [gooddata_sdk.compute.service,](#) [192](#)
[108](#) [gooddata_sdk.insight,](#) [193](#)
[gooddata_sdk.catalog.user.service,](#) [112](#) [gooddata_sdk.sdk,](#) [198](#)
[gooddata_sdk.catalog.workspace,](#) [115](#) [gooddata_sdk.support,](#) [199](#)
[gooddata_sdk.catalog.workspace.content_service,](#) [gooddata_sdk.table,](#) [200](#)
[115](#) [gooddata_sdk.type_converter,](#) [202](#)
[gooddata_sdk.catalog.workspace.declarative_model,](#) [gooddata_sdk.utils,](#) [209](#)
[117](#)
[gooddata_sdk.catalog.workspace.declarative_model.workspace,](#)
[117](#) [NegativeAttributeFilter \(class in good-](#)
[gooddata_sdk.catalog.workspace.declarative_model.workspace.analytics_model,](#) [185](#)
[gooddata_sdk.compute.model.filter,](#)

O

`ObjId` (class in `gooddata_sdk.compute.model.base`), 175
`one_scan_true()` (in module `gooddata_sdk.catalog.data_source.action_requests.scan_model_request`), 40

P

`PopDate` (class in `gooddata_sdk.compute.model.metric`), 189
`PopDateDataset` (class in `gooddata_sdk.compute.model.metric`), 190
`PopDateMetric` (class in `gooddata_sdk.compute.model.metric`), 190
`PopDatesetMetric` (class in `gooddata_sdk.compute.model.metric`), 191
`PositiveAttributeFilter` (class in `gooddata_sdk.compute.model.filter`), 186
`PostgresAttributes` (class in `gooddata_sdk.catalog.data_source.entity_model.data_source`), 69

Q

`Qual` (in module `gooddata_fdw.environment`), 17
`QualStub` (class in `gooddata_fdw.environment`), 18

R

`RankingFilter` (class in `gooddata_sdk.compute.model.filter`), 186
`read_all()` (`gooddata_sdk.table.ExecutionTable` method), 201
`read_layout_from_file()` (in module `gooddata_sdk.utils`), 211
`read_result()` (`gooddata_sdk.compute.model.execution.BareExecutionTable` method), 177
`RedshiftAttributes` (class in `gooddata_sdk.catalog.data_source.entity_model.data_source`), 70
`register()` (`gooddata_sdk.type_converter.AttributeConverterStore` class method), 203
`register()` (`gooddata_sdk.type_converter.ConverterRegistryStore` class method), 204
`register()` (`gooddata_sdk.type_converter.DBTypeConverterStore` class method), 205
`register()` (`gooddata_sdk.type_converter.TypeConverterRegistry` method), 209
`RelativeDateFilter` (class in `gooddata_sdk.compute.model.filter`), 187
`reset()` (`gooddata_sdk.type_converter.AttributeConverterStore` class method), 203
`reset()` (`gooddata_sdk.type_converter.ConverterRegistryStore` class method), 204
`reset()` (`gooddata_sdk.type_converter.DBTypeConverterStore` class method), 205

`restriction_type` (`gooddata_fdw.import_workspace.ImporterInitData` property), 24
`restricts` (`gooddata_fdw.import_workspace.ImporterInitData` property), 24
`ResultSizeLimitsExceeded`, 182

S

`sdk` (`gooddata_fdw.executor.InitData` property), 21
`sdk` (`gooddata_fdw.import_workspace.ImporterInitData` property), 24
`SemanticLayerWorkspaceImporter` (class in `gooddata_fdw.import_workspace`), 25
`server_options` (`gooddata_fdw.executor.InitData` property), 21
`server_options` (`gooddata_fdw.import_workspace.ImporterInitData` property), 24
`ServerOptions` (class in `gooddata_fdw.options`), 30
`SideLoads` (class in `gooddata_sdk.utils`), 217
`SimpleMetric` (class in `gooddata_sdk.compute.model.metric`), 192
`snake_to_camel()` (in module `gooddata_sdk.utils`), 211
`SnowflakeAttributes` (class in `gooddata_sdk.catalog.data_source.entity_model.data_source`), 70
`StringConverter` (class in `gooddata_sdk.type_converter`), 208
`SupportService` (class in `gooddata_sdk.support`), 199

T

`table_col_as_computable()` (in module `gooddata_fdw.column_utils`), 14
`TableOptions` (`gooddata_fdw.executor.InitData` property), 21
`TableDefinition` (in module `gooddata_fdw.environment`), 18
`TableDefinitionStub` (class in `gooddata_fdw.environment`), 18
`TableOptions` (class in `gooddata_fdw.options`), 31
`TableResultReader` (class in `gooddata_fdw.result_reader`), 32
`TableService` (class in `gooddata_sdk.table`), 201
`time_comparison_master` (`gooddata_sdk.insight.InsightMetric` property), 197
`to_date()` (`gooddata_sdk.type_converter.DateConverter` class method), 206
`to_datetime()` (`gooddata_sdk.type_converter.DatetimeConverter` class method), 207
`to_dict()` (`gooddata_sdk.catalog.base.Base` method), 34

`to_dict()` (`gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.date_dataset.date_dataset.CatalogGranula`
`method`), 144
`workspace` (`gooddata_fdw.import_workspace.ImporterInitData`
`workspace`), 144
`to_dict()` (`gooddata_sdk.catalog.workspace.declarative_model.workspace.logical_model.ldm.CatalogDeclarativeLdm`
`method`), 146
`WorkspaceImporter` (class in `good-`
`data_sdk.catalog.import_model.khpc.CatalogDeclarativeModel`
`WorkspaceImportersLocator` (class in `good-`
`data_sdk.catalog.import_model.khpc.CatalogDeclarativeModel`
`method`), 147
`WorkspaceImportersLocator` (class in `good-`
`data_sdk.catalog.import_model.khpc.CatalogDeclarativeModel`
`method`), 149
`write_layout_to_file()` (in module `good-`
`data_sdk.catalog.import_model.khpc.CatalogDeclarativeModel`
`method`), 149
`workspace_sdk.khpc.CatalogDeclarativeWorkspaceDataFilter`
`method`), 151
`workspace_sdk.khpc.CatalogDeclarativeWorkspaceDataFilters`
`method`), 154
`workspace_sdk.khpc.CatalogDeclarativeWorkspaceDataFilterSetting`
`method`), 153
`workspace_sdk.khpc.CatalogDeclarativeWorkspaceModel`
`method`), 155
`workspace_sdk.khpc.CatalogDeclarativeWorkspaces`
`method`), 156
`workspace_sdk.khpc.CatalogDependentEntitiesGraph`
`method`), 163
`workspace_sdk.khpc.CatalogDependentEntitiesNode`
`method`), 164
`workspace_sdk.khpc.CatalogDependentEntitiesRequest`
`method`), 165
`workspace_sdk.khpc.CatalogDependentEntitiesResponse`
`method`), 166
`workspace_sdk.khpc.CatalogEntityIdentifier`
`method`), 167
`TokenCredentials` (class in `good-`
`data_sdk.catalog.entity`), 78
`TokenCredentialsFromFile` (class in `good-`
`data_sdk.catalog.entity`), 79
`TotalDefinition` (class in `good-`
`data_sdk.compute.model.execution`), 180
`TotalDimension` (class in `good-`
`data_sdk.compute.model.execution`), 181
`TypeConverterRegistry` (class in `good-`
`data_sdk.type_converter`), 208

U

`USER_AGENT` (in module `gooddata_fdw.fdw`), 22

V

`validate_columns_in_table_def()` (in module
`gooddata_fdw.column_validation`), 14
`value_in_allowed()` (in module `good-`
`data_sdk.catalog.base`), 34
`VerticaAttributes` (class in `good-`
`data_sdk.catalog.data_source.entity_model.data_source`),
71

W

`wait_till_available()` (`good-`
`data_sdk.support.SupportService` `method`),