Kpi engine

Generated by Doxygen 1.12.0

Chapter 1

KPI calculation engine introduction

The KPI calculation engine is responsible for the calculation of kpis; it interacts with other parts of the system to calculate kpis. Along with the kpi engine we also have the alert monitor which has the responsibility of monitoring kpis from the machines and does soe by interaction with the kpi engine and other parts of the system as well.

Authors: Mirko Michele D'angelo and Riccardo Marcaccio

1.1 the calculation engine

The kpi calculation engine computations goes through the following steps to make a calculation

- parameter validation
- semantical validation
- data retrieval
- calculating the expression The parameter validation is handled by pydantic that can validate incoming parameters and a bit of code when necessary, for the semantic validation we have to check if the kpis requested can be calculated with a certain machine, if so then we proceed with data retrieval from the database and finally invoke the calculation engine to get the result.

This workflow is used for both calculation of normal expressions and simlarly for the expressions of the alert monitoring.

1.1.1 the APIs

the points exposed are 5 of which only 4 are meant for external communications while 1 is meant for internal ones(as in the picture above):

- **calculate** used by other parts of the system to ask for custom calculations.
- **add_alert** which adds a new alert to be monitored.
- **remove_alerts** which can remove specified alerts from the monitoring
- **get_all_alerts** gives all the currently monitored alerts.
- **alert** which is used as an internal communication between the alert monitor and the kpi engine to ask for custom calculations on the alerts expressions

all the endpoints are documented and can be tried at the address `localhost:8000/docs` please ensure that the docker container of the kpi engine is running and also the docker container for api layer is running and if you want to try out the apis also the containers for database and knowledge base.

1.1.2 Usage

Note: you need Docker installed, if not please install it.

To use test the apis make sure to enter the KPI engine directory, from there use the following command inside the KPI engine directory:

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

after that you can visit the address specified before from your browser.

1.2 technologies used

For this part we used different technologies, in particular we have a `Docker` container to make everything easy to run on different machines. All the requests are handled using `FastAPI` and `Pydantic` which proved useful, their combination allowed us to easily write a documentation of the apis to interact with other parts and also Pydantic can take care of most of the parameter validation thanks to it's models while fastapi is very flexible and reliable.

To realize the alert monitor we decided to use `APScheduler` which allows very flexible scheduling of jobs as well as a lot of options for job execution for different kind of tasks and need for storing the jobs to launch.

1.3 Code organization

Here we have a tree view to help understand the organization of code and how things are done.

```
KPI engine
 Dockerfile
 Doxyfile
 README.md
 app
    Alert_Monitor
       Tests
       alert_monitor.py
    Database
       Database_interface.py
    Documentation.md
    KPI_engine
       EngineCalculation
          calculation_engine.py
       Tests
    Knowledge_base
       knowledge_base_interface.py
    Tests
       test_alert_monitor.py
       test_api.py
       test_engine_kpi.py
    main.py
    models
       alert_requests.py
       calculation_request.py
    run_tests.sh
    utils
        calculation_utils.py
        utils.py
 demo
    demo.ipynb
 locustfile.py
 docker-compose.yml
 requirements.txt
 stress_load_test.py
```

This part is more technical and is thought to help navigate the code base of the kpi engine, for this reason when a path to something is specified it is assumed to start from the kpi engine folder. To avoid redundancy we are gonna explain what the most important files contain and then you can also view the code yourself which contains comments and other thing help understanding better how the system works.

- The *app/Database* and *app/Knowledge_base* folders contain the utilities used to interact with the knowledge base and database respectively and are used by the calculation engine to retrieve data and do semantical validation and also the alert monitor uses the knowledge base to do also semantical checking when adding new alerts.
- Inside the *app/KPI_engine/EngineCalculation* is the calculation engine code for the logic please refer to the Calculation logic section since it is more articulate and needs more explanations.
- The *app/models* folder contains the pydantic models definitions used to validate parameters and generate the documentation, too nderstand better their usage you can see the code in the relative files which is quite simple or read the summary in the dedicated section.
- *main.py* contains all the endpoint definitions explained in the previous section.
- *app/Alert_monitor.py* contains the alert monitor code, please visit the dedicated section for more details.
- *app/utils* contains utility files that we used to realize other components of the engine.
- *stress_load_test.py* contains the code used to test the calculation capabilities of the engine
- Finally to test the code we have the *Tests* folder which contains our unit tests for the kpi engine various parts.

Finally we also have a requirements.txt file containig all the dependencies needed, Dockerfile and docker-compose file used to setup our docker container, and the file used to generate this documentation.

1.3.1 Load testing

To test the capabilities we also decided to do some load testing, for this part we decide to use `locust`. The framework allows us to send multiple request with settings like:

- Number of users: upper bound on the number of users simulated.
- Ramp up: Number of users simulated added by second (until the upper bound is reached).
- Run time: Total time of test.
- Task: It is possible to describe task that every user can perform.

It is possible to get also quantitative graphs of data about KPI the engine performances according to number of users.

For our tests we also tried the alert monitor using multiple parallel users, each user can pick a random action between calculate,add,remove and get alerts, such test code is in the stress_load_test.py file.

1.3.2 Resource consumption

using docker can be heavy both from disk usage and memory usage however such problem is counterbalanced by the possibility offered by using the containers, at normal pace with no data the engine uses 165 mb of memory ram and has a cpu usage of 1.7% so it's not too much considered the capacities of normal machines.

Chapter 2

Calculation logic

The main point of the KPI engine is parsing string expression to do calculations. To do so, step by step,the KPI engine will do the following actions:

- **Take** the **string expression**
- **Parsing** the string to get a **syntax tree**
- Do **semantic checking**, to check if everything is ok (like no division by 0 etc.).
- **calculate the expression** after taking information from DataBase and KnowledgeBase.

The expression can either be a **mathematical expression** or an **alert** the difference being that mathematical expression can only use numerical operations and aggregation functions while alert expressions allows to use boolean operators and evaluate to only scalar-boolean expressions.

Considering variable *EXPR* like an expression that gives either a scalar number or a column of numbers, an **alert** is defined like:

Alert = **EXPR** *boolean operator* **EXPR**

Where **boolean operator** can be: $**=**$, $**<$, $**>$, etc.

To calculate values we use the following parameters:

- **string expression**: an expression in the format above.
- **temporal range**: a range specifing how far back the data used in the calculation can be taken.
- **machine id**: the id of the machine that has to be used.

Inside the expression, every variable can be:

- **Scalar-like** a single valuie.
- **base KPI/column-like** which is either a list of kpi values coming directly from the machines, it's values are drawn from the specified time range, or is the result of doing operations other column-like / base kpis.

The type of results that engine gets depends by syntax tree created during parsing phase, which depends on the formula written inside the string. In general there are 4 cases: a **float** and a **float's list** for mathematical expression, and a **bool** or a **bool's list** for alerts.

It is possible to apply some **aggregation** base functions on base kpi values that give back a scalar like sum, min, avg, var, max etc., they can also be applied to the result of doing operations between other column-like values and are appliable only to float column-like values.

In this case there are some clarifications that are due, with float values of both kinds you can do operations on both scalar and column-like types together also it is possible to sum scalar values to non scalar values if they both are of float type, we have three cases:

- operations on two column-like, which end up in another column-like and the operation is done element wise.
- operations on two scalars, which end up in another scalar and operation is appleid to single elements.
- operations on a column-like and a scalar, which end up in another column-like the operation is done element wise reusing the scalar with each element of the column using the operation.

This allows a lot of flexibility in the use of the engine to accommodate the functionalities needed by parts outside the system.

There are also some cases where engine can refuse to do calculations, giving an error, this cases are:

- String expression doesn' t follow the correct syntax structure.
- There is a division by 0.
- A variable is not a *base function* or a KPI base name.

We also put in place *Base functions* are an abstraction used to add custom calculations, this allows us to add dynamical kpis drawn from a KB or other future needs,note the KB must contain the formula corresponding to the kpis requested or the calculation cannot be done.

Such feature is used for alert and calculation capabilities offered in the API endpoints **calculate** and **alert**, you can see them from the `localhost:8000/docs` link offered by fastapi, here we are gonna explain how it works since it differs by the endpoints.

Please make sure the docker containers are up and running as asked in the introductory section if you want to test them out.

2.1 normal calculations

Note before starting: the data received by the engine is in the form of aggregated values meaning we don't get raw data but only aggregations of it at a certain point in time, in particular we get the sum, average, minimum and maximum to do our operation on the kpi values we need to select one of them using an aggregation selector.

For the the **calculate endpoint** we have to specify a time range, an aggregation operation and the expression we want to parse. To better undestand this imagine a set of filters and group operations being applied:

- select for which machine you want to do the calculation
- first we filter only the time range we want
- then we filter by picking the aggregation selector between 'sum','max','min','avg'
- now we can group together in time segments, in particular values can be grouped by day,week,month,year or not be grouped at all.
- finally for each segment the expression is applied as specified before and we have the result returned to us.

Note here boolean operators are not possible, use only mathematical ones.

2.2 alert calculations

Here the logic is similar, we use expressions but now they only give back scalar boolean values, for this part we still use the filter logic but it's different it's not possible to aggregate instead we have only a sliding window.

Each time the alert calculation is done the data selected is taken looking back starting from the current time minus the **sliding window size** which determines how far back we can look in the past.

the calculation goes as follows:

- pick the machine you want the data for
- pick a sliding window size to decide how far back you want to to take data for.
- write an expression as explained in this section
- finally you get the result back

this is meant for the alert monitor to use however it is testable using the openapi documentation available at `localhost:8000/docs`

2.2.0.1 technologies used

Parsing and **semantic checking** are done by `Lark` library, this tool allows semantic checking with more flexibility, e.g. check if there is a division by 0.
Calculation phase are made with `py_expression_eval` library, because this tool allows us to operate with strings using parallel calculations. In this way the calculations are faster.

To check the capacity of engine KPI and Alert monitor to manage more request in parallel, we used the **`Locust`** please visit the section from the kpi engine introduction to see how to use it.

Chapter 3

Knowledge base

It is a interface that comunicates with Knowledge based. This interface contains base query functions that extracts base information for the KB:

- *check_kpi_availability* : takes a **machine_id** and a **KPI's list**, this function gives a **boolean** saying wehter the request kpis can be calculated or not on the requested machine.
- *retrieve_kpi_data* = taking a **KPI's list**, this functions gives **information foreach KPI** inside the list, the information for each kpi is a dictionary with informations from the KB.
- *calculate_unit* = taking a **set of unit metrics**, this function gives a **unit**,it is used since the engine can also calculate and give back the unit of the calculation requested when it can be calculated.
- *get_base_kpis* = taking nothing, this functions gives a list of all **base KPI names** taken from the KB.

These functions are used only for taking information, from the KB, about KPI and machines during **semantic validation phase**, by **engine KPI** and **API**.

Chapter 4

Alert monitoring

The alert monitor has the responsibility of checking kpi expressions e.g. "sum(cycles)", it knows which are the kpis to monitor and does so by interacting with the kpi engine to calculate alert expressions. The way it works is that it does polling on a defined time-interval using the following informations:

- the machine for which the values are to be calculated.
- the expression used.
- the sliding window size i.e. how far back from the time it was polling it has to take data from.

The actual calculation is done by the kpi engine, the alert monitor uses the time range determined by the sliding window size and asks the kpi engine to calculate a boolean expression, if it evaluates to true then it has to contact the api layer and notify it that an alert has been fired giving the necessary informations.

Also the alert monitor holds information about which expression are to be monitored.

Chapter 5

Database

It is a interface that comunicates with Databse. This interface contains base query functions that extract values:

- *retrieve_data_db* = Taking **machine_id**, the **KPI's list**, **aggregation_operation**, and **range**, this function get a matrix of values, where every row is a **KPI** (inside **KPI's list**) of **machine** (identified by **machine_id**). All rows are referred to the temporal range described by **range**.

This function are used only for taking information during calculation phase by KPI engine.

As an additional feature, there is also *get_time_range* function, that taking **time_range**, **start_time**, **end_time**, this function gives time inside **time_range** (and between **start_time** - **end_time**) converting to datetime objects (the conventional form). This function is not directly correlated to DB connection, but it is used for converting string date in more conventional form for calculation, and also for filtering date outside the temporal range.

Chapter 6

Namespace Index

6.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 8

Class Index

8.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 9

File Index

9.1 File List

Here is a list of all files with brief descriptions:

Chapter 10

Namespace Documentation

10.1 Alert_monitor Namespace Reference

Namespaces

- namespace alert_monitor

10.2 Alert_monitor.alert_monitor Namespace Reference

Classes

- class Alert
- class AlertMonitor

Functions

- None fire_alert (int machine_id, List[str] kpis, str expression, float time_range)
- test_alerts ()

Variables

- parent_dir = os.path.abspath(os.path.join(os.getcwd(), ".."))
- level
- INFO
- format
- logger = logging.getLogger()

10.2.1 Function Documentation

10.2.1.1 fire_alert()

```
None Alert_monitor.alert_monitor.fire_alert (
            int machine_id,
            List[str] kpis,
            str expression,
            float time_range)
```

notify the alert to the api layer

```
Args:
    machine_id: the id of the machine
    kpis: the kpis involved
    expression: the expression calculated
    time_range: the sliding window time range
```

10.2.1.2 test_alerts()

```
Alert_monitor.alert_monitor.test_alerts ()
```

10.2.2 Variable Documentation

10.2.2.1 format

```
Alert_monitor.alert_monitor.format
```

10.2.2.2 INFO

```
Alert_monitor.alert_monitor.INFO
```

10.2.2.3 level

```
Alert_monitor.alert_monitor.level
```

10.2.2.4 logger

```
Alert_monitor.alert_monitor.logger = logging.getLogger()
```

10.2.2.5 parent_dir

```
Alert_monitor.alert_monitor.parent_dir = os.path.abspath(os.path.join(os.getcwd(), ".."))
```

10.3 Database Namespace Reference

Namespaces

- namespace Database_interface

10.4 Database.Database_interface Namespace Reference

Classes

- class DBConnection

Variables

- str DB_URL = f"{BASE_URL}/data/raw"

10.4.1 Variable Documentation

10.4.1.1 DB_URL

```
str Database.Database_interface.DB_URL = f"{BASE_URL}/data/raw"
```

10.5 Knowledge_base Namespace Reference

Namespaces

- namespace knowledge_base_interface

10.6 Knowledge_base.knowledge_base_interface Namespace Reference

Classes

- class KnowledgeBaseInterface

10.7 KPI_engine Namespace Reference

Namespaces

- namespace EngineCalculation

10.8 KPI_engine.EngineCalculation Namespace Reference

Namespaces

- namespace calculation_engine

10.9 KPI_engine.EngineCalculation.calculation_engine Namespace Reference

Classes

- class CalculationEngine

Chapter 11

Class Documentation

11.1 Alert_monitor.alert_monitor.Alert Class Reference

Public Member Functions

- __init__ (self, float date_range_seconds, str expression, int machine_id)

Public Attributes

- float date_range = date_range_seconds
- str expression = expression
- int machine_id = machine_id

11.1.1 Constructor & Destructor Documentation

11.1.1.1 __init__()

```
Alert_monitor.alert_monitor.Alert.__init__ (
              self,
             float date_range_seconds,
             str expression,
             int  machine_id)
```

11.1.2 Member Data Documentation

11.1.2.1 date_range

```
float Alert_monitor.alert_monitor.Alert.date_range = date_range_seconds
```

11.1.2.2 expression

```
str Alert_monitor.alert_monitor.Alert.expression = expression
```

11.1.2.3 machine_id

```
int Alert_monitor.alert_monitor.Alert.machine_id = machine_id
```

The documentation for this class was generated from the following file:

- app/Alert_monitor/alert_monitor.py

11.2 Alert_monitor.alert_monitor.AlertMonitor Class Reference

Public Member Functions

- __new__ (cls)
- str add_alert (self, Alert alert)
- None load_config (self)
- None start (self)
- List[Dict] get_all_alerts (self, list[str] alert_ids=[ ])
- List[Dict[str, bool]] remove_alerts (self, List[str] alerts_ids=[ ])

Protected Member Functions

- None _make_alert_request (self, float date_range, str expression, int machine_id, datetime starting_↩ date=datetime.now())
- _reset (self)

Protected Attributes

- _make_alert_request

11.2.1 Member Function Documentation

11.2.1.1 __new__()

```
Alert_monitor.alert_monitor.AlertMonitor.__new__ (
              cls)
```

11.2.1.2 _make_alert_request()

```
None Alert_monitor.alert_monitor.AlertMonitor._make_alert_request (
              self,
             float date_range,
             str expression,
             int machine_id,
             datetime starting_date = datetime.now())  [protected]
```

```
make an alert calculation request to the kpi engine

Args:
    date_range: how far back in time the sliding window should go, specified in seconds
    expression: the expression to calculate
    machine_id: on which machine it should be calculated
    starting_date: from which point in time the sliding window should start
```

11.2.1.3 _reset()

```
Alert_monitor.alert_monitor.AlertMonitor._reset (
              self)  [protected]
```

11.2.1.4 add_alert()

```
str Alert_monitor.alert_monitor.AlertMonitor.add_alert (
              self,
             Alert alert)
```

add a new alert to be monitored

11.2.1.5 get_all_alerts()

```
List[Dict] Alert_monitor.alert_monitor.AlertMonitor.get_all_alerts (
              self,
             list[str] alert_ids = [])
```

retrieve all stored alerts

Args:
    alert_ids: contains the ids to remove

Returns:
    a list containing all the laerts stored

11.2.1.6 load_config()

```
None Alert_monitor.alert_monitor.AlertMonitor.load_config (
              self)
```

11.2.1.7 remove_alerts()

```
List[Dict[str,bool]] Alert_monitor.alert_monitor.AlertMonitor.remove_alerts (
              self,
             List[str] alerts_ids = [])
```

remove all specified alerts

Args:
    alert_ids: contains the ids to remove

Returns:
    a list of key value pairs each one, for each key we have true if it was removed or false if it wasn't

11.2.1.8 start()

```
None Alert_monitor.alert_monitor.AlertMonitor.start (
              self)
```

start the engine

11.2.2 Member Data Documentation

11.2.2.1 _make_alert_request

```
Alert_monitor.alert_monitor.AlertMonitor._make_alert_request  [protected]
```

The documentation for this class was generated from the following file:

- app/Alert_monitor/alert_monitor.py

11.3 KPI_engine.EngineCalculation.calculation_engine.Calculation↩ Engine Class Reference

Classes

- class Calculator
- class GeneralChecking

Public Member Functions

- Union[float, list[float]] direct_calculation_KPI (machine, str formula, str start_date, str end_date)
- Union[float, list[float]] direct_calculation_alert (str machine, str formula, str start_date, str end_date)
- bool add_complex_KPI (str name, str description, str expression)
- bool remove_complex_KPI (str name)
- Union[None, Calculator] get_complex_KPI (str name)
- bool add_alert (str name, str description, str expression)
- bool remove_alert (str name)
- Union[None, Calculator] get_alert (str name)
- list[str] get_alert_names ()
- list[str] get_complex_KPI_names ()
- None save_state (path="")
- None load_state (path="")

Protected Member Functions

- None _update_parser ()

Static Protected Attributes

- dict _base_functions_dict
- _complex_KPIs_dict = dict()
- _total_calculators = dict()

11.3.1 Detailed Description

```
@class CalculationEngine
@brief Descrizione di CalculationEngine

This class demonstrates detailed documentation for Python.
```

11.3.2 Member Function Documentation

11.3.2.1 _update_parser()

```
None KPI_engine.EngineCalculation.calculation_engine.CalculationEngine._update_parser ()  [protected]
```

11.3.2.2 add_alert()

```
bool KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.add_alert (
             str name,
             str description,
             str expression)
```

11.3.2.3 add_complex_KPI()

```
bool KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.add_complex_KPI (
             str name,
             str description,
             str expression)
```

11.3.2.4 direct_calculation_alert()

```
Union[float, list[float]] KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.↩
direct_calculation_alert (
             str machine,
             str formula,
             str start_date,
             str end_date)


@param: machine, formula, temporal range (start_date, end_date)
Compile, check and calculate alert

@return: value of alert
```

11.3.2.5 direct_calculation_KPI()

```
Union[float, list[float]] KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.↩
direct_calculation_KPI (
              machine,
             str formula,
             str start_date,
             str end_date)
```

@param: machine, formula, temporal range (start_date, end_date)
Compile, check and calculate complex KPI

@return: value of complex KPI

11.3.2.6 get_alert()

```
Union[None, Calculator] KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.↩
get_alert (
             str name)
```

11.3.2.7 get_alert_names()

```
list[str] KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.get_alert_names ()
```

11.3.2.8 get_complex_KPI()

```
Union[None, Calculator] KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.↩
get_complex_KPI (
             str name)
```

11.3.2.9 get_complex_KPI_names()

```
list[str] KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.get_complex_KPI_↩
names ()
```

11.3.2.10 load_state()

```
None KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.load_state (
              path = "")
```

11.3.2.11 remove_alert()

```
bool KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.remove_alert (
             str name)
```

11.3.2.12 remove_complex_KPI()

```
bool KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.remove_complex_KPI (
            str name)
```

11.3.2.13 save_state()

```
None KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.save_state (
             path = "")
```

11.3.3 Member Data Documentation

11.3.3.1 _base_functions_dict

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine._base_functions_dict
[static], [protected]
```

Initial value:

```
=  {
        "max": lambda x: max(x),
        "min": lambda x: min(x),
        "sum": lambda x: sum(x),
        "avg": lambda x: np.mean(x),
        "var": lambda x: np.var(x),
    }
```

11.3.3.2 _complex_KPIs_dict

```
KPI_engine.EngineCalculation.calculation_engine.CalculationEngine._complex_KPIs_dict = dict()
[static], [protected]
```

11.3.3.3 _total_calculators

```
KPI_engine.EngineCalculation.calculation_engine.CalculationEngine._total_calculators = dict()
[static], [protected]
```

The documentation for this class was generated from the following file:

- app/KPI_engine/EngineCalculation/calculation_engine.py

11.4 KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.Calculator Class Reference

Public Member Functions

- __init__ (self, name, description, expression, final_type, KPIs, base_functions, complex_KPIs)
- Union[float, bool, list[float], list[bool]] __call__ (self, str machine, str start_date, str end_date)
- str get_name (self)
- str get_description (self)
- str get_expression (self)
- list[str] get_KPIs (self)
- get_complex_KPIs (self)
- type get_result_type (self)
- list[str] get_base_functions (self)

11.4.1 Constructor & Destructor Documentation

11.4.1.1 __init__()

```
KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.Calculator.__init__ (
              self,
              name,
              description,
              expression,
              final_type,
              KPIs,
              base_functions,
              complex_KPIs)
```

11.4.2 Member Function Documentation

11.4.2.1 __call__()

```
Union[float, bool, list[float], list[bool]] KPI_engine.EngineCalculation.calculation_engine.↩
CalculationEngine.Calculator.__call__ (
              self,
             str machine,
             str start_date,
             str end_date)
```

11.4.2.2 get_base_functions()

```
list[str] KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.Calculator.get_↩
base_functions (
              self)
```

11.4.2.3 get_complex_KPIs()

```
KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.Calculator.get_complex_KPIs
(
              self)
```

11.4.2.4 get_description()

```
str KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.Calculator.get_description
(
              self)
```

11.4.2.5 get_expression()

```
str KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.Calculator.get_expression
(
              self)
```

11.4.2.6 get_KPIs()

```
list[str] KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.Calculator.get_↩
KPIs (
              self)
```

11.4.2.7 get_name()

```
str KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.Calculator.get_name (
              self)
```

11.4.2.8 get_result_type()

```
type KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.Calculator.get_result↩
_type (
              self)
```

The documentation for this class was generated from the following file:

- app/KPI_engine/EngineCalculation/calculation_engine.py

11.5 Database.Database_interface.DBConnection Class Reference

Public Member Functions

- tuple retrieve_data_db (int machine, List[str] KPIs, str aggregation_operation, tuple range)
- np.ndarray[datetime] get_time_range (list[str] time_range, datetime start_time, datetime end_time)

11.5.1 Member Function Documentation

11.5.1.1 get_time_range()

```
np.ndarray[datetime] Database.Database_interface.DBConnection.get_time_range (
             list[str] time_range,
             datetime start_time,
             datetime end_time)
```

11.5.1.2 retrieve_data_db()

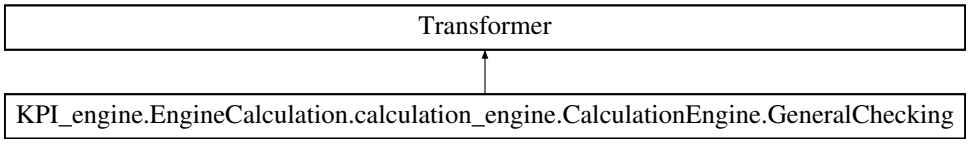
```
tuple Database.Database_interface.DBConnection.retrieve_data_db (
             int machine,
             List[str] KPIs,
             str aggregation_operation,
             tuple range)

retrieve specified data from the db in a certain time range

Args:
    machine: the machine for which the data is reuqested
    KPIs: the KPIs for which data is requested
    aggregation_operation: the aggregation operation used on the data
    range: the time range for which to retrieve data
```

The documentation for this class was generated from the following file:

- app/Database/Database_interface.py

11.6 KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking Class Reference

Inheritance diagram for KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking:



Public Member Functions

- dict base (self, dict args)
- dict le (self, dict args)
- dict ge (self, dict args)
- dict eq (self, dict args)
- dict neq (self, dict args)
- dict leq (self, dict args)
- dict geq (self, dict args)
- dict add (self, dict args)
- dict sub (self, dict args)
- dict mul (self, dict args)
- dict div (self, dict args)
- dict pow (self, dict args)
- dict inverse_sign (self, dict args)
- dict kpi (self, dict args)
- dict number (self, dict args)
- dict apply_base_function (self, dict args)
- dict apply_calculators (self, dict args)
- dict brackets (self, dict args)

11.6.1 Member Function Documentation

11.6.1.1 add()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.add (
              self,
             dict args)
```

11.6.1.2 apply_base_function()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.apply↩
_base_function (
              self,
             dict args)
```

11.6.1.3 apply_calculators()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.apply↩
_calculators (
              self,
             dict args)
```

11.6.1.4 base()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.base (
              self,
             dict args)
```

11.6.1.5 brackets()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.↩
brackets (
              self,
             dict args)
```

11.6.1.6 div()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.div (
              self,
             dict args)
```

11.6.1.7 eq()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.eq (
              self,
             dict args)
```

11.6.1.8 ge()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.ge (
              self,
             dict args)
```

11.6.1.9 geq()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.geq (
              self,
             dict args)
```

11.6.1.10 inverse_sign()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.↩
inverse_sign (
              self,
             dict args)
```

11.6.1.11 kpi()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.kpi (
              self,
             dict args)
```

11.6.1.12 le()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.le (
              self,
             dict args)
```

11.6.1.13 leq()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.leq (
              self,
             dict args)
```

11.6.1.14 mul()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.mul (
              self,
             dict args)
```

11.6.1.15 neq()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.neq (
              self,
             dict args)
```

11.6.1.16 number()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.number
(
              self,
             dict args)
```

11.6.1.17 pow()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.pow (
             self,
            dict args)
```

11.6.1.18 sub()

```
dict KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking.sub (
             self,
            dict args)
```

The documentation for this class was generated from the following file:

- app/KPI_engine/EngineCalculation/calculation_engine.py

11.7 Knowledge_base.knowledge_base_interface.KnowledgeBaseInterface Class Reference

Public Member Functions

- bool check_kpi_availability (int machine_id, list[str] kpis)
- list[dict] retrieve_kpi_data (list[str] kpis)
- str calculate_unit (set[str] units)
- list[str] get_base_kpis ()

11.7.1 Member Function Documentation

11.7.1.1 calculate_unit()

```
str Knowledge_base.knowledge_base_interface.KnowledgeBaseInterface.calculate_unit (
            set[str] units)
```

11.7.1.2 check_kpi_availability()

```
bool Knowledge_base.knowledge_base_interface.KnowledgeBaseInterface.check_kpi_availability (
            int machine_id,
            list[str] kpis)
```

11.7.1.3 get_base_kpis()

```
list[str] Knowledge_base.knowledge_base_interface.KnowledgeBaseInterface.get_base_kpis ()
```

11.7.1.4 retrieve_kpi_data()

```
list[dict] Knowledge_base.knowledge_base_interface.KnowledgeBaseInterface.retrieve_kpi_data (
            list[str] kpis)
```

The documentation for this class was generated from the following file:

- app/Knowledge_base/knowledge_base_interface.py

Chapter 12

File Documentation

12.1 app/Alert_monitor/Alert_monitor.md File Reference

12.2 app/Alert_monitor/alert_monitor.py File Reference

Classes

- class Alert_monitor.alert_monitor.Alert
- class Alert_monitor.alert_monitor.AlertMonitor

Namespaces

- namespace Alert_monitor
- namespace Alert_monitor.alert_monitor

Functions

- None Alert_monitor.alert_monitor.fire_alert (int machine_id, List[str] kpis, str expression, float time_range)
- Alert_monitor.alert_monitor.test_alerts ()

Variables

- Alert_monitor.alert_monitor.parent_dir = os.path.abspath(os.path.join(os.getcwd(), ".."))
- Alert_monitor.alert_monitor.level
- Alert_monitor.alert_monitor.INFO
- Alert_monitor.alert_monitor.format
- Alert_monitor.alert_monitor.logger = logging.getLogger()

12.3 app/Database/Database.md File Reference

12.4 app/Database/Database_interface.py File Reference

Classes

- class Database.Database_interface.DBConnection

Namespaces

- namespace Database
- namespace Database.Database_interface

Variables

- str Database.Database_interface.DB_URL = f"{BASE_URL}/data/raw"

12.5 app/Documentation.md File Reference

12.6 app/Knowledge_base/Knowledge_base.md File Reference

12.7 app/Knowledge_base/knowledge_base_interface.py File Reference

Classes

- class Knowledge_base.knowledge_base_interface.KnowledgeBaseInterface

Namespaces

- namespace Knowledge_base
- namespace Knowledge_base.knowledge_base_interface

12.8 app/KPI_engine/Calculation_logic.md File Reference

12.9 app/KPI_engine/EngineCalculation/calculation_engine.py File Reference

Classes

- class KPI_engine.EngineCalculation.calculation_engine.CalculationEngine
- class KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.Calculator
- class KPI_engine.EngineCalculation.calculation_engine.CalculationEngine.GeneralChecking

Namespaces

- namespace KPI_engine
- namespace KPI_engine.EngineCalculation
- namespace KPI_engine.EngineCalculation.calculation_engine

Index