TECO Research Group

Marcel Köpke
Matthias Budde
Till Riedel

Entwurfsdokument
Version 0.1

Visualizing & Mining of Geospatial Sensorstreams with Apache Kafka

---

Jean Baumgarten
Thomas Frank
Oliver Liu
Patrick Ries
Erik Wessel

1. Juli 2018

Inhaltsverzeichnis

1 Einleitung

2 Sequenzdiagramme

Die folgenden Sequenzdiagramme sollen den Ablauf von einzelnen Anwendungsfällen im PaVoS-System illustrieren. Die Interaktionen der Klassen miteinander in verschiedenen Situationen wird somit verdeutlicht.

2.1 Bridge

In diesem Sequenzdiagramm wird der Ablauf der Bridge beschrieben, die MQTT-Nachrichten in Records umwandelt und diese an Kafka weiterleitet. Die Bridge läuft komplett unabhängig vom restlichen System.
Die Bridge kann sich in einer von drei Phasen befinden:

1. **Aufbauphase:** Hier findet die Prüfung der Parameter und das Initialisieren der benötigten Klassen statt.
2. **Bereitschaftsphase:** Hier ist die Bridge bereit, Nachrichten von MQTT anzunehmen, zu konvertieren und an Kafka weiter zu senden.
3. **Abbauphase:** Hier werden die Verbindungen zu MQTT und Kafka getrennt, anschließend wird die Bridge beendet.

In der Aufbauphase (in diesem Diagramm Operationen 1-5) wird zunächst ein `JmkbKafkaProducer` erstellt, der intern einen `KafkaProducer` mit bestimmten Einstellungen initialisiert und eine Verbindung zum Kafka Broker aufbaut. Danach wird ein `JmkbMqttConsumer` erstellt, der intern einen `MqttClient` mit bestimmten Einstellungen initialisiert, welcher eine Verbindung zum MQTT-Server aufbaut und die Topics abonniert, die vom FROST-Server angeboten werden.

Nun beginnt die Bereitschaftsphase. Sobald eine Nachricht beim MqttClient ankommt, wird die Methode `messageArrived` des `JmkbMqttConsumer`s aufgerufen. In dieser Methode wird aus der erhaltenen Nachricht die IOT-ID des Sensors gefiltert und die Nachricht wird in das Avro-Format konvertiert. Diese zwei Daten sind dann key und value für das Kafka `ProducerRecord` und werden über einen Aufruf der `send`-Methode des `JmkbKafkaProducer`s in ein solches Format gewandelt. Anschließend wird das Record durch den KafkaProducer an Kafka gesendet.

In der Abbauphase werden die `disconnect`-Methoden von `JmkbMqttConsumer` und `JmkbKafkaProducer` aufgerufen, die jeweils die Verbindungen zu MQTT und Kafka sauber trennen und die Clients schließen. Die Abbauphase beginnt nur dann, wenn der Nutzer des Programms es willkürlich schließt oder das System es beendet.

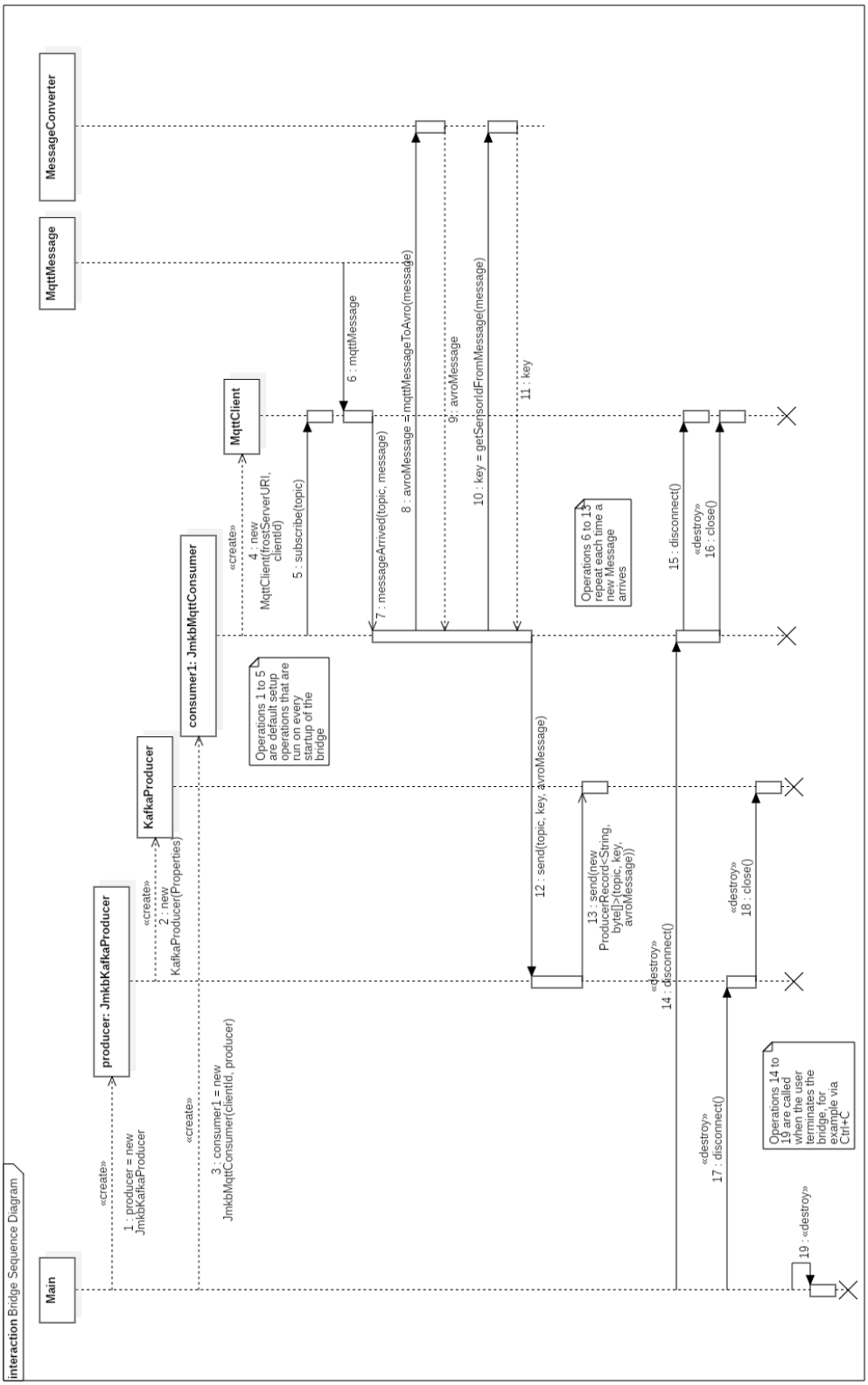


Abb. 2.1: Sequenzdiagramm Bridge

2.2 Core

Beim Controller werden alle Topics welche von dem MQTT Producer generierten wurden subscribed (abonniert) in einer Schleife. Dann macht der Controller mit der generateOutputtopic einen neuen Output Topic für eine StreamProcessingStrategy, weil dieser einen Output Topic benötigt um die verarbeiteten Daten abzulagern. Der Controller macht ein TopolgyBuilder Objekt, weil mit diesem die StreamProcessingStrategy, ausgeführt werden können. Der Controller übergibt mit addSource dem TopolgyBuilder einen Input Topic wo sich die zu verarbeiteten Daten in einem Kafka Stream enthalten sind. Der Controller macht eine neues StreamProcessingStrategy, welche die Methode ist wie die Inputdaten verarbeitet werden sollen. Der Controller übergibt den TopolgyBuilder mit addProcessor diese StreamProcessingStrategy. Der Controller übergibt den TopolgyBuilder mit addSink den vorhin generieten Output Topic, welcher dieser als Daten Sink für von dem StreamProcessingStrategy verarbeiteten Daten nutzt. Der TopolgyBuilder startet dann mit kafkaStreamStart die StreamProcessingStrategy und dieser fängt mit apply aus dem Input Topic Daten in den Output Topic zu schreiben bis der TopolgyBuilder kafkaStreamclose aufruft und dann die Verarbeitung stoppt und der TopolgyBuilder und StreamProcessingStrategy zerstört werden.

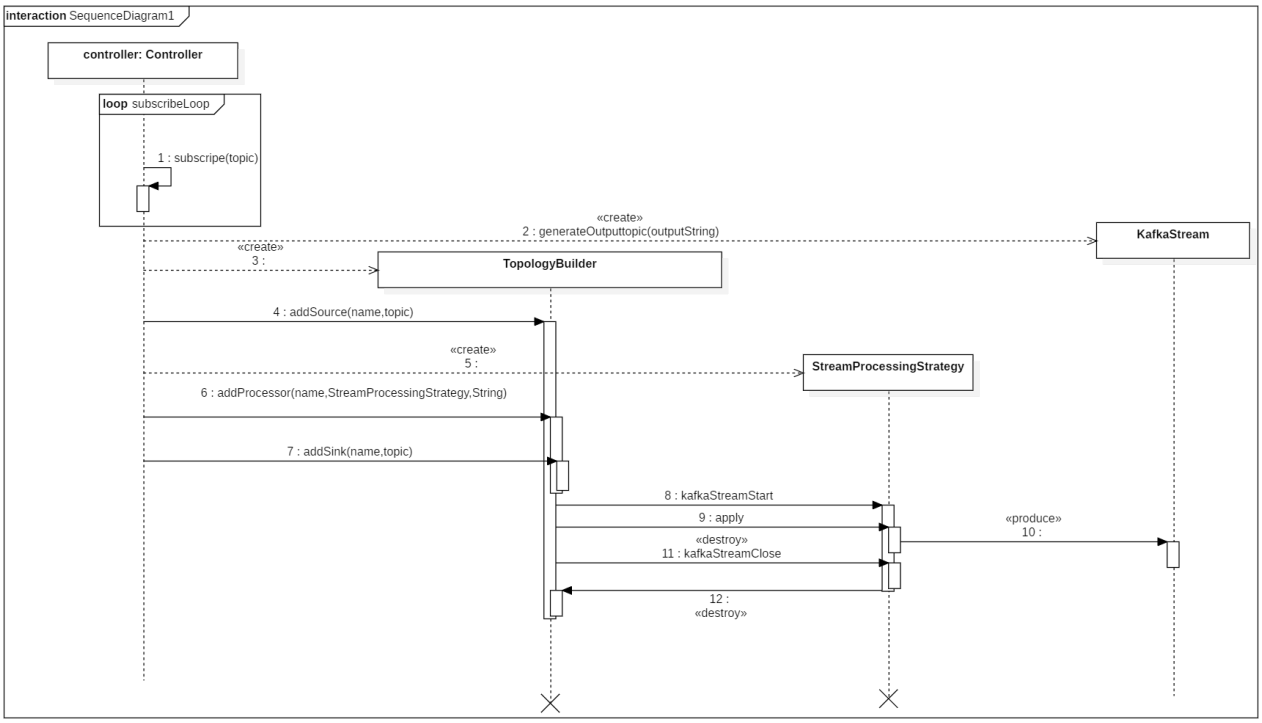


Abb. 2.2: Sequenzdiagramm Core

2.3 Import

Bei dem Import wird zuerst in dem Importordner nach Dateien gesucht und danach für jede vorhandene Datei ein separater Importprozess gestartet. Das folgende Sequenzdiagram stellt diesen Vorgang dar. Hier wird ausschließlich der Import behandelt, wer diesen Anstößt soll nicht Teil des Diagrams sein. `External` soll hier das Element darstellen, das den Import aufruft. Dazu wird ein `DataImporter` erstellt und seine Methode `startImportingFileData` aufgerufen, womit der Importvorgang startet.

Für jede Datei in dem Importordner wird nun ein `FrostSender` und einen `FilePath` der zum Pfad der Datei passt. Ist dies geschehen wird der `FileImporter` für diese Datei erschaffen und mit `addFileData` gestartet. Dazu wird der Pfad und der `FrostSender` mitübergeben. Aus dem Pfad wird jetzt eine `FileExtension` generiert, die dazu genutzt wird über den `ReaderType` eine Instanz einer Implementierung der `FileReaderStrategy` zu erhalten. Ist die `FileExtension` nicht bekannt würde es hier zu einer Exception kommen und der Import für diese Datei beendet.

In diesem Fall wurde als Beispiel eine `CSVReaderStrategy` genommen. Diese übernimmt den tatsächlichen Import der Daten aus der Datei zum FROST-Server vor. Dazu werden nach und nach einzelne Datensätze aus der Datei ausgelesen und über den `FrostSender` an den Server gesendet.

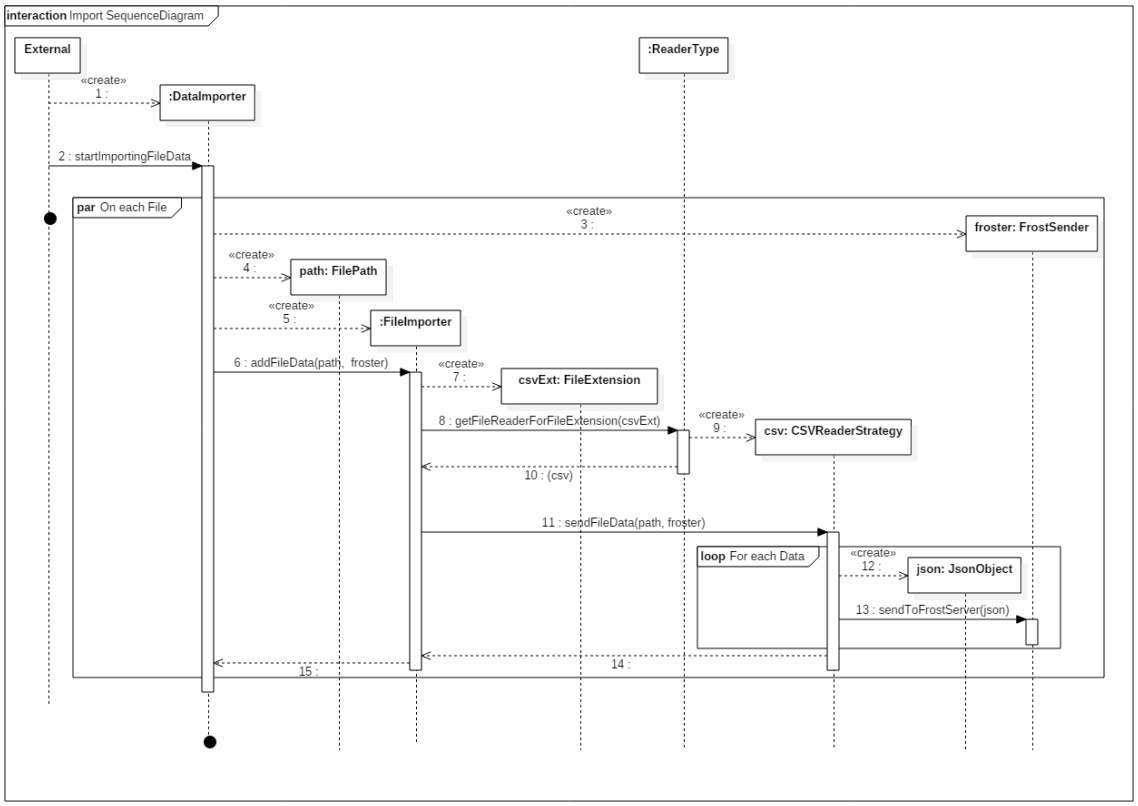


Abb. 2.3: Sequenzdiagramm Import

2.4 Graphite

The user selects data to be requested through the use of the webinterface. Then the Servlet receives that request and tells us, what data we have to transfer. This information is passed on to the GraphDataTransferController, which in return passes it to a newly created KafkaToGraphiteConsumer. This also creates us a GraphiteSender automatically in the constructor. This KafkaToGraphiteConsumer is then run on the information. It gathers different properties that are needed for transfering data from Kafka from the GraphiteConfig and creates a KafkaConsumer to subscribe to Kafka. We then check whether we want to seek to the beginning . After that, we enter a loop. Here we start polling data from Kafka and storing them in a ConsumerRecords object. Finally, we check if there was any new information in the polled data. If there was, we send the data by using our GraphiteSender. If we want to end the transferring of data, we have to call the wakeup method of our KafkaToGraphiteConsumer.

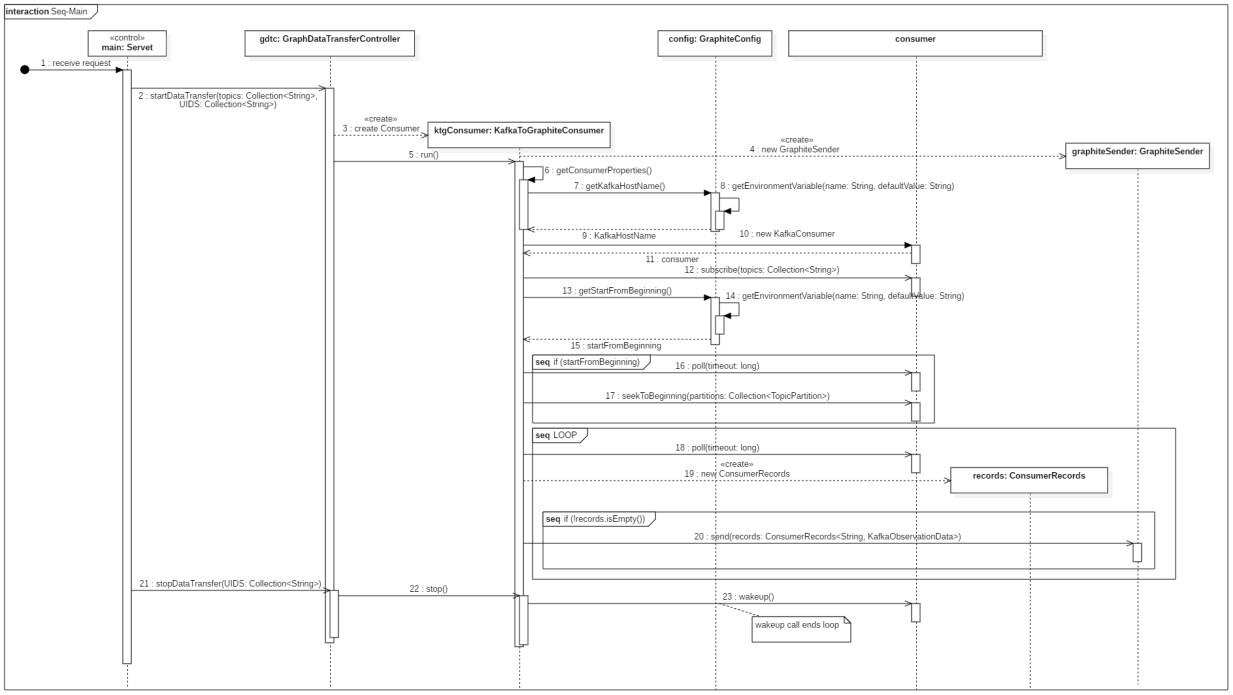


Abb. 2.4: Sequenzdiagramm Graphite

Here the data that we want to send to Graphite is given to us directly. To do the job, the GraphiteSender gathers properties from the GraphiteConfig, that are needed for transferring data to Graphite. Afterwards, we start a loop. In this loop the GraphiteSender adds every observed property to the list of data to send to Graphite. The GraphiteSender does this by converting the data to metrics and then documenting the results. After adding all observed properties, we can finally sent the data to Graphite.

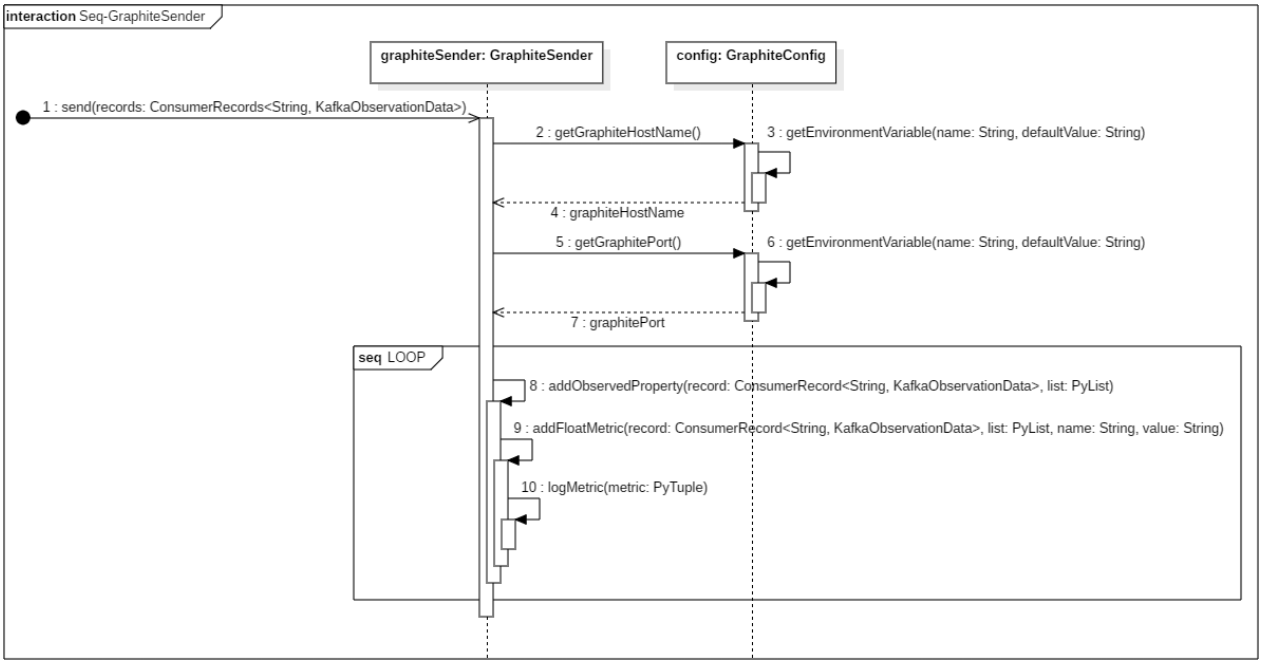


Abb. 2.5: Sequenzdiagramm GraphiteSender

2.5 Export

Der Export wird in der WebGUI von einem Nutzer angefragt. Die Daten für diesen Export werden an das `ExportServlet` übertragen, das den tatsächlich export der Daten in eine Datei Verwaltet. Ist diese Datei einmal erstellt kann diese von dem Nutzer heruntergeladen werden. Dazu mehr im folgenden Abschnitt über den Download. Dieses Sequenzdiagram zeigt wie der Export der Daten in eine Datei durchgeführt wird.

Sobald ein Export angestoßen wird, startet das `ExportServlet` und die Methode `doGet` wird ausgeführt. Darin werden zuerst die `ExportProperties` ais der Datei ausgelesen und zu einem Objekt zusammengestellt, das unter anderem Eine `FileExtension` enthält. Danach wird ein `FileExporter` konstruiert, der in zwei Schritten vorgehen wird, um die Daten zu exportieren.

Im ersten Schritt, wird durch den Aufruf der `createFileInformation`-Methode der Export für den späteren Download eindeutig identifiziert indem ihm eine `DownloadID` zugewiesen wird. Ein `AlterableDownloadState` wird erstellt und dessen Methode `savePersistent` ausgeführt, damit die Information über den Download auch auf dem Server hinterlegt wird, sodass paralell zum Export auch eine Anfrage gesendet werden kann, ob die Datei bereits fertig für den Download ist. Die `DownloadID` wird dann an den Nutzer zurückgesendet sobald der zweite Teil mit der `createFile`-Methode des `FileExporters` gestartet wurde.

Im zweiten Teil findet dann der tatsächliche Export der Daten in eine Datei statt. Dazu wird zuerst ein `ExportStreamGenerator` konstruiert, dessen Methode `createExportStream` einen `KStream` der gewünschten Daten für den Export erzeugt. Die Gewünschten Daten gehen aus den `ExportProperties` hervor. Mit der `FileExtension` aus den `ExportProperties` kann jetzt ein `FileType` generiert werden, über dessen Methode `getFileWriter` eine neue Instanz einer Implementierung einer `FileWriterStrategy` zurückgegeben wird. Dazu wird die statische Methode `getFileWriterForFileExtension` der Utilityklasse `FileTypesUtility` verwendet. In diesem Sequenzdiagram wird als Beispiel eine Instanz der `CSVWriterStrategy` verwendet.

Nun wird ein passender neuer Pfad als `FilePath` erzeugt, um die Datei zu erzeugen. Dazu wird die Methode `saveToFile` eines `FileWriterStrategy` genutzt. In diesem Fall einer `CSVWriterStrategy`. Diese Methode benötigt den ZielPfad und den Stream der Daten und erzeugt daraus eine Datei. Ist dies beendet, wird der `AlterableDownloadState` dazu genutzt die nötigen Informationen abzuspeichern. Zuerst wird der Pfad der Datei eingegeben und anschließend, dass die Datei bereit für den Download ist. Zum Schluß wird noch mit `savePersistent` sichergestellt, dass andere Instanzen eines `DownloadState` diese Information abrufen können.

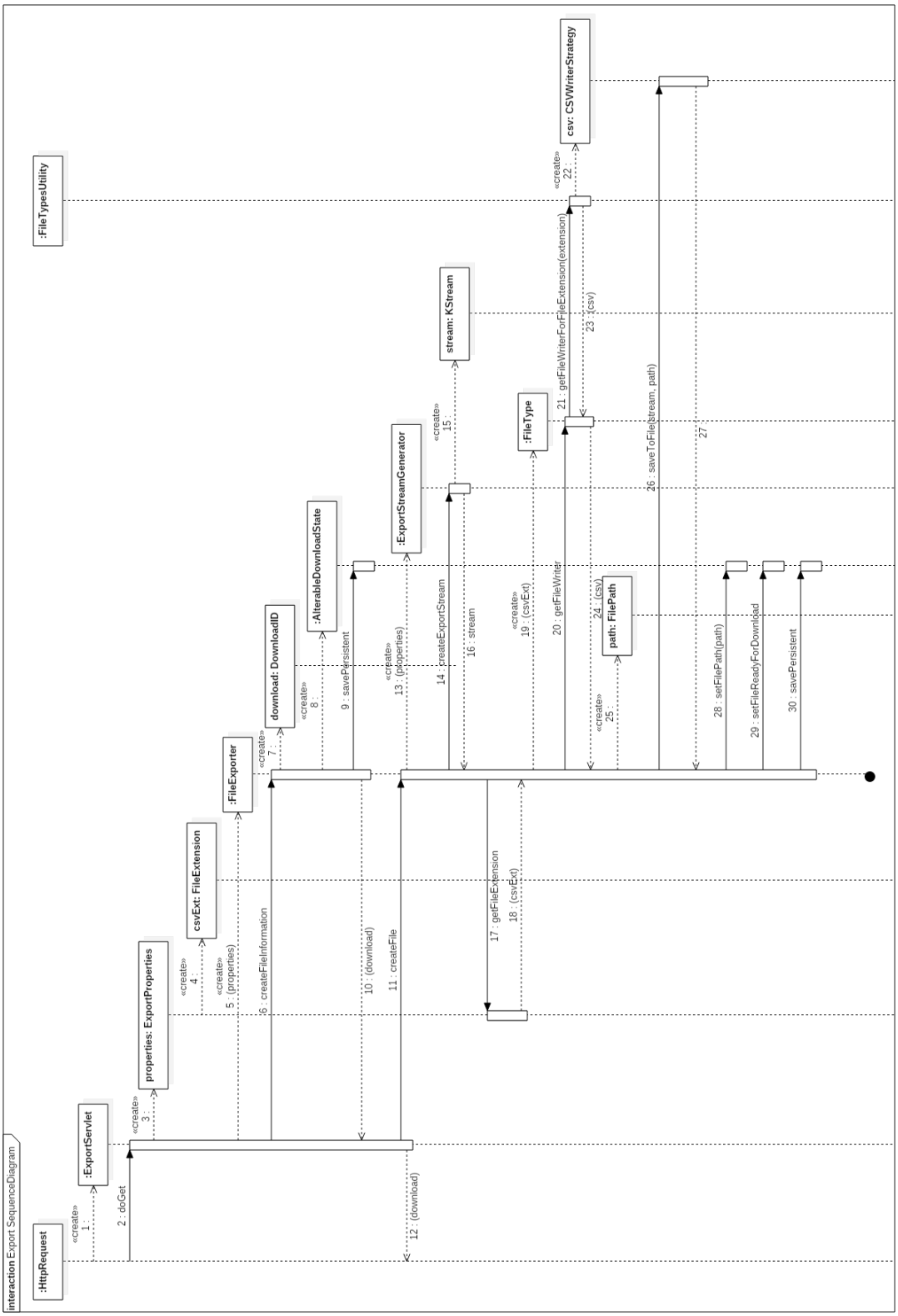


Abb. 2.6: Sequenzdiagramm Export

Ein Download wird grundsätzlich von einem Nutzer aus einem Browserfenster angefragt. Dazu wird das `DownloadServlet` benutzt. Diese wird vom Server erstellt sobald eine Anfrage des Nutzer einkommt. Dann wird `doGet` aufgerufen und das Servlet beginnt mit seiner Aufgabe, die in diesem Sequenzdiagram dargestellt wird.

Die Anfrage des nutzers enthält eine DownloadID, die für eine bestimmte Datei auf dem Server steht. Diese wird benutzt um eine `DownloadID` Objekt zu erstellen, das dazu dient einen `DownloadState` zu konstruieren. Dieser holt sich, sobald er erstellt wurde, die Informationen zu dem betreffenden Download. Diese Informationen könnten in einer Datei liegen. Nun wird zuerst geprüft, ob die Datei bereit für den Download ist, dazu dient die Methode `isFileReadyForDownload`. Ist dies der Fall kann nun mit der `getFilePath`-Methode nach dem Pfad der Datei gefragt werden. Dieser wird nun vom `DownloadServlet` genutzt, um die Datei dem Nutzer zu schicken.

Der Vorgang bei dem `StatusServlet` ist sehr Ähnlich. Dort geht es darum in Erfahrung zu bringen, ob eine Download bereit ist, um zum Beispiel zu wissen, ob dem Nutzer bereits ein Download-Button gezeigt werden kann. Der einzige Unterschied liegt darin, dass dort nicht nach dem Pfad gesucht wird, sobndern gleich das Ergebnis der `isFileReadyForDownload` zurückgeschickt wird. Aus diesem Grund wurde darauf verzichtet ein separates Sequenzdiagram dafür zu erstellen.

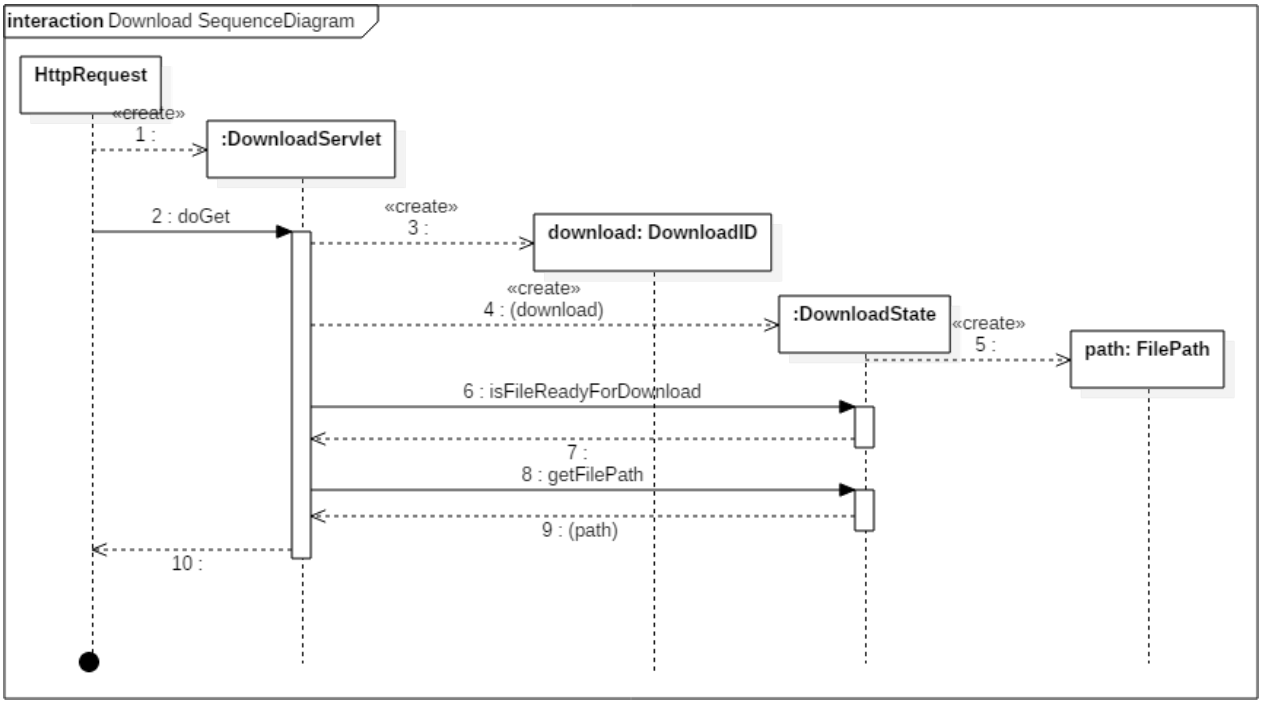


Abb. 2.7: Sequenzdiagramm Download

3 Klassendiagramme

Class Hierarchy

Classes

- java.lang.Object
  - Bridge.JmkbKafkaProducer (in 3.1.1, page 17)
  - Bridge.JmkbMqttConsumer (in 3.1.2, page 18)
  - Bridge.MessageConverter (in 3.1.3, page 20)
  - Bridge.PropertiesFileReader (in 3.1.4, page 21)
  - Bridge.SchemaRegistryConnector (in 3.1.5, page 22)

3.1 Package Bridge

Package Contents *Page*

Classes

3.1.1 Class JmkbKafkaProducer

This class creates a Kafka producer using defined settings and publishes records to the Kafka Cluster.

Declaration

```
public class JmkbKafkaProducer
 extends java.lang.Object
```

Constructor summary

JmkbKafkaProducer() Default constructor

Method summary

disconnect() Disconnects this Kafka producer from the Kafka Cluster and closes the producer.
send(String, byte[]) Asynchronously sends a record to the topic.

Constructors

- **JmkbKafkaProducer**

  ```
  public JmkbKafkaProducer()
  ```

  - **Description**

    Default constructor

Methods

- **disconnect**

  ```
  public void disconnect()
  ```

  - **Description**

    Disconnects this Kafka producer from the Kafka Cluster and closes the producer.

- **send**

  ```
  public void send(java.lang.String topic,byte[] avroMessage)
  ```

  - **Description**

    Asynchronously sends a record to the topic.

  - **Parameters**

    * `topic` – The topic.
    * `avroMessage` – The message to send.

3.1.2 Class JmkbMqttConsumer

This class serves as an MqttClient that consumes messages from the specified FROST-Server address. On message arrival, it will initiate the conversion of the message to a desired format via MqttMessageConverter and supply the converted message to a JmkbKafkaProducer. An instance of this class should be destroyed with a call to the disconnect() method.

Declaration

```
public class JmkbMqttConsumer
 extends java.lang.Object
```

Constructor summary

JmkbMqttConsumer() Default constructor

Method summary

connectionLost(Throwable) This method is called when the connection to the server is lost.
deliveryComplete(IMqttDeliveryToken) Called when delivery for a message has been completed, and all acknowledgments have been received.
disconnect() Disconnects client from MQTT and closes the client.
JmkbMqttConsumer(String, JmkbKafkaProducer) This constructor for this class.
messageArrived(String, MqttMessage) This method is called when a message arrives from the server.

Constructors

- **JmkbMqttConsumer**

  ```
  public JmkbMqttConsumer()
  ```

- **Description**

  Default constructor

Methods

- **connectionLost**

  ```
  public void connectionLost(java.lang.Throwable cause)
  ```

  - **Description**

    This method is called when the connection to the server is lost.

  - **Parameters**

    * `cause` – the reason behind the loss of connection.

- **deliveryComplete**

  ```
  public void deliveryComplete(IMqttDeliveryToken token)
  ```

  - **Description**

    Called when delivery for a message has been completed, and all acknowledgments have been received. In this implementation of this method, nothing happens.

  - **Parameters**

    * `token` – the delivery token associated with the message.

- **disconnect**

  ```
  public void disconnect()
  ```

  - **Description**

    Disconnects client from MQTT and closes the client.

- **JmkbMqttConsumer**

  ```
  public void JmkbMqttConsumer(java.lang.String clientId,
      JmkbKafkaProducer producer)
  ```

  - **Description**

    This constructor for this class. Creates a new MqttClient and subscribes to the topics specified in the SensorThings API standard. A unique identifier and a JmkbKafkaProducer should be supplied.

- **Parameters**
  - `clientId` – The unique identifier for the MqttClient.
  - `producer` – A JmkbKafkaProducer.

- **messageArrived**

```
public void messageArrived(java.lang.String topic,MqttMessage
    message)
```

  - **Description**

    This method is called when a message arrives from the server. This method is invoked synchronously by the MQTT client. An acknowledgment is not sent back to the server until this method returns cleanly. Any additional messages which arrive while this method is running will build up in memory, and will then back up on the network. When this method is called, the supplied message will be converted to an Avro message and forwarded to an instance of JmkbKafkaProducer, which will then send the message to the Kafka Cluster.

  - **Parameters**
    - `topic` – name of the topic on the message was published to
    - `message` – the actual message.

3.1.3 Class MessageConverter

This convenience class provides static methods to convert a given message to another format.

Declaration

```
public class MessageConverter
 extends java.lang.Object
```

Constructor summary

MessageConverter() Default constructor

Method summary

getSensorIdFromMessage(byte[]) This method returns the sensor ID that has supplied the information in the message.

mqttMessageToAvro(MqttMessage) This method converts a given MqttMessage, which contains information in the JSON format, to an Avro message in a byte array.

Constructors

- **MessageConverter**

  ```
  public MessageConverter()
  ```

  - **Description**

    Default constructor

Methods

- **getSensorIdFromMessage**

  ```
  public static java.lang.String getSensorIdFromMessage(byte[] message
      )
  ```

  - **Description**

    This method returns the sensor ID that has supplied the information in the message. In detail, this method searches for the key 'iot.id' in the message and returns the value associated with the key.

  - **Parameters**
    - `message` – The message from which to extract the sensor ID.
  - **Returns** – The sensor ID.

- **mqttMessageToAvro**

  ```
  public static byte[] mqttMessageToAvro(MqttMessage message)
  ```

  - **Description**

    This method converts a given MqttMessage, which contains information in the JSON format, to an Avro message in a byte array.

  - **Parameters**
    - `message` – The message to convert.
  - **Returns** – The message in Avro format.

3.1.4 Class PropertiesFileReader

A class that reads properties from the configuration file (jmkb.properties) and provides a method for getting a property by key.

Declaration

```
public class PropertiesFileReader
 extends java.lang.Object
```

Constructor summary

PropertiesFileReader() Default constructor

Method summary

getProperty(String) Searches for the property with the specified key in jmkb.property.

Constructors

- **PropertiesFileReader**

  ```
  public PropertiesFileReader()
  ```

  - **Description**

    Default constructor

Methods

- **getProperty**

  ```
  public void getProperty(java.lang.String key)
  ```

  - **Description**

    Searches for the property with the specified key in jmkb.property.
  - **Parameters**
    - `key` – The value associated with the key or null if the key is not found.

3.1.5 Class SchemaRegistryConnector

Convenience class which provides methods for interacting with the schema registry.

Declaration

```
public class SchemaRegistryConnector
 extends java.lang.Object
```

Constructor summary

SchemaRegistryConnector() Default constructor

Method summary

getSchemaById(int) Requests the schema associated with the schema ID from the schema registry.

getSchemaBySubject(String) Requests the latest version of the schema associated with the given subject from the schema registry.

getSchemaBySubject(String, int) Requests the given version of the schema associated with the given subject from the schema registry.

Constructors

- **SchemaRegistryConnector**

  ```
  public SchemaRegistryConnector()
  ```

  - **Description**

    Default constructor

Methods

- **getSchemaById**

  ```
  public java.lang.String getSchemaById(int id)
  ```

  - **Description**

    Requests the schema associated with the schema ID from the schema registry. Returns the schema if successful, null if not.

  - **Parameters**

    * `id` – The schema id.

  - **Returns** – The schema if successful, null if not.

- **getSchemaBySubject**

  ```
  public java.lang.String getSchemaBySubject(java.lang.String subject)
  ```

  - **Description**

    Requests the latest version of the schema associated with the given subject from the schema registry. Returns the schema if successful, null if not.

  - **Parameters**

    * `subject` – The subject of the schema.

  - **Returns** – The schema if successful, null if not.

- **getSchemaBySubject**

  ```
  public java.lang.String getSchemaBySubject(java.lang.String subject,
      int version)
  ```

  - **Description**

    Requests the given version of the schema associated with the given subject from the schema registry. Returns the schema if successful, null if not.

  - **Parameters**
    - `subject` – The subject of the schema.
    - `version` – The schema version.
  - **Returns** – the schema if successful, null if not.

Class Hierarchy

Classes

- java.lang.Object
  - CommandRequestPattern.GetClusterCommand (in 3.2.3, page 28)
  - CommandRequestPattern.GetSensorCommand (in 3.2.4, page 29)
  - CommandRequestPattern.GetTileCommand (in 3.2.5, page 30)
  - CommandRequestPattern.Replier (in 3.2.6, page 31)
  - CommandRequestPattern.Requestor (in 3.2.7, page 32)
  - ConfigGUI.JFrame (in 3.3.2, page 34)
    - ConfigGUI.DeleteFrame (in 3.3.1, page 34)
    - ConfigGUI.MainFrame (in 3.3.3, page 35)
    - ConfigGUI.SensorFrame (in 3.3.4, page 35)
  - Controller.ClusterProcessStrategy (in 3.4.1, page 36)
  - Controller.CombinerProcessStrategy (in 3.4.2, page 38)
  - Controller.Controller (in 3.4.3, page 39)
  - Controller.ExportProcessStrategy (in 3.4.4, page 41)
  - Controller.GraphiteProcessStrategy (in 3.4.5, page 43)
  - Controller.TopologyBuilder (in 3.4.6, page 44)
  - Controller.UncaughtExceptionHandler (in 3.4.7, page 46)
  - Properties.PropertiesFile (in 3.5.2, page 49)

Interfaces

- CommandRequestPattern.RequestCommand (in 3.2.1, page 26)
- CommandRequestPattern.StreamProcessingStrategy (in 3.2.2, page 27)
- Properties.PropertiesFileInterface (in 3.5.1, page 47)

3.2 Package CommandRequestPattern

Package Contents *Page*

Interfaces

Classes

3.2.1 Interface RequestCommand

All CommandsRequest implements this Interface. CommandRequest are sendet form the View to request something out of the System.

Declaration

```
public interface RequestCommand
```

All known subinterfaces

GetTileCommand (in 3.2.5, page 30), GetSensorCommand (in 3.2.4, page 29), GetClusterCommand (in 3.2.3, page 28)

All classes known to implement interface

GetTileCommand (in 3.2.5, page 30), GetSensorCommand (in 3.2.4, page 29), GetClusterCommand (in 3.2.3, page 28)

Method summary

execute() This is the Execution form the requested Command
getObject() This Method Return the Requested Object

Methods

- **execute**

  ```
  void execute()
  ```

  - **Description**

    This is the Execution form the requested Command

- **getObject**

```
void getObject()
```

- **Description**

  This Method Return the Requested Object

3.2.2 Interface StreamProcessingStrategy

This Class is a Interface for the Stream Builder Applications which genereates an Output topic to provides data transformations. The ProcessingApplication will use Kafka DSL API to process the data.

Declaration

```
public interface StreamProcessingStrategy
```

Method summary

apply() This Methode definite the Process of the Application.
kafkaStreamClose() This Method is used to explicitly close the Kafka Stream thread.
kafkaStreamStart() This Method is used to explicitly start the Kafka Stream thread.

Methods

- **apply**

  ```
  boolean apply()
  ```

  - **Description**

    This Methode definite the Process of the Application. What Application does specificly.
  - **Returns** – true if the Process got Successfully worked

- **kafkaStreamClose**

  ```
  boolean kafkaStreamClose()
  ```

  - **Description**

    This Method is used to explicitly close the Kafka Stream thread. So that the Processing stops.
  - **Returns** – true if the Kafka Stream closed, false otherwise

- **kafkaStreamStart**

  ```
  boolean kafkaStreamStart()
  ```

- **Description**

  This Method is used to explicitly start the Kafka Stream thread. So that theProcessing need to get started.

- **Returns** – true if the Kafka Stream Started false otherwise

3.2.3 Class GetClusterCommand

This Command request a Cluster in the System.

Declaration

```
public class GetClusterCommand
 extends java.lang.Object implements RequestCommand
```

Constructor summary

GetClusterCommand() Default constructor

Method summary

execute() This is the Execution form the requested Command.
getObject() This Method Return the Requested Cluster as a KStream

Constructors

- **GetClusterCommand**

  ```
  public GetClusterCommand()
  ```

  - **Description**

    Default constructor

Methods

- **execute**

  ```
  public void execute()
  ```

  - **Description**

    This is the Execution form the requested Command. So it will search for the Cluster

- **getObject**

```
public void getObject()
```

- **Description**

  This Method Return the Requested Cluster as a KStream

3.2.4 Class GetSensorCommand

This Command request a Sensor in the System.

Declaration

```
public class GetSensorCommand
 extends java.lang.Object implements RequestCommand
```

Constructor summary

GetSensorCommand() Default constructor

Method summary

execute() This is the Execution form the requested Command.
getObject() This Method Return the Requested Sensor as a KStream

Constructors

- **GetSensorCommand**

```
public GetSensorCommand()
```

  - **Description**

    Default constructor

Methods

- **execute**

```
public void execute()
```

  - **Description**

    This is the Execution form the requested Command. So it will search for the Sensor Uid

- **getObject**

```
public void getObject()
```

- **Description**

  This Method Return the Requested Sensor as a KStream

3.2.5 Class GetTileCommand

This Command request a Tile in the System.

Declaration

```
public class GetTileCommand
 extends java.lang.Object implements RequestCommand
```

Constructor summary

GetTileCommand() Default constructor

Method summary

execute() This is the Execution form the requested Command.
getObject() This Method Return the Requested Tile as a KStream

Constructors

- **GetTileCommand**

  ```
  public GetTileCommand()
  ```

  - **Description**

    Default constructor

Methods

- **execute**

  ```
  public void execute()
  ```

  - **Description**

    This is the Execution form the requested Command. So it will search for the Tile

- **getObject**

  ```
  public void getObject()
  ```

  - **Description**

    This Method Return the Requested Tile as a KStream

3.2.6 Class Replier

This Class handels the Requests and Replies to them

Declaration

```
public class Replier
 extends java.lang.Object
```

Constructor summary

Replier() Default constructor

Method summary

initialize(Connection, String) This is the initialisation Method for the Replier to connect to different Requestors

onMessage(Message, RequestCommand) This Methode triggers something in the System waht has to be done

Constructors

- **Replier**

  ```
  public Replier()
  ```

  - **Description**

    Default constructor

Methods

- **initialize**

  ```
  public void initialize(Connection connection,java.lang.String
      requestQueueName)
  ```

  - **Description**

    This is the initialisation Method for the Replier to connect to different Requestors

  - **Parameters**

    * `connection` – This is the Connection parameter, so taht the replier knows where he answers
    * `requestQueueName` – This a Simple name for the request Queue

- **onMessage**

  ```
  public void onMessage(Message message,RequestCommand request)
  ```

  - **Description**

    This Methode triggers something in the System waht has to be done

  - **Parameters**

    * `message` – This is a simple Message parameter
    * `request` – This is the RequestCommand Object wich Contains the Real request.

3.2.7 Class Requestor

The Implemente this class and request something to the System and a Replier answer to it.

Declaration

```
public class Requestor
 extends java.lang.Object
```

Constructor summary

Requestor() Default constructor

Method summary

initialize(Connection)
receiveSync(RequestCommand) This Methode is there to got the Request again when it get lost or something
send(RequestCommand)

Constructors

- **Requestor**

  ```
  public Requestor()
  ```

  - **Description**

    Default constructor

Methods

- **initialize**

  ```
  public void initialize(Connection connection)
  ```

  - **Parameters**
    - `connection` – This is the Connection parameter, so taht the repuestor knows where he requests something

- **receiveSync**

  ```
  public RequestCommand receiveSync(RequestCommand request)
  ```

  - **Description**

    This Methode is there to got the Request again when it get lost or something
  - **Parameters**
    - `request` – It Returns the Requested RequestCommand
  - **Returns** – A RequestCommand which contains a Request for a RequestCommand

- **send**

  ```
  public boolean send(RequestCommand request)
  ```

  - **Parameters**
    - `request` – This is the RequestCommand Object wich Conntains the Real request.
  - **Returns** – true if the RequestCommand got send and false otherwise

3.3 Package ConfigGUI

Package Contents *Page*

Classes

3.3.1 Class DeleteFrame

This Frame is the Delete Frame, where you delete Topics out of the Programm

Declaration

```
public class DeleteFrame
 extends ConfigGUI.JFrame
```

Constructor summary

DeleteFrame() Default constructor

Constructors

- **DeleteFrame**

  ```
  public DeleteFrame()
  ```

  - **Description**

    Default constructor

3.3.2 Class JFrame

This is the Basic Interface from Java for building a Frame.

Declaration

```
public class JFrame
 extends java.lang.Object
```

All known subclasses

SensorFrame (in 3.3.4, page 35), MainFrame (in 3.3.3, page 35), DeleteFrame (in 3.3.1, page 34)

Constructor summary

JFrame() Default constructor

Constructors

- **JFrame**

  ```
  public JFrame()
  ```

- **Description**

  Default constructor

3.3.3 Class MainFrame

This Class holds the main functionality of the PaVoS program. It starts/stops the whole System and manages the export/import.

Declaration

```
public class MainFrame
 extends ConfigGUI.JFrame
```

Constructor summary

MainFrame() Default constructor

Constructors

- **MainFrame**

  ```
  public MainFrame()
  ```

  - **Description**

    Default constructor

3.3.4 Class SensorFrame

This Frame hold the data of all possible Sensors in the System.

Declaration

```
public class SensorFrame
 extends ConfigGUI.JFrame
```

Constructor summary

SensorFrame() Default constructor

Constructors

- **SensorFrame**

  ```
  public SensorFrame()
  ```

– **Description**

Default constructor

3.4 Package Controller

Package Contents *Page*

Classes

3.4.1 Class ClusterProcessStrategy

This Class is for the generation of the Clusters for the View. It Generates a Cluster Outputtopic

Declaration

```
public class ClusterProcessStrategy
 extends java.lang.Object
```

Constructor summary

ClusterProcessStrategy() Default constructor

Method summary

apply() This Methode definite the Process of the Application.

kafkaStreamClose() This Method is used to explicitly close the Kafka Stream thread.
kafkaStreamStart() This Method is used to explicitly start the Kafka Stream thread.

Constructors

- **ClusterProcessStrategy**

  ```
  public ClusterProcessStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **apply**

  ```
  public boolean apply()
  ```

  - **Description**

    This Methode definite the Process of the Application. What Application does specificly.
  - **Returns** – true if the Cluster Process got Successfully worked, false otherwise

- **kafkaStreamClose**

  ```
  public boolean kafkaStreamClose()
  ```

  - **Description**

    This Method is used to explicitly close the Kafka Stream thread. So that the Processing stops.
  - **Returns** – true if the Kafka Stream closed false otherwise

- **kafkaStreamStart**

  ```
  public boolean kafkaStreamStart()
  ```

  - **Description**

    This Method is used to explicitly start the Kafka Stream thread. So that theProcessing need to get started.
  - **Returns** – true if the Kafka Stream Started, false otherwise

3.4.2 Class CombinerProcessStrategy

This Class does combinate the Clusters to bigger Cluster for the Different Zoom Levels

Declaration

```
public class CombinerProcessStrategy
 extends java.lang.Object
```

Constructor summary

CombinerProcessStrategy() Default constructor

Method summary

apply() This Methode definite the Process of the Application.
kafkaStreamClose() This Method is used to explicitly close the Kafka Stream thread.
kafkaStreamStart() This Method is used to explicitly start the Kafka Stream thread.

Constructors

- **CombinerProcessStrategy**

  ```
  public CombinerProcessStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **apply**

  ```
  public boolean apply()
  ```

  - **Description**

    This Methode definite the Process of the Application. What Application does specificly.
  - **Returns** – true if the Combiner Process got Successfully worked

- **kafkaStreamClose**

  ```
  public boolean kafkaStreamClose()
  ```

- **Description**

  This Method is used to explicitly close the Kafka Stream thread. So that the Processing stops.

- **Returns** – true if the Kafka Stream closed, false otherwise

- **kafkaStreamStart**

```
public boolean kafkaStreamStart()
```

  - **Description**

    This Method is used to explicitly start the Kafka Stream thread. So that theProcessing need to get started.

  - **Returns** – true if the Kafka Stream Started false otherwise

3.4.3 Class Controller

This Class is the ControllerClass which manages the Requests and start new TopologyBuilders to start new Processing Application.

Declaration

```
public class Controller
 extends java.lang.Object
```

Constructor summary

Controller() Default constructor

Method summary

generateOutputtopic(String) This Method generates a Output Topic, which uses a ProcessApplikation as OutputSink.
init() This Method initialise the Controler
setProperties(PropertiesFileInterface) This Method sets the Properties File
setTopolgyBuilder(StreamProcessingStrategy, String, String) Thsi Method starts a TopolgyBuilder to start a Kafka Stream Process.
subscribe(String) This method subscribe the controller to the Input Kafka Stream
workRequest(RequestCommand) This Method process the single Reuqest form the View

Constructors

- **Controller**

  ```
  public Controller()
  ```

  - **Description**

    Default constructor

Methods

- **generateOutputtopic**

  ```
  public boolean generateOutputtopic(java.lang.String topic)
  ```

  - **Description**

    This Method generates a Output Topic, which uses a ProcessApplikation as OutputSink. This will use Apache Avro Format.
  - **Parameters**
    - `topic` – topic name of the new Topic in Kafka
  - **Returns** – true when the Output Topic got successful generated

- **init**

  ```
  public boolean init()
  ```

  - **Description**

    This Method initialise the Controler
  - **Returns** – true when the initialise was successful and false otherwise

- **setProperties**

  ```
  public void setProperties(PropertiesFileInterface props)
  ```

  - **Description**

    This Method sets the Properties File
  - **Parameters**
    - `props` – props is the Propertyfile form where the controller reads his Settings

- **setTopolgyBuilder**

```
public void setTopolgyBuilder(StreamProcessingStrategy process,java.
    lang.String inputTopic,java.lang.String outputTopic)
```

  - **Description**

    Thsi Method starts a TopolgyBuilder to start a Kafka Stream Process.

  - **Parameters**

    * `process` – process name of the Process Application
    * `inputTopic` – inputTopic of the Kafka Topic
    * `outputTopic` – outputTopic of the Kafka Topic

- **subscribe**

```
public void subscribe(java.lang.String topic)
```

  - **Description**

    This method subscribe the controller to the Input Kafka Stream

  - **Parameters**

    * `topic` – The Name of the Topic which you want to subscribe

- **workRequest**

```
public void workRequest(RequestCommand command)
```

  - **Description**

    This Method process the single Reuqest form the View

  - **Parameters**

    * `command` – command is Instance of the RequestCommand Interface which contains a Job Request

3.4.4 Class ExportProcessStrategy

This Class is for The Processing of the Export Stream and it generates a Output Stream

Declaration

```
public class ExportProcessStrategy
 extends java.lang.Object
```

Constructor summary

ExportProcessStrategy() Default constructor

Method summary

apply() This Methode definite the Process of the Application.
ExportApplication(ExportProperties) This is the default Contructer for the Export Process
kafkaStreamClose() This Method is used to explicitly close the Kafka Stream thread.
kafkaStreamStart() This Method is used to explicitly start the Kafka Stream thread.

Constructors

- **ExportProcessStrategy**

  ```
  public ExportProcessStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **apply**

  ```
  public boolean apply()
  ```

  - **Description**

    This Methode definite the Process of the Application. What Application does specificly.
  - **Returns** – true if the Export Process got Successfully worked.

- **ExportApplication**

  ```
  public void ExportApplication(ExportProperties props)
  ```

  - **Description**

    This is the default Contructer for the Export Process
  - **Parameters**
    - `props` – ExportProperties is the Properties Object for the Application

- **kafkaStreamClose**

  ```
  public boolean kafkaStreamClose()
  ```

- **Description**

  This Method is used to explicitly close the Kafka Stream thread. So that the Processing stops.

- **Returns** – true if the Kafka Stream Started false otherwise

- **kafkaStreamStart**

```
public boolean kafkaStreamStart()
```

  - **Description**

    This Method is used to explicitly start the Kafka Stream thread. So that theProcessing need to get started.

  - **Returns** – true if the Kafka Stream Started false otherwise

3.4.5 Class GraphiteProcessStrategy

This Class is for The Processing of the Data for Graphite, to represente the Sensors. It Generates a Graphite Output Stream

Declaration

```
public class GraphiteProcessStrategy
 extends java.lang.Object
```

Constructor summary

GraphiteProcessStrategy() Default constructor

Method summary

apply() This Methode definite the Process of the Application.
kafkaStreamClose() This Method is used to explicitly close the Kafka Stream thread.
kafkaStreamStart() This Method is used to explicitly start the Kafka Stream thread.

Constructors

- **GraphiteProcessStrategy**

```
public GraphiteProcessStrategy()
```

  - **Description**

    Default constructor

Methods

- **apply**

  ```
  public boolean apply()
  ```

  - **Description**

    This Methode definite the Process of the Application. What Application does specificly.
  - **Returns** – true if the Graphite Process got Successfully worked

- **kafkaStreamClose**

  ```
  public boolean kafkaStreamClose()
  ```

  - **Description**

    This Method is used to explicitly close the Kafka Stream thread. So that the Processing stops.
  - **Returns** – true if the Kafka Stream closed, false otherwise

- **kafkaStreamStart**

  ```
  public boolean kafkaStreamStart()
  ```

  - **Description**

    This Method is used to explicitly start the Kafka Stream thread. So that theProcessing need to get started.
  - **Returns** – true if the Kafka Stream Started false otherwise

3.4.6 Class TopologyBuilder

A component that is used to build a ProcessorTopology. A topology contains an acyclic graph of sources, processors, and sinks. A source is a node in the graph that consumes one or more Kafka topics and forwards them to its child nodes. A processor is a node in the graph that receives input records from upstream nodes, processes that records, and optionally forwarding new records to one or all of its children. Finally, a sink is a node in the graph that receives records from upstream nodes and writes them to a Kafka topic. This builder allows you to construct an acyclic graph of these nodes, and the builder is then passed into a new KafkaStreams instance that will then begin consuming, processing, and producing records

Declaration

```
public class TopologyBuilder
 extends java.lang.Object
```

Constructor summary

TopologyBuilder() Default constructor

Method summary

addProcessor(String, StreamProcessingStrategy, String) Add a new processor node that receives and processes records output by one or more parent source or processor node.

addSink(String, String) Add a new sink that forwards records from upstream parent processor and/or source nodes to the named Kafka topic.

addSource(String, topic) Add a new source that consumes from topics matching the given pattern and forward the records to child processor and/or sink nodes.

Constructors

- **TopologyBuilder**

  ```
  public TopologyBuilder()
  ```

  - **Description**

    Default constructor

Methods

- **addProcessor**

  ```
  public void addProcessor(java.lang.String name,
    StreamProcessingStrategy supplier,java.lang.String input)
  ```

  - **Description**

    Add a new processor node that receives and processes records output by one or more parent source or processor node.

  - **Parameters**

    * `name` – is the name of the Processor Stratgie
    * `supplier` – supplier is the supplier of the Process instant to generate more then 1 Process
    * `input` – input Topic Stream name

- **addSink**

  ```
  public void addSink(java.lang.String name,java.lang.String topic)
  ```

  - **Description**

    Add a new sink that forwards records from upstream parent processor and/or source nodes to the named Kafka topic.

  - **Parameters**

    * `name` – name of the Sink
    * `topic` – name of the Topic Stream

- **addSource**

  ```
  public void addSource(java.lang.String name,topic topicPattern)
  ```

  - **Description**

    Add a new source that consumes from topics matching the given pattern and forward the records to child processor and/or sink nodes.

  - **Parameters**

    * `name` – name of the Input Topic Stream
    * `topicPattern` – topicPattern is a Pattern to filter the data from the Input Topic Stream

3.4.7 Class UncaughtExceptionHandler

To catch any unexpected exceptions, you can set before you start the application. This handler is called whenever a stream thread is terminated by an unexpected exception.

Declaration

```
public class UncaughtExceptionHandler
 extends java.lang.Object
```

Constructor summary

UncaughtExceptionHandler() Default constructor

Method summary

getMessage() Returns the detail message string of this throwable.

Constructors

- **UncaughtExceptionHandler**

  ```
  public UncaughtExceptionHandler()
  ```

  - **Description**

    Default constructor

Methods

- **getMessage**

  ```
  public java.lang.String getMessage()
  ```

  - **Description**

    Returns the detail message string of this throwable.
  - **Returns** – String with the error Message

3.5 Package Properties

Package Contents *Page*

Interfaces

Classes

3.5.1 Interface PropertiesFileInterface

The Properties Interface is a special form of associative memory in which key-value pairs are always of type string. Since the entries can be stored in a file and read out again, hardwired character strings can be externalized from the program text so that the values can be easily changed without retranslation.

Declaration

```
public interface PropertiesFileInterface
```

All known subinterfaces

PropertiesFile (in 3.5.2, page 49)

All classes known to implement interface

PropertiesFile (in 3.5.2, page 49)

Method summary

getPropValues(String) This Methodes returns the requestet propertie Value
putProperty(String, String) The Method adds a key-value pair to the Properties object.
save(boolean) This Method saves the PropertiesFile with the Option to do a Backup of the File

Methods

- **getPropValues**

  ```
  java.lang.String getPropValues(java.lang.String propertyName)
  ```

  - **Description**

    This Methodes returns the requestet propertie Value

  - **Parameters**

    * `propertyName` – propertyName is the name of the Requested Property

  - **Returns** – Return the Value to the Requested Property

- **putProperty**

  ```
  boolean putProperty(java.lang.String propertyName,java.lang.String
      propertyValue)
  ```

  - **Description**

    The Method adds a key-value pair to the Properties object. To get back to the value later, is called with the key and then return

  - **Parameters**

    * `propertyName` – propertyName is the Name of the Property which you want to edit
    * `propertyValue` – propertyValue is the Value of the Property which you want to edit

  - **Returns** – true wenn the property got set false otherwise

- **save**

  ```
  boolean save(boolean makeBackup)
  ```

  - **Description**

    This Method saves the PropertiesFile with the Option to do a Backup of the File
  - **Parameters**
    * `makeBackup` – true if you want to make a Bachup
  - **Returns** – true when the file got saved, false otherwise

3.5.2 Class PropertiesFile

The Properties class is a special form of associative memory in which key-value pairs are always of type string. Since the entries can be stored in a file and read out again, hardwired character strings can be externalized from the program text so that the values can be easily changed without retranslation.

Declaration

```
public class PropertiesFile
 extends java.lang.Object implements PropertiesFileInterface
```

Constructor summary

PropertiesFile() Default constructor

Method summary

getPropValues(String) This Methodes returns the requestet propertie Value
putProperty(String, String) The Method adds a key-value pair to the Properties object.
save(boolean) This Method saves the PropertiesFile with the Option to do a Backup of the File

Constructors

- **PropertiesFile**

  ```
  public PropertiesFile()
  ```

  - **Description**

    Default constructor

Methods

- **getPropValues**

  ```
  public java.lang.String getPropValues(java.lang.String propertyName)
  ```

  - **Description**

    This Methodes returns the requestet propertie Value

  - **Parameters**
    - `propertyName` – propertyName is the name of the Requested Property
  - **Returns** – Return the Value to the Requested Property

- **putProperty**

  ```
  public boolean putProperty(java.lang.String propertyName,java.lang.
      String propertyValue)
  ```

  - **Description**

    The Method adds a key-value pair to the Properties object. To get back to the value later, is called with the key and then return

  - **Parameters**
    - `propertyName` – propertyName is the Name of the Property which you want to edit
    - `propertyValue` – propertyValue is the Value of the Property which you want to edit
  - **Returns** – true wenn the property got set false otherwise

- **save**

  ```
  public boolean save(boolean makeBackup)
  ```

  - **Description**

    This Method saves the PropertiesFile with the Option to do a Backup of the File

  - **Parameters**
    - `makeBackup` – true if you want to make a Bachup
  - **Returns** – true when the file got saved, false otherwise

Class Hierarchy

Classes

- java.lang.Object
  - Import.CSVReaderStrategy (in 3.6.2, page 52)
  - Import.DataImporter (in 3.6.3, page 54)
  - Import.FileImporter (in 3.6.4, page 55)
  - Import.FrostSender (in 3.6.5, page 56)
  - Import.NetCDFReaderStrategy (in 3.6.6, page 57)
  - Import.ReaderType (in 3.6.7, page 59)

Interfaces

- Import.FileReaderStrategy (in 3.6.1, page 51)

3.6 Package Import

Package Contents *Page*

Interfaces

Classes

3.6.1 Interface FileReaderStrategy

Interface for the FileReaderStrategy classes. Realization of a Strategy to be able to swap out the way a File has to be read.

Declaration

```
public interface FileReaderStrategy
```

All known subinterfaces

NetCDFReaderStrategy (in 3.6.6, page 57), CSVReaderStrategy (in 3.6.2, page 52)

All classes known to implement interface

NetCDFReaderStrategy (in 3.6.6, page 57), CSVReaderStrategy (in 3.6.2, page 52)

Method summary

sendFileData(FilePath, FrostSender) Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

Methods

- **sendFileData**

  ```
  void sendFileData(FilePath path,FrostSender froster)
  ```

  - **Description**

    Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

  - **Parameters**

    * `path` – Is the FilePath of the File to Import.
    * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

3.6.2 Class CSVReaderStrategy

Implementation of the FileReaderStrategy interface for CSV files.

Declaration

```
public class CSVReaderStrategy
 extends java.lang.Object implements FileReaderStrategy
```

Constructor summary

CSVReaderStrategy() Default constructor

Method summary

sendFileData(FilePath, FrostSender) Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

sendFileData(FilePath, FrostSender) Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

Constructors

- **CSVReaderStrategy**

  ```
  public CSVReaderStrategy()
  ```

  - **Description**

    Default constructor

Methods

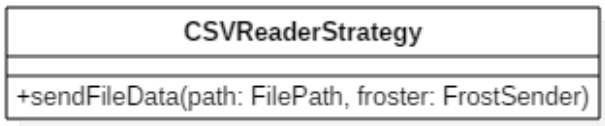
- **sendFileData**

  ```
  public void sendFileData(FilePath path,FrostSender froster)
  ```

  - **Description**

    Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

  - **Parameters**



    * `path` – Is the FilePath of the File to Import.
    * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

- **sendFileData**

```
public void sendFileData(FilePath path,FrostSender froster)
```

  - **Description**

    Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

  - **Parameters**
    * `path` – Is the FilePath of the File to Import.
    * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

3.6.3 Class DataImporter

Importer for data that should be added to PaVoS. Import takes place for files in a specified folder of the server.

Declaration

```
public class DataImporter
 extends java.lang.Object
```

Constructor summary

DataImporter() Default constructor

Method summary

startImportingFileData() Checks for files in the specified import folder and opens a new thread for each of them, where a FileImporter is started to import the contained data.

Constructors

- **DataImporter**

```
public DataImporter()
```

  - **Description**

    Default constructor

Methods

- **startImportingFileData**

```
public void startImportingFileData()
```

  - **Description**

    Checks for files in the specified import folder and opens a new thread for each of them, where a FileImporter is started to import the contained data.

3.6.4 Class FileImporter

Importer for the Data contained in a File. Takes the Data and sends them to the FROST-Server.

Declaration

```
public class FileImporter
 extends java.lang.Object
```

Constructor summary

FileImporter() Default constructor

Method summary

addFileData(FilePath, FrostSender) Adds the Data of a File at a specified FilePath to the FROST-Server.

Constructors

- **FileImporter**

```
public FileImporter()
```

  - **Description**

    Default constructor

Methods

- **addFileData**

```
public void addFileData(FilePath path,FrostSender froster)
```

  - **Description**

    Adds the Data of a File at a specified FilePath to the FROST-Server. To do so, the FileExtension of the File is determined.With help of the readerTypeClass the matching implementation of the FileReaderStrategy interface for the FileExtension is generated and can be used to get the Data from then File.

  - **Parameters**

    * `path` – Is the FilePath of the File to Import.
    * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

3.6.5 Class FrostSender

sends Data to the FROST-Server.

Declaration

```
public class FrostSender
 extends java.lang.Object
```

Constructor summary

FrostSender() Default constructor

Method summary

sendToFrostServer(JsonObject) Sends the given JsonObject to the FROST-Server.

Constructors

- **FrostSender**

```
public FrostSender()
```

  - **Description**

    Default constructor

Methods

- **sendToFrostServer**

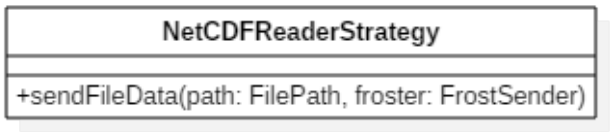
```
public void sendToFrostServer(JsonObject json)
```

  - **Description**

    Sends the given JsonObject to the FROST-Server.
  - **Parameters**
    - `json` – Represents a single ObservedProperty.

3.6.6 Class NetCDFReaderStrategy

Implementation of the FileReaderStrategy interface for NetCDF files.



Declaration

```
public class NetCDFReaderStrategy
 extends java.lang.Object implements FileReaderStrategy
```

Constructor summary

NetCDFReaderStrategy() Default constructor

Method summary

sendFileData(FilePath, FrostSender) Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

sendFileData(FilePath, FrostSender) Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

Constructors

- **NetCDFReaderStrategy**

  ```
  public NetCDFReaderStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **sendFileData**

  ```
  public void sendFileData(FilePath path,FrostSender froster)
  ```

  - **Description**

    Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

  - **Parameters**

    * `path` – Is the FilePath of the File to Import.
    * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

- **sendFileData**

  ```
  public void sendFileData(FilePath path,FrostSender froster)
  ```

  - **Description**

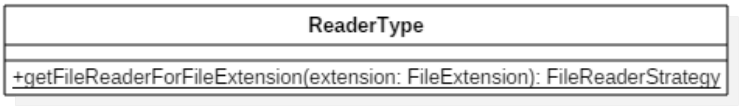
    Reades from a File as specified by the FilePath and sends the information in it to the FROST-Server using the FrostSender that was provided.

  - **Parameters**

    * `path` – Is the FilePath of the File to Import.
    * `froster` – Is the FrostSender instance that will be used to send the files data to the Frost-Server.

3.6.7 Class ReaderType

Is like a chooser for the right FileReaderStrategy. If a new Strategy is added, this class needs some changes to use the new Strategy.



Declaration

```
public class ReaderType
 extends java.lang.Object
```

Constructor summary

ReaderType() Default constructor

Method summary

getFileReaderForFileExtension(FileExtension) Gives a new Instance of a FileReaderStrategy for the specified FileExtension.

Constructors

- **ReaderType**

  ```
  public ReaderType()
  ```

  - **Description**

    Default constructor

Methods

- **getFileReaderForFileExtension**

  ```
  public static FileReaderStrategy getFileReaderForFileExtension(
    FileExtension extension)
  ```

  - **Description**

    Gives a new Instance of a FileReaderStrategy for the specified FileExtension.

  - **Parameters**

    * `extension` – is the FileExtension for which a FileReaderStrategy has to be generated.
- **Returns** – An instance of an implementation of the FileReaderStrategy interface.

Class Hierarchy

Classes

- java.lang.Object
  - DatabaseConnection.ClusterID (in 3.7.1, page 62)
  - DatabaseConnection.Facade (in 3.7.3, page 63)
  - DatabaseConnection.HttpServlet (in 3.7.5, page 67)
    - DatabaseConnection.GridDataServlet (in 3.7.4, page 65)
    - DatabaseConnection.SensorListServlet (in 3.7.9, page 71)
  - DatabaseConnection.KafkaToStorageProcessor (in 3.7.6, page 68)
  - DatabaseConnection.Maintainer (in 3.7.7, page 69)
    - DatabaseConnection.DataMaintainer (in 3.7.2, page 62)
    - DatabaseConnection.SensorMaintainer (in 3.7.10, page 72)
  - DatabaseConnection.MaintenanceManager (in 3.7.8, page 70)
  - DatabaseConnection.ZoomLevel (in 3.7.11, page 73)

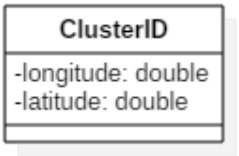
3.7 Package DatabaseConnection

Package Contents *Page*

Classes

3.7.1 Class ClusterID

This class describes a unique identification of a cluster via longitude and latitude.



Declaration

```
public class ClusterID
 extends java.lang.Object
```

Constructor summary

ClusterID() Default constructor

Constructors

- **ClusterID**

  ```
  public ClusterID()
  ```

  - **Description**

    Default constructor

3.7.2 Class DataMaintainer

This class maintains the sensordata in the StorageSolution.

Declaration

```
public class DataMaintainer
 extends DatabaseConnection.Maintainer
```

Constructor summary

DataMaintainer() Default constructor

Method summary

summarize(TimeUnit) This method takes data of a certain TimeUnit and summarizes it into the next higher TimeUnit.

Constructors

- **DataMaintainer**

  ```
  public DataMaintainer()
  ```

  - **Description**

    Default constructor

Methods

- **summarize**

  ```
  public void summarize(TimeUnit timeUnit)
  ```

  - **Description**

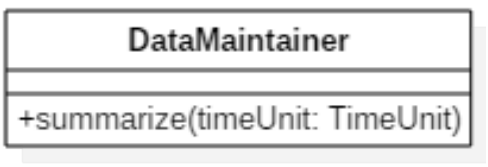
    This method takes data of a certain TimeUnit and summarizes it into the next higher TimeUnit. The summarized data is then saved back into the StorageSolution. The original data of the lower TimeUnit is then deleted from the database.

  - **Parameters**

    * `timeUnit` – The TimeUnit to summarize.

3.7.3 Class Facade

A facade to simplify access to a StorageSolution, such as a database. Through the methods, data can be inserted into the StorageSolution and certain information about its content requested.



Declaration

```
public class Facade
 extends java.lang.Object
```

Constructor summary

Facade() Default constructor

Method summary

getGrid(ClusterID[], ZoomLevel, Time) Returns an appropriate grid of clusters in the requested grid section for the specified ZoomLevel and time.
getSensors(ObservationType, ClusterID) Fetches all sensors from the given cluster that observe the given ObservedProperty and returns an array of sensors.
subscribeToZoomLevelStream(KStream) Subscribes to the given KafkaStream, which contains ZoomLevel-specific data and initiates processing of its records.

Constructors

- **Facade**

  ```
  public Facade()
  ```

  - **Description**

    Default constructor

Methods

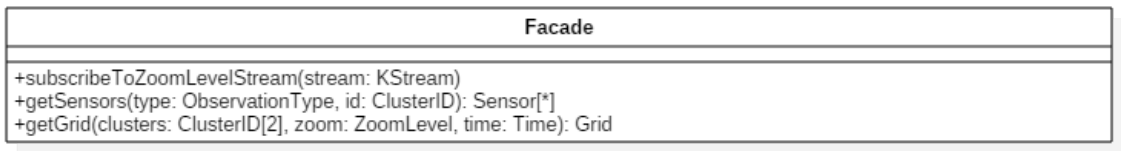
- **getGrid**

  ```
  public Grid getGrid(ClusterID[] clusters,ZoomLevel zoom,Time time)
  ```

  - **Description**

    Returns an appropriate grid of clusters in the requested grid section for the specified ZoomLevel and time. The (first) two values of the ClusterID array define the grid section from which to get the data.

  - **Parameters**



    * `clusters` – An array of ClusterIDs from which the first two entries are taken to compute the section of the Grid to get the data from.
    * `zoom` – The ZoomLevel from which to get the data.
    * `time` – The point in time.
  - **Returns** – A grid with the computed data.

- **getSensors**

  ```
  public java.util.Set getSensors(ObservationType type,ClusterID id)
  ```

  - **Description**

    Fetches all sensors from the given cluster that observe the given ObservedProperty and returns an array of sensors.
  - **Parameters**
    * `type` – The ObservationType of the requested sensors.
    * `id` – The ID of the cluster.
  - **Returns** – An array of sensors.

- **subscribeToZoomLevelStream**

  ```
  public void subscribeToZoomLevelStream(KStream stream)
  ```

  - **Description**

    Subscribes to the given KafkaStream, which contains ZoomLevel-specific data and initiates processing of its records.
  - **Parameters**
    * `stream` – The stream to subscribe to.

3.7.4 Class GridDataServlet

An HTTPServlet for requesting Grid data.

Declaration

```
public class GridDataServlet
 extends DatabaseConnection.HttpServlet
```

Constructor summary

GridDataServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) This method calls the getGrid method of the Facade to get a Grid of clusters at a certain ZoomLevel and Time .

Constructors

- **GridDataServlet**

  ```
  public GridDataServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```

  - **Description**

    This method calls the getGrid method of the Facade to get a Grid of clusters at a certain ZoomLevel and Time . This saves the Grid into res.

  - **Parameters**

    * `req` – An HttpServletRequest object that contains the request the client has made of the servlet.
    * `res` – An HttpServletResponse object that contains the response the servlet sends to the client.

Members inherited from class HttpServlet

`DatabaseConnection.HttpServlet` (in 3.7.5, page 67)

- `public void` **doGet**`(HttpServletRequest req, HttpServletResponse res)`

3.7.5 Class HttpServlet

An abstract HTTPServlet.

Declaration

```
public class HttpServlet
 extends java.lang.Object
```

All known subclasses

SensorListServlet (in 3.7.9, page 71), GridDataServlet (in 3.7.4, page 65)

Constructor summary

HttpServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Called by the server (via the service method) to allow a servlet to handle a GET request.

Constructors

- **HttpServlet**

  ```
  public HttpServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```

- **Description**

  Called by the server (via the service method) to allow a servlet to handle a GET request.

- **Parameters**

  * `req` – An HttpServletRequest object that contains the request the client has made of the servlet.
  * `res` – An HttpServletResponse object that contains the response the servlet sends to the client.

3.7.6 Class KafkaToStorageProcessor

This class converts KafkaStream records to data that can be inserted into the StorageSolution.

Declaration

```
public class KafkaToStorageProcessor
 extends java.lang.Object
```

Constructor summary

KafkaToStorageProcessor() Default constructor

Method summary

subscribe(KStream) Subscribes to the given KafkaStream and converts the data to the appropriate format for the StorageSolution.

Constructors

- **KafkaToStorageProcessor**

  ```
  public KafkaToStorageProcessor()
  ```

  - **Description**

    Default constructor

Methods

- **subscribe**

  ```
  public void subscribe(KStream stream)
  ```

  - **Description**

    Subscribes to the given KafkaStream and converts the data to the appropriate format for the StorageSolution. If a stream is already subscribed to, unsubscribes from the old stream and subscribes to the new one.

  - **Parameters**

    * `stream` – The KStream to subscribe to.

3.7.7 Class Maintainer

An abstract class describing a Maintainer, which performs maintenance on certain data in the StorageSolution.

Declaration

```
public class Maintainer
 extends java.lang.Object
```

All known subclasses

SensorMaintainer (in 3.7.10, page 72), DataMaintainer (in 3.7.2, page 62)

Constructor summary

Maintainer() Default constructor

Constructors

- **Maintainer**

  ```
  public Maintainer()
  ```

  - **Description**

    Default constructor

3.7.8 Class MaintenanceManager

This class manages the way the methods of Maintainers are called to make sure the StorageSolution content is maintained.

Declaration

```
public class MaintenanceManager
 extends java.lang.Object
```

Constructor summary

MaintenanceManager() Default constructor

Method summary

startMaintenance() This method should be called as soon as the database is started.

Constructors

- **MaintenanceManager**

  ```
  public MaintenanceManager()
  ```

  - **Description**

    Default constructor

Methods

- **startMaintenance**

```
public void startMaintenance()
```

  - **Description**

    This method should be called as soon as the database is started. Through calls to instances of Maintainers, summarizes data in the database and deletes data that has become obsolete as a result of the summarization.

3.7.9 Class SensorListServlet

An HTTPServlet for requesting a list of sensors.

Declaration

```
public class SensorListServlet
 extends DatabaseConnection.HttpServlet
```

Constructor summary

SensorListServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) This method calls the getSensors method of the Facade to get a list of Sensors that are in a certain cluster.

Constructors

- **SensorListServlet**

```
public SensorListServlet()
```

  - **Description**

    Default constructor

Methods

- **doGet**

```
public void doGet(HttpServletRequest req,HttpServletResponse res)
```

  - **Description**

    This method calls the getSensors method of the Facade to get a list of Sensors that are in a certain cluster.

  - **Parameters**

    * `req` – An HttpServletRequest object that contains the request the client has made of the servlet.
    * `res` – An HttpServletResponse object that contains the response the servlet sends to the client.

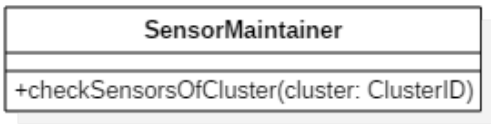
Members inherited from class HttpServlet

`DatabaseConnection.HttpServlet` (in 3.7.5, page 67)

- `public void` **doGet**`(HttpServletRequest` **req**`, HttpServletResponse` **res**`)`

3.7.10 Class SensorMaintainer

This class maintains the list of sensors saved in the StorageSolution.



Declaration

```
public class SensorMaintainer
 extends DatabaseConnection.Maintainer
```

Constructor summary

SensorMaintainer() Default constructor

Method summary

checkSensorsOfCluster(ClusterID) This method checks if the sensors registered to the given cluster are up to date.

Constructors

- **SensorMaintainer**

  ```
  public SensorMaintainer()
  ```

  - **Description**

    Default constructor

Methods

- **checkSensorsOfCluster**

  ```
  public void checkSensorsOfCluster(ClusterID cluster)
  ```

  - **Description**

    This method checks if the sensors registered to the given cluster are up to date. A sensor is up to date if data has been received from it in the last 24 hours. If this requirement is not met, the sensor is deleted from the database.

  - **Parameters**

    * `cluster` – The cluster to check.

3.7.11 Class ZoomLevel

This class describes a zoom level for the map.

Declaration

```
public class ZoomLevel
 extends java.lang.Object
```

Constructor summary

ZoomLevel() Default constructor

Constructors

- **ZoomLevel**

  ```
  public ZoomLevel()
  ```

  - **Description**
    Default constructor

Class Hierarchy

Classes

- java.lang.Object
    - DataTransferControl.Collection (in 3.8.1, page 76)
    - DataTransferControl.Config (in 3.8.2, page 76)
        - DataTransferControl.GraphiteConfig (in 3.8.7, page 81)
    - DataTransferControl.Consumer (in 3.8.3, page 78)
        - DataTransferControl.KafkaToGraphiteConsumer (in 3.8.10, page 84)
    - DataTransferControl.ConsumerRecord (in 3.8.4, page 79)
    - DataTransferControl.ConsumerRecords (in 3.8.5, page 79)
    - DataTransferControl.GraphDataTransferController (in 3.8.6, page 80)
    - DataTransferControl.KafkaConsumer (in 3.8.9, page 83)
    - DataTransferControl.Properties (in 3.8.11, page 85)
    - DataTransferControl.Sender (in 3.8.12, page 86)
        - DataTransferControl.GraphiteSender (in 3.8.8, page 82)
    - DataTransferControl.SerializationDeserialization.KafkaObservationData (in 3.9.1, page 88)
    - DataTransferControl.SerializationDeserialization.ObservationDataDeserializer (in 3.9.2, page 89)
    - DataTransferControl.Servet (in 3.8.13, page 87)

3.8 Package DataTransferControl

Package Contents *Page*

Classes

3.8.1 Class Collection

A Collection that stores multiple objects of one type

Declaration

```
public class Collection
 extends java.lang.Object
```

Constructor summary

Collection() Default constructor

Constructors

- **Collection**

  ```
  public Collection()
  ```

  - **Description**

    Default constructor

3.8.2 Class Config

The specified configuration-object that stores all needed configurations for the connection from Kafka to another specified component

Declaration

```
public class Config
 extends java.lang.Object
```

All known subclasses

GraphiteConfig (in 3.8.7, page 81)

Constructor summary

Config() Default constructor

Method summary

getKafkaHostName() Gets the Kafka-host-name
getStartFromBeginning() Returns whether a start from the beginning is required

Constructors

- **Config**

  ```
  public Config()
  ```

  - **Description**

    Default constructor

Methods

- **getKafkaHostName**

  ```
  public static java.lang.String getKafkaHostName()
  ```

  - **Description**

    Gets the Kafka-host-name
  - **Returns** – The host-name of Kafka

- **getStartFromBeginning**

  ```
  public static boolean getStartFromBeginning()
  ```

  - **Description**

    Returns whether a start from the beginning is required
  - **Returns** – Tells us whether a start from the beginning is required

3.8.3 Class Consumer

Consumes data from Kafka

Declaration

```
public class Consumer
 extends java.lang.Object
```

All known subclasses

KafkaToGraphiteConsumer (in 3.8.10, page 84)

Constructor summary

Consumer() Default constructor

Method summary

run() Starts the transferring-process
stop() Stops the transferring-process

Constructors

- **Consumer**

  ```
  public Consumer()
  ```

  - **Description**

    Default constructor

Methods

- **run**

  ```
  public void run()
  ```

  - **Description**

    Starts the transferring-process

- **stop**

  ```
  public void stop()
  ```

  - **Description**

    Stops the transferring-process

3.8.4 Class ConsumerRecord

One single record of data from Kafka

Declaration

```
public class ConsumerRecord
 extends java.lang.Object
```

Constructor summary

ConsumerRecord() Default constructor

Constructors

- **ConsumerRecord**

  ```
  public ConsumerRecord()
  ```

  - **Description**

    Default constructor

3.8.5 Class ConsumerRecords

Multiple records of data from Kafka

Declaration

```
public class ConsumerRecords
 extends java.lang.Object
```

Constructor summary

ConsumerRecords() Default constructor

Constructors

- **ConsumerRecords**

  ```
  public ConsumerRecords()
  ```

  - **Description**

    Default constructor

3.8.6 Class GraphDataTransferController

The Control-Unit in charge of creating and destroying KafkaToGraphiteConsumer as well as passing on the users request.

Declaration

```
public class GraphDataTransferController
 extends java.lang.Object
```

Constructor summary

GraphDataTransferController() Default constructor

Method summary

startDataTransfer(,) Starts data-transfer
stopDataTransfer() Stoppt den Datentransfer.

Constructors

- **GraphDataTransferController**

  ```
  public GraphDataTransferController()
  ```

  - **Description**

    Default constructor

Methods

- **startDataTransfer**

  ```
  public void startDataTransfer(Collection<String> topics,Collection<
      String> UIDS)
  ```

  - **Description**

    Starts data-transfer

  - **Parameters**

    * `topics` – Kafka-Topics that should be subscribed
    * `UIDS` – The unique identifiers, that tell us which data should be transfered. Everything else will be ignored.

- **stopDataTransfer**

```
public void stopDataTransfer(Collection<String> UIDS)
```

- **Description**

  Stoppt den Datentransfer.

- **Parameters**

  * `UIDS` – The unique identifiers, that tell us which data should no longer be transfered. Everything else will be ignored.

3.8.7 Class GraphiteConfig

The specified configuration-object that stores all needed configurations for the connection from Kafka to Graphite

Declaration

```
public class GraphiteConfig
 extends DataTransferControl.Config
```

Constructor summary

GraphiteConfig() Default constructor

Method summary

getGraphiteHostName() Returns the host-name of Graphite
getGraphitePort() Returns the port of the Graphite-connection

Constructors

- **GraphiteConfig**

```
public GraphiteConfig()
```

  - **Description**

    Default constructor

Methods

- **getGraphiteHostName**

```
public static java.lang.String getGraphiteHostName()
```

- **Description**

  Returns the host-name of Graphite

- **Returns** – The Graphite-host-name

- **getGraphitePort**

```
public static java.lang.Integer getGraphitePort()
```

  - **Description**

    Returns the port of the Graphite-connection

  - **Returns** – The port of the Graphite-connection

Members inherited from class Config

`DataTransferControl.Config` (in 3.8.2, page 76)

- `public static String` **getKafkaHostName()**
- `public static boolean` **getStartFromBeginning()**

3.8.8 Class GraphiteSender

Reformats the data and sends it to Graphite

Declaration

```
public class GraphiteSender
 extends DataTransferControl.Sender
```

Constructor summary

GraphiteSender() Default constructor

Constructors

- **GraphiteSender**

```
public GraphiteSender()
```

  - **Description**

    Default constructor

Members inherited from class Sender

`DataTransferControl.Sender` (in 3.8.12, page 86)

- `public void` **send(records)**

3.8.9 Class KafkaConsumer

The Kafka Consumer is described in Apache-Kafka and will only be included in this diagram for a better understanding of the required functionality.

Declaration

```
public class KafkaConsumer
 extends java.lang.Object
```

Constructor summary

KafkaConsumer() Default constructor

Method summary

close() Closes the KafkaConsumer
poll(long) Gathers the data
seekToBeginning() Jumps to the beginning of an existing record
subscribe() The Consumer subscribes Kafka-Topics.
wakeup() Wakes up the KafkaConsumer, which then stops any current requests.

Constructors

- **KafkaConsumer**

  ```
  public KafkaConsumer()
  ```

  - **Description**

    Default constructor

Methods

- **close**

  ```
  public void close()
  ```

  - **Description**

    Closes the KafkaConsumer

- **poll**

  ```
  public void poll(long timeout)
  ```

  - **Description**

    Gathers the data
  - **Parameters**
    - `timeout` – A timeframe, limiting the longest possible duration of the poll request

- **seekToBeginning**

```
public void seekToBeginning(Collection<TopicPartition> partitions)
```

  - **Description**

    Jumps to the beginning of an existing record
  - **Parameters**
    - `partitions` – Kafka-Partitions

- **subscribe**

```
public void subscribe(Collection<String> topics)
```

  - **Description**

    The Consumer subscribes Kafka-Topics.
  - **Parameters**
    - `topics` – Kafka-Topics that should be subscribed

- **wakeup**

```
public void wakeup()
```

  - **Description**

    Wakes up the KafkaConsumer, which then stops any current requests. Useful to limit polls in general.

3.8.10 Class KafkaToGraphiteConsumer

Receives the data from Kafka and sends it to Graphite

Declaration

```
public class KafkaToGraphiteConsumer
 extends DataTransferControl.Consumer
```

Constructor summary

KafkaToGraphiteConsumer() Default constructor

Method summary

run() Starts the process of consumation and readying the sender object
stop() Starts the process

Constructors

- **KafkaToGraphiteConsumer**

  ```
  public KafkaToGraphiteConsumer()
  ```

  - **Description**

    Default constructor

Methods

- **run**

  ```
  public void run()
  ```

  - **Description**

    Starts the process of consumation and readying the sender object

- **stop**

  ```
  public void stop()
  ```

  - **Description**

    Starts the process

Members inherited from class Consumer

`DataTransferControl.Consumer` (in 3.8.3, page 78)

- `public void` **`run()`**
- `public void` **`stop()`**

3.8.11 Class Properties

The Properties of the KafkaConsumer, using Java.Util.Properties

Declaration

```
public class Properties
 extends java.lang.Object
```

Constructor summary

Properties() Default constructor

Constructors

- **Properties**

  ```
  public Properties()
  ```

  - **Description**

    Default constructor

3.8.12 Class Sender

Reformats the data and sends it to another component

Declaration

```
public class Sender
 extends java.lang.Object
```

All known subclasses

GraphiteSender (in 3.8.8, page 82)

Constructor summary

Sender() Default constructor

Method summary

send() Sends the resulting data to the specified component

Constructors

- **Sender**

  ```
  public Sender()
  ```

  - **Description**

    Default constructor

Methods

- **send**

```
public void send(ConsumerRecords<String, KafkaObservationData>
    records)
```

  - **Description**

    Sends the resulting data to the specified component

  - **Parameters**

    * `records` – Multiple records of data from Kafka

3.8.13 Class Servet

A Servlet, which accepts the user-requests from the webinterface and passes them on to the responsible structures

Declaration

```
public class Servet
 extends java.lang.Object
```

Constructor summary

Servet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Receives the information of the data, that will be send back

Constructors

- **Servet**

```
public Servet()
```

  - **Description**

    Default constructor

Methods

- **doGet**

```
public void doGet(HttpServletRequest req,HttpServletResponse resp)
```

  - **Description**

    Receives the information of the data, that will be send back

  - **Parameters**

    * `req` – A http servlet request
    * `resp` – A http servlet response

3.9 Package DataTransferControl.SerializationDeserialization

Package Contents *Page*

Classes

3.9.1 Class KafkaObservationData

A serializable object that contains the observed data from kafka

Declaration

```
public class KafkaObservationData
 extends java.lang.Object implements java.io.Serializable
```

Field summary

locationElevation The height of the observations location
locationID The id of the observations location
locationName The name of the observations location
observationDate The date of the observation
observedProperty The observed property

Constructor summary

KafkaObservationData() Default constructor

Fields

- `public java.lang.String` **observationDate**
  - The date of the observation
- `public java.lang.String` **locationName**
  - The name of the observations location
- `public java.lang.String` **locationElevation**
  - The height of the observations location
- `public java.lang.String` **locationID**
  - The id of the observations location
- `public java.lang.String` **observedProperty**
  - The observed property

Constructors

- **KafkaObservationData**

```
public KafkaObservationData()
```

  - **Description**

    Default constructor

3.9.2 Class ObservationDataDeserializer

Deserializes KafkaObservationData objects

Declaration

```
public class ObservationDataDeserializer
 extends java.lang.Object
```

Constructor summary

ObservationDataDeserializer() Default constructor

Method summary

close() Closes this object
configure(Map, boolean) Configures the deserializer
deserialize(Collection, Set) Deserializes an object

Constructors

- **ObservationDataDeserializer**

  ```
  public ObservationDataDeserializer()
  ```

  - **Description**

    Default constructor

Methods

- **close**

  ```
  public void close()
  ```

  - **Description**

    Closes this object

- **configure**

  ```
  public void configure(java.util.Map configs,boolean isKey)
  ```

  - **Description**

    Configures the deserializer
  - **Parameters**
    - `configs` – The Configuration
    - `isKey` – A variable, telling us whether we want to configure the key or the value

- **deserialize**

  ```
  public KafkaObservationData deserialize(java.util.Collection topics,
      java.util.Set data)
  ```

  - **Description**

    Deserializes an object
  - **Parameters**
    - `topics` – Kafka-Topics that should be subscribed
    - `data` – These are our serialized bytes
  - **Returns** – A serializable object that contains the observed data from kafka

Class Hierarchy

Classes

- java.lang.Object
    - Grid.Cluster (in 3.10.1, page 93)
    - Grid.Dimension (in 3.10.2, page 95)
    - Grid.Grid (in 3.10.3, page 96)
    - Grid.Image (in 3.10.4, page 97)
    - Grid.Tile (in 3.10.7, page 99)
        - Grid.ImageTile (in 3.10.5, page 97)
        - Grid.ShapeTile (in 3.10.6, page 98)
    - View.AbstractView (in 3.11.1, page 101)
        - View.View (in 3.11.3, page 106)
    - View.AbstractViewFactory (in 3.11.2, page 104)
        - View.ViewFactory (in 3.11.5, page 107)
    - View.Graph.GraphDisplayType (in 3.13.4, page 119)
    - View.Map.MapLayer (in 3.14.6, page 130)
    - View.Map.TileType (in 3.14.8, page 132)
    - View.SensorOption.ObservedProperty (in 3.15.3, page 136)
    - View.TimeOption.RefreshConfiguration (in 3.17.6, page 149)
    - View.TimeOption.RefreshContext (in 3.17.7, page 151)
    - View.TimeOption.RefreshState (in 3.17.8, page 153)
        - View.TimeOption.HistoricalRefreshState (in 3.17.3, page 144)
        - View.TimeOption.LiveRefreshState (in 3.17.4, page 146)
        - View.TimeOption.LoopRefreshState (in 3.17.5, page 147)
    - View.Util.Date (in 3.18.2, page 157)
    - View.Util.Identifier (in 3.18.3, page 158)
        - View.Util.ClusterID (in 3.18.1, page 157)
        - View.Util.SensorID (in 3.18.5, page 160)
    - View.Util.Point (in 3.18.4, page 159)
    - View.Util.TimeFrame (in 3.18.6, page 161)
    - View.ViewComponent (in 3.11.4, page 107)
    - View.ViewManager (in 3.11.6, page 108)
    - ViewComponent
        - View.ExportOption.AbstractExportOptionPanel (in 3.12.1, page 109)
            - View.ExportOption.ExportOptionPanel (in 3.12.2, page 112)
        - View.Graph.AbstractGraph (in 3.13.2, page 114)
            - View.Graph.GraphiteGraph (in 3.13.5, page 119)
        - View.Graph.AbstractGraphOptionPanel (in 3.13.3, page 117)
            - View.Graph.GraphOptionPanel (in 3.13.6, page 120)
        - View.Map.AbstractMap (in 3.14.3, page 123)
            - View.Map.LeafletMap (in 3.14.5, page 129)
        - View.Map.AbstractMapOptionPanel (in 3.14.4, page 127)

- View.Map.MapOptionPanel (in 3.14.7, page 131)
- View.SensorOption.AbstractSensorOptionPanel (in 3.15.2, page 134)
  - View.SensorOption.SensorOptionPanel (in 3.15.4, page 136)
- View.SensorTable.AbstractSensorTable (in 3.16.1, page 137)
  - View.SensorTable.SensorTable (in 3.16.2, page 139)
- View.TimeOption.AbstractTimeOptionPanel (in 3.17.2, page 142)
  - View.TimeOption.TimeOptionPanel (in 3.17.9, page 155)

Interfaces

- View.Graph.GraphOptionPanelObserver (in 3.13.1, page 114)
- View.Map.MapObserver (in 3.14.1, page 122)
- View.Map.MapOptionPanelObserver (in 3.14.2, page 122)
- View.SensorOption.SensorOptionPanelObserver (in 3.15.1, page 133)
- View.TimeOption.TimeOptionPanelObserver (in 3.17.1, page 141)

3.10 Package Grid

Package Contents *Page*

Classes

3.10.1 Class Cluster

Encapsulates multiple sensors into a single object by using their specific SensorIDs and provides a graphical representation of their values average by using a Tile.

Declaration

```
public class Cluster
 extends java.lang.Object
```

Constructor summary

Cluster() Default constructor

Method summary

getClusterId() Get the ClusterID of this Cluster.
getObservedProperty() Get the ObservedProperty of this Cluster.
getSensorIds() Get all SensorIDs of the sensors contained in this cluster.
getTile() Get the Tile of this Cluster.
setObservedProperty(ObservedProperty) Set the ObservedProperty of this Cluster.
setSensorIds(Set) Set the SensorIDs of the sensors contained in this cluster.

Constructors

- **Cluster**

  ```
  public Cluster()
  ```

  - **Description**

    Default constructor

Methods

- **getClusterId**

  ```
  public ClusterID getClusterId()
  ```

  - **Description**

    Get the ClusterID of this Cluster.
  - **Returns** – the ClusterID of this Cluster.

- **getObservedProperty**

  ```
  public ObservedProperty getObservedProperty()
  ```

  - **Description**

    Get the ObservedProperty of this Cluster.
  - **Returns** – the ObservedProperty of this Cluster.

- **getSensorIds**

  ```
  public java.util.Set getSensorIds()
  ```

  - **Description**

    Get all SensorIDs of the sensors contained in this cluster.
  - **Returns** – all SensorIDs of the sensors contained in this cluster.

- **getTile**

  ```
  public Tile getTile()
  ```

  - **Description**

    Get the Tile of this Cluster.
  - **Returns** – the Tile of this Cluster.

- **setObservedProperty**

  ```
  public void setObservedProperty(ObservedProperty observedProperty)
  ```

  - **Description**

    Set the ObservedProperty of this Cluster.
  - **Parameters**
    - `observedProperty` –

- **setSensorIds**

  ```
  public void setSensorIds(java.util.Set sensorIds)
  ```

  - **Description**

    Set the SensorIDs of the sensors contained in this cluster.
  - **Parameters**
    - `sensorIds` –

3.10.2 Class Dimension

Encapsulates the width and height of a component in float precision.

Declaration

```
public class Dimension
 extends java.lang.Object
```

Constructor summary

Dimension() Default constructor

Method summary

getHeight() Get the height of this Dimension.
getWidth() Get the width of this Dimension.

Constructors

- **Dimension**

  ```
  public Dimension()
  ```

  - **Description**

    Default constructor

Methods

- **getHeight**

  ```
  public float getHeight()
  ```

  - **Description**

    Get the height of this Dimension.
  - **Returns** – the height of this Dimension.

- **getWidth**

  ```
  public float getWidth()
  ```

  - **Description**

    Get the width of this Dimension.
  - **Returns** – the width of this Dimension.

3.10.3 Class Grid

Encapsulates multiple Clusters into a single object.

Declaration

```
public class Grid
 extends java.lang.Object
```

Constructor summary

Grid() Default constructor

Method summary

getClusters() Get all Clusters contained in this Grid.
selectClusters(Set) Select Clusters contained in this Grid by using their specific ClusterIDs.
setClusters(Set) Set the Clusters contained in this Grid.

Constructors

- **Grid**

  ```
  public Grid()
  ```

  - **Description**

    Default constructor

Methods

- **getClusters**

  ```
  public java.util.Set getClusters()
  ```

  - **Description**

    Get all Clusters contained in this Grid.
  - **Returns** – all Clusters contained in this Grid.

- **selectClusters**

  ```
  public java.util.Set selectClusters(java.util.Set ids)
  ```

- **Description**

  Select Clusters contained in this Grid by using their specific ClusterIDs.

- **Parameters**

  * `ids` –

- **Returns** – selected Clusters contained in this Grid identified by their specific ClusterIDs.

- **setClusters**

```
public void setClusters(java.util.Set clusters)
```

  - **Description**

    Set the Clusters contained in this Grid.

  - **Parameters**

    * `clusters` –

3.10.4 Class Image

Represents a graphical image.

Declaration

```
public class Image
 extends java.lang.Object
```

Constructor summary

Image() Default constructor

Constructors

- **Image**

```
public Image()
```

  - **Description**

    Default constructor

3.10.5 Class ImageTile

A Tile whose graphical representation consists of an image.

Declaration

```
public class ImageTile
 extends Grid.Tile
```

Constructor summary

ImageTile() Default constructor

Constructors

- **ImageTile**

  ```
  public ImageTile()
  ```

  - **Description**

    Default constructor

Members inherited from class Tile

`Grid.Tile` (in 3.10.7, page 99)

- `protected` **color**
- `public void` **display**`(java.util.AbstractMap` **map**`)`
- `protected` **opacity**
- `protected` **position**
- `public void` **setColor**`(Color` **color**`)`
- `public void` **setOpacity**`(float` **opacity**`)`
- `public void` **setPosition**`(Point` **position**`)`

3.10.6 Class ShapeTile

A Tile whose graphical representation consists of a shape, specified by an array of vertices.

Declaration

```
public class ShapeTile
 extends Grid.Tile
```

Constructor summary

ShapeTile() Default constructor

Constructors

- **ShapeTile**

  ```
  public ShapeTile()
  ```

  - **Description**

    Default constructor

Members inherited from class Tile

`Grid.Tile` (in 3.10.7, page 99)

- `protected` **color**
- `public void` **display**`(java.util.AbstractMap` **map**`)`
- `protected` **opacity**
- `protected` **position**
- `public void` **setColor**`(Color` **color**`)`
- `public void` **setOpacity**`(float` **opacity**`)`
- `public void` **setPosition**`(Point` **position**`)`

3.10.7 Class Tile

A graphical structure that can be displayed on an AbstractMap.

Declaration

```
public class Tile
 extends java.lang.Object
```

All known subclasses

ShapeTile (in 3.10.6, page 98), ImageTile (in 3.10.5, page 97)

Field summary

color
opacity
position

Constructor summary

Tile() Default constructor

Method summary

display(AbstractMap) Display this tile on the submitted map.
setColor(Color) Set the color of this Tile.
setOpacity(float) Set the opacity of this Tile.
setPosition(Point) Set the position of this Tile.

Fields

- `protected Point` **position**
- `protected Color` **color**
- `protected float` **opacity**

Constructors

- **Tile**

  ```
  public Tile()
  ```

  - **Description**

    Default constructor

Methods

- **display**

  ```
  public void display(java.util.AbstractMap map)
  ```

  - **Description**

    Display this tile on the submitted map.
  - **Parameters**
    * `map` –

- **setColor**

  ```
  public void setColor(Color color)
  ```

  - **Description**

    Set the color of this Tile.
  - **Parameters**
    * `color` –

- **setOpacity**

  ```
  public void setOpacity(float opacity)
  ```

  - **Description**

    Set the opacity of this Tile.

  - **Parameters**

    * `opacity` –

- **setPosition**

  ```
  public void setPosition(Point position)
  ```

  - **Description**

    Set the position of this Tile.

  - **Parameters**

    * `position` –

3.11 Package View

Package Contents *Page*

Classes

3.11.1 Class AbstractView

Encapsulates all ViewComponents created by the AbstractViewFactory into a single object.

Declaration

```
public class AbstractView
 extends java.lang.Object
```

All known subclasses

View (in 3.11.3, page 106)

Constructor summary

AbstractView() Default constructor

Method summary

getExportOptionPanel() Get the AbstractExportOptionPanel.
getGraph() Get the AbstractGraph.
getGraphOptionPanel() Get the AbstractGraphOptionPanel.
getMap() Get the AbstractMap.
getMapOptionPanel() Get the AbstractMapOptionPanel.
getSensorOptionPanel() Get the AbstractSensorOptionPanel.
getSensorTable() Get the AbstractSensorTable.
getTimeOptionPanel() Get the AbstractTimeOptionPanel.

Constructors

- **AbstractView**

  ```
  public AbstractView()
  ```

  - **Description**

    Default constructor

Methods

- **getExportOptionPanel**

  ```
  public AbstractExportOptionPanel getExportOptionPanel()
  ```

  - **Description**

    Get the AbstractExportOptionPanel.
  - **Returns** – the AbstractExportOptionPanel.

- **getGraph**

```
public AbstractGraph getGraph()
```

  - **Description**
    Get the AbstractGraph.
  - **Returns** – the AbstractGraph.

- **getGraphOptionPanel**

```
public AbstractGraphOptionPanel getGraphOptionPanel()
```

  - **Description**
    Get the AbstractGraphOptionPanel.
  - **Returns** – the AbstractGraphOptionPanel.

- **getMap**

```
public java.util.AbstractMap getMap()
```

  - **Description**
    Get the AbstractMap.
  - **Returns** – the AbstractMap.

- **getMapOptionPanel**

```
public AbstractMapOptionPanel getMapOptionPanel()
```

  - **Description**
    Get the AbstractMapOptionPanel.
  - **Returns** – the AbstractMapOptionPanel.

- **getSensorOptionPanel**

```
public AbstractSensorOptionPanel getSensorOptionPanel()
```

  - **Description**
    Get the AbstractSensorOptionPanel.
  - **Returns** – the AbstractSensorOptionPanel.

- **getSensorTable**

```
public AbstractSensorTable getSensorTable()
```

  - **Description**

    Get the AbstractSensorTable.
  - **Returns** – the AbstractSensorTable.

- **getTimeOptionPanel**

```
public AbstractTimeOptionPanel getTimeOptionPanel()
```

  - **Description**

    Get the AbstractTimeOptionPanel.
  - **Returns** – the AbstractTimeOptionPanel.

3.11.2 Class AbstractViewFactory

A factory for the creation of a View.

Declaration

```
public class AbstractViewFactory
 extends java.lang.Object
```

All known subclasses

ViewFactory (in 3.11.5, page 107)

Constructor summary

AbstractViewFactory() Default constructor

Method summary

createExportOptionPanel() Create an AbstractExportOptionPanel.
createGraph() Create an AbstractGraph.
createGraphOptionPanel() Create an AbstractGraphOptionPanel.
createMap() Create an AbstractMap.
createMapOptionPanel() Create an AbstractMapOptionPanel.
createSensorOptionPanel() Create an AbstractSensorOptionPanel.
createSensorTable() Create an AbstractSensorTable.
createTimeOptionPanel() Create an AbstractTimeOptionPanel.

Constructors

- **AbstractViewFactory**

  ```
  public AbstractViewFactory()
  ```

  - **Description**

    Default constructor

Methods

- **createExportOptionPanel**

  ```
  public void createExportOptionPanel()
  ```

  - **Description**

    Create an AbstractExportOptionPanel.

- **createGraph**

  ```
  public void createGraph()
  ```

  - **Description**

    Create an AbstractGraph.

- **createGraphOptionPanel**

  ```
  public void createGraphOptionPanel()
  ```

  - **Description**

    Create an AbstractGraphOptionPanel.

- **createMap**

  ```
  public void createMap()
  ```

  - **Description**

    Create an AbstractMap.

- **createMapOptionPanel**

  ```
  public void createMapOptionPanel()
  ```

- **Description**

  Create an AbstractMapOptionPanel.

- **createSensorOptionPanel**

  ```
  public void createSensorOptionPanel()
  ```

  - **Description**

    Create an AbstractSensorOptionPanel.

- **createSensorTable**

  ```
  public void createSensorTable()
  ```

  - **Description**

    Create an AbstractSensorTable.

- **createTimeOptionPanel**

  ```
  public void createTimeOptionPanel()
  ```

  - **Description**

    Create an AbstractTimeOptionPanel.

3.11.3 Class View

An implementation of AbstractView.

Declaration

```
public class View
 extends View.AbstractView
```

Constructor summary

View() Default constructor

Constructors

- **View**

  ```
  public View()
  ```

  - **Description**

    Default constructor

Members inherited from class AbstractView

`View.AbstractView` (in 3.11.1, page 101)

- `public AbstractExportOptionPanel` **getExportOptionPanel()**
- `public AbstractGraph` **getGraph()**
- `public AbstractGraphOptionPanel` **getGraphOptionPanel()**
- `public AbstractMap` **getMap()**
- `public AbstractMapOptionPanel` **getMapOptionPanel()**
- `public AbstractSensorOptionPanel` **getSensorOptionPanel()**
- `public AbstractSensorTable` **getSensorTable()**
- `public AbstractTimeOptionPanel` **getTimeOptionPanel()**

3.11.4 Class ViewComponent

A view component which the View is made up of.

Declaration

```
public class ViewComponent
 extends java.lang.Object
```

Constructor summary

ViewComponent() Default constructor

Constructors

- **ViewComponent**

  ```
  public ViewComponent()
  ```

  - **Description**

    Default constructor

3.11.5 Class ViewFactory

An Implementation of AbstractViewFactory.

Declaration

```
public class ViewFactory
 extends View.AbstractViewFactory
```

Constructor summary

ViewFactory() Default constructor

Constructors

- **ViewFactory**

  ```
  public ViewFactory()
  ```

  – **Description**

  Default constructor

Members inherited from class AbstractViewFactory

`View.AbstractViewFactory` (in 3.11.2, page 104)

- `public void` **createExportOptionPanel()**
- `public void` **createGraph()**
- `public void` **createGraphOptionPanel()**
- `public void` **createMap()**
- `public void` **createMapOptionPanel()**
- `public void` **createSensorOptionPanel()**
- `public void` **createSensorTable()**
- `public void` **createTimeOptionPanel()**

3.11.6 Class ViewManager

Initializes and runs the AbstractView.

Declaration

```
public class ViewManager
 extends java.lang.Object
```

Constructor summary

ViewManager() Default constructor

Method summary

init() Initializes the ViewManager by creating the View and its ViewComponents.
run() Run the View by looping the refresh method located in the AbstractTimeOption-Panel in your AbstractView.

Constructors

- **ViewManager**

  ```
  public ViewManager()
  ```

  – **Description**

  Default constructor

Methods

- **init**

  ```
  public void init()
  ```

  - **Description**

    Initializes the ViewManager by creating the View and its ViewComponents.

- **run**

  ```
  public void run()
  ```

  - **Description**

    Run the View by looping the refresh method located in the AbstractTimeOptionPanel in your AbstractView.

3.12 Package View.ExportOption

Package Contents *Page*

Classes

3.12.1 Class AbstractExportOptionPanel

A panel for handling user input, that deals with exporting datasets. The user can select Clusters by their ClusterIDs, Sensors by their SensorIDs, a time frame, sensor types and a file format.

Declaration

```
public class AbstractExportOptionPanel
 extends ViewComponent
```

All known subclasses

ExportOptionPanel (in 3.12.2, page 112)

Field summary

clusterIds
fileExtension
observedProperties
sensorIds
timeFrame

Constructor summary

AbstractExportOptionPanel() Default constructor

Method summary

export() Request an export with the given parameters.
mapUpdate()
sensorOptionUpdate() Update the observer with the current SensorOptionPanel state.
setClusterIds(Set) Set the ClusterIDs.
setFileExtension(FileExtension) Set the ExportFormat.
setObservedProperties(Set) Set the SensorTypes.
setSensorIds(Set) Set the SensorIDs.
setTimeFrame(TimeFrame) Set the TimeFrame.
timeOptionUpdate() Update the observer with the current TimeOptionPanel state.

Fields

- `protected java.util.Set` **clusterIds**
- `protected java.util.Set` **sensorIds**
- `protected TimeFrame` **timeFrame**
- `protected java.util.Set` **observedProperties**
- `protected FileExtension` **fileExtension**

Constructors

- **AbstractExportOptionPanel**

  public AbstractExportOptionPanel()

  – **Description**

  Default constructor

Methods

- **export**

  ```
  public void export()
  ```

  - **Description**

    Request an export with the given parameters.

- **mapUpdate**

  ```
  public void mapUpdate()
  ```

- **sensorOptionUpdate**

  ```
  public void sensorOptionUpdate()
  ```

  - **Description**

    Update the observer with the current SensorOptionPanel state.

- **setClusterIds**

  ```
  public void setClusterIds(java.util.Set clusterIds)
  ```

  - **Description**

    Set the ClusterIDs.

  - **Parameters**

    * `clusterIds` –

- **setFileExtension**

  ```
  public void setFileExtension(FileExtension fileExtension)
  ```

  - **Description**

    Set the ExportFormat.

  - **Parameters**

    * `fileExtension` –

- **setObservedProperties**

```
public void setObservedProperties(java.util.Set observedProperties)
```

  - **Description**
    Set the SensorTypes.
  - **Parameters**
    * `observedProperties` –

- **setSensorIds**

```
public void setSensorIds(java.util.Set sensorIds)
```

  - **Description**
    Set the SensorIDs.
  - **Parameters**
    * `sensorIds` –

- **setTimeFrame**

```
public void setTimeFrame(TimeFrame timeFrame)
```

  - **Description**
    Set the TimeFrame.
  - **Parameters**
    * `timeFrame` –

- **timeOptionUpdate**

```
public void timeOptionUpdate()
```

  - **Description**
    Update the observer with the current TimeOptionPanel state.

3.12.2 Class ExportOptionPanel

An implementation of AbstractExportOptionPanel.

Declaration

```
public class ExportOptionPanel
 extends View.ExportOption.AbstractExportOptionPanel
```

Constructor summary

ExportOptionPanel() Default constructor

Constructors

- **ExportOptionPanel**

  ```
  public ExportOptionPanel()
  ```

  - **Description**

    Default constructor

Members inherited from class AbstractExportOptionPanel

`View.ExportOption.AbstractExportOptionPanel` (in 3.12.1, page 109)

- `protected` **clusterIds**
- `public void` **export()**
- `protected` **fileExtension**
- `public void` **mapUpdate()**
- `protected` **observedProperties**
- `protected` **sensorIds**
- `public void` **sensorOptionUpdate()**
- `public void` **setClusterIds(**`java.util.Set` **clusterIds)**
- `public void` **setFileExtension(**`FileExtension` **fileExtension)**
- `public void` **setObservedProperties(**`java.util.Set` **observedProperties)**
- `public void` **setSensorIds(**`java.util.Set` **sensorIds)**
- `public void` **setTimeFrame(**`TimeFrame` **timeFrame)**
- `protected` **timeFrame**
- `public void` **timeOptionUpdate()**

3.13 Package View.Graph

Package Contents *Page*

Interfaces

Classes

3.13.1 Interface GraphOptionPanelObserver

An observer that is meant to observe changes in the GraphOptionPanel.

Declaration

```
public interface GraphOptionPanelObserver
```

All known subinterfaces

GraphiteGraph (in 3.13.5, page 119), AbstractGraph (in 3.13.2, page 114)

All classes known to implement interface

AbstractGraph (in 3.13.2, page 114)

Method summary

graphOptionUpdate() Update the observer with the current GraphOptionPanel state.

Methods

- **graphOptionUpdate**

  ```
  void graphOptionUpdate()
  ```

  - **Description**

    Update the observer with the current GraphOptionPanel state.

3.13.2 Class AbstractGraph

A graph that visualizes the data in its dataset.

Declaration

```
public class AbstractGraph
 extends ViewComponent implements GraphOptionPanelObserver
```

All known subclasses

GraphiteGraph (in 3.13.5, page 119)

Field summary

dataset
timeFrame
timeStamp

Constructor summary

AbstractGraph() Default constructor

Method summary

graphOptionUpdate() Update the observer with the current GraphOptionPanel state.
mapUpdate()
sensorOptionUpdate() Update the observer with the current SensorOptionPanel state.
setTimeFrame(TimeFrame) Set the starting and end time point of the displayed dataset segment.
setTimeStamp(Date) Set a time stamp.
timeOptionUpdate() Update the observer with the current TimeOptionPanel state.
updateDataset(RequestCommand) Update the dataset of this AbstractGraph by giving it a new RequestCommand.

Fields

- `protected RequestCommand` **dataset**
- `protected TimeFrame` **timeFrame**
- `protected java.util.Date` **timeStamp**

Constructors

- **AbstractGraph**

  ```
  public AbstractGraph()
  ```

  - **Description**

    Default constructor

Methods

- **graphOptionUpdate**

  ```
  public void graphOptionUpdate()
  ```

  - **Description**

    Update the observer with the current GraphOptionPanel state.

- **mapUpdate**

  ```
  public void mapUpdate()
  ```

- **sensorOptionUpdate**

  ```
  public void sensorOptionUpdate()
  ```

  - **Description**

    Update the observer with the current SensorOptionPanel state.

- **setTimeFrame**

  ```
  public void setTimeFrame(TimeFrame timeFrame)
  ```

  - **Description**

    Set the starting and end time point of the displayed dataset segment.
  - **Parameters**
    - `timeFrame` –

- **setTimeStamp**

  ```
  public void setTimeStamp(java.util.Date timeStamp)
  ```

  - **Description**

    Set a time stamp.
  - **Parameters**
    - `timeStamp` –

- **timeOptionUpdate**

  ```
  public void timeOptionUpdate()
  ```

  - **Description**

    Update the observer with the current TimeOptionPanel state.

- **updateDataset**

  ```
  public void updateDataset(RequestCommand dataset)
  ```

  - **Description**

    Update the dataset of this AbstractGraph by giving it a new RequestCommand.
  - **Parameters**
    - `dataset` –

3.13.3 Class AbstractGraphOptionPanel

A panel for handling user input, that deals with which time segment of the graphs dataset is being displayed, how that is done and notifying all observers about changes in its state.

Declaration

```
public class AbstractGraphOptionPanel
 extends ViewComponent
```

All known subclasses

GraphOptionPanel (in 3.13.6, page 120)

Field summary

timeframe

Constructor summary

AbstractGraphOptionPanel() Default constructor

Method summary

getDisplayTypes() Get the GraphDisplayTypes.
notify() Notify all subscribed GraphOptionPanelObservers about a change in this AbstractGraphOptionPanel.
setDisplayTypes(Set) Set the GraphDisplayTypes.
setTimeFrame(TimeFrame) Set the starting and end time point of the displayed dataset segment..
subscribeObserver(GraphOptionPanelObserver) Subscribe a GraphOptionPanelObserver to this AbstractGraphOptionPanel.
unsubscribeObserver(GraphOptionPanelObserver) Unsubscribe a GraphOptionPanelObserver from this AbstractGraphOptionPanel.

Fields

- `protected TimeFrame` **timeframe**

Constructors

- **AbstractGraphOptionPanel**

```
public AbstractGraphOptionPanel()
```

- **Description**

  Default constructor

Methods

- **getDisplayTypes**

  ```
  public java.util.Set getDisplayTypes()
  ```

  - **Description**

    Get the GraphDisplayTypes.

  - **Returns** – the GraphDisplayTypes.

- **notify**

  ```
  public void notify()
  ```

  - **Description**

    Notify all subscribed GraphOptionPanelObservers about a change in this AbstractGraphOptionPanel.

- **setDisplayTypes**

  ```
  public void setDisplayTypes(java.util.Set displayTypes)
  ```

  - **Description**

    Set the GraphDisplayTypes.

  - **Parameters**

    * `displayTypes` –

- **setTimeFrame**

  ```
  public void setTimeFrame(TimeFrame timeFrame)
  ```

  - **Description**

    Set the starting and end time point of the displayed dataset segment..

  - **Parameters**

    * `timeFrame` –

- **subscribeObserver**

```
public void subscribeObserver(GraphOptionPanelObserver observer)
```

- **Description**

  Subscribe a GraphOptionPanelObserver to this AbstractGraphOptionPanel.

- **Parameters**

  * `observer` –

- **unsubscribeObserver**

```
public void unsubscribeObserver(GraphOptionPanelObserver observer)
```

- **Description**

  Unsubscribe a GraphOptionPanelObserver from this AbstractGraphOptionPanel.

- **Parameters**

  * `observer` –

3.13.4 Class GraphDisplayType

The display type of a graph.

Declaration

```
public class GraphDisplayType
 extends java.lang.Object
```

Constructor summary

GraphDisplayType() Default constructor

Constructors

- **GraphDisplayType**

```
public GraphDisplayType()
```

- **Description**

  Default constructor

3.13.5 Class GraphiteGraph

An AbstractGraph that uses the Graphite API.

Declaration

```
public class GraphiteGraph
 extends View.Graph.AbstractGraph
```

Constructor summary

GraphiteGraph() Default constructor

Constructors

- **GraphiteGraph**

  ```
  public GraphiteGraph()
  ```

  - **Description**

    Default constructor

Members inherited from class AbstractGraph

`View.Graph.AbstractGraph` (in 3.13.2, page 114)

- `protected` **dataset**
- `public void` **graphOptionUpdate()**
- `public void` **mapUpdate()**
- `public void` **sensorOptionUpdate()**
- `public void` **setTimeFrame(`**`TimeFrame` **timeFrame)**
- `public void` **setTimeStamp(`**`java.util.Date` **timeStamp)**
- `protected` **timeFrame**
- `public void` **timeOptionUpdate()**
- `protected` **timeStamp**
- `public void` **updateDataset(`**`RequestCommand` **dataset)**

3.13.6 Class GraphOptionPanel

An implementation of AbstractGraphOptionPanel.

Declaration

```
public class GraphOptionPanel
 extends View.Graph.AbstractGraphOptionPanel
```

Constructor summary

GraphOptionPanel() Default constructor

Constructors

- **GraphOptionPanel**

  public GraphOptionPanel()

  – **Description**

  Default constructor

Members inherited from class AbstractGraphOptionPanel

`View.Graph.AbstractGraphOptionPanel` (in 3.13.3, page 117)

- `public Set` **getDisplayTypes**`()`
- `public void` **notify**`()`
- `public void` **setDisplayTypes**`(java.util.Set` **displayTypes**`)`
- `public void` **setTimeFrame**`(TimeFrame` **timeFrame**`)`
- `public void` **subscribeObserver**`(GraphOptionPanelObserver` **observer**`)`
- `protected` **timeframe**
- `public void` **unsubscribeObserver**`(GraphOptionPanelObserver` **observer**`)`

3.14 Package View.Map

Package Contents *Page*

Interfaces

Classes

3.14.1 Interface MapObserver

Declaration

```
public interface MapObserver
```

Method summary

mapUpdate()

Methods

- **mapUpdate**

  ```
  void mapUpdate()
  ```

3.14.2 Interface MapOptionPanelObserver

An observer that is meant to observe changes in the MapOptionPanel.

Declaration

```
public interface MapOptionPanelObserver
```

All known subinterfaces

LeafletMap (in 3.14.5, page 129), AbstractMap (in 3.14.3, page 123)

All classes known to implement interface

AbstractMap (in 3.14.3, page 123)

Method summary

mapOptionUpdate() Update the observer with the current MapOptionPanel state.

Methods

- **mapOptionUpdate**

  ```
  void mapOptionUpdate()
  ```

  - **Description**
    Update the observer with the current MapOptionPanel state.

3.14.3 Class AbstractMap

A world map with displayable and hideable MapLayers, move and zoom function. It notifies its observers about changes in its state.

Declaration

```
public class AbstractMap
 extends ViewComponent implements MapOptionPanelObserver
```

All known subclasses

LeafletMap (in 3.14.5, page 129)

Field summary

dataset
position
timeStamp
zoom

Constructor summary

AbstractMap() Default constructor

Method summary

addLayer(MapLayer) Add a MapLayer.
displayLayer(MapLayer) Display a MapLayer.
hideLayer(MapLayer) Hide a MapLayer.
mapOptionUpdate() Update the observer with the current MapOptionPanel state.
notify() Notify all subscribed MapObservers about a change in this AbstractMap.
removeLayer(MapLayer) Remove a MapLayer.
sensorOptionUpdate() Update the observer with the current SensorOptionPanel state.
setPosition(Point) Set the center position of the AbstractMap.
setTimeStamp(Date) Set a time stamp and display the data from the dataset at the specified point in time.
setZoom(int) Set the zoom level of this AbstractMap.
subscribeObserver(MapObserver) Subscribe a MapObserver to this AbstractMap.
timeOptionUpdate() Update the observer with the current TimeOptionPanel state.
unsubscribeObserver(MapObserver) Unsubscribe a MapObserver from this AbstractMap.
updateDataset(RequestCommand) Update the dataset of this AbstractMap by giving it a new RequestCommand.

Fields

- `protected RequestCommand` **dataset**
- `protected java.util.Date` **timeStamp**
- `protected Point` **position**
- `protected int` **zoom**

Constructors

- **AbstractMap**

  ```
  public AbstractMap()
  ```

  - **Description**
    Default constructor

Methods

- **addLayer**

  ```
  public void addLayer(MapLayer layer)
  ```

  - **Description**
    Add a MapLayer.
  - **Parameters**
    * `layer` –
- **displayLayer**

  ```
  public void displayLayer(MapLayer layer)
  ```

  - **Description**
    Display a MapLayer.
  - **Parameters**
    * `layer` –
- **hideLayer**

  ```
  public void hideLayer(MapLayer layer)
  ```

  - **Description**

    Hide a MapLayer.
  - **Parameters**
    * `layer` –

- **mapOptionUpdate**

```
public void mapOptionUpdate()
```

  - **Description**

    Update the observer with the current MapOptionPanel state.

- **notify**

```
public void notify()
```

  - **Description**

    Notify all subscribed MapObservers about a change in this AbstractMap.

- **removeLayer**

```
public void removeLayer(MapLayer layer)
```

  - **Description**

    Remove a MapLayer.
  - **Parameters**
    * `layer` –

- **sensorOptionUpdate**

```
public void sensorOptionUpdate()
```

  - **Description**

    Update the observer with the current SensorOptionPanel state.

- **setPosition**

```
public void setPosition(Point point)
```

- **Description**

  Set the center position of the AbstractMap.

- **Parameters**

  * `point` –

- **setTimeStamp**

```
public void setTimeStamp(java.util.Date timeStamp)
```

  - **Description**

    Set a time stamp and display the data from the dataset at the specified point in time.

  - **Parameters**

    * `timeStamp` –

- **setZoom**

```
public void setZoom(int zoom)
```

  - **Description**

    Set the zoom level of this AbstractMap.

  - **Parameters**

    * `zoom` –

- **subscribeObserver**

```
public void subscribeObserver(MapObserver observer)
```

  - **Description**

    Subscribe a MapObserver to this AbstractMap.

  - **Parameters**

    * `observer` –

- **timeOptionUpdate**

```
public void timeOptionUpdate()
```

  - **Description**

    Update the observer with the current TimeOptionPanel state.

- **unsubscribeObserver**

  ```
  public void unsubscribeObserver(MapObserver observer)
  ```

  - **Description**

    Unsubscribe a MapObserver from this AbstractMap.
  - **Parameters**
    - `observer` –

- **updateDataset**

  ```
  public void updateDataset(RequestCommand dataset)
  ```

  - **Description**

    Update the dataset of this AbstractMap by giving it a new RequestCommand.
  - **Parameters**
    - `dataset` –

3.14.4 Class AbstractMapOptionPanel

A panel for handling user input, that deals with setting a new TileType and notifying its observers about the change.

Declaration

```
public class AbstractMapOptionPanel
 extends ViewComponent
```

All known subclasses

MapOptionPanel (in 3.14.7, page 131)

Field summary

observers

Constructor summary

AbstractMapOptionPanel() Default constructor

Method summary

getTileTypes() Get the TileTypes.
notify() Notify all subscribed MapOptionPanelObservers about a change in this AbstractMapOptionPanel.
setTileTypes(Set) Set the TileTypes.
subscribeObserver(MapOptionPanelObserver) Subscribe a MapOptionPanelObserver to this AbstractMapOptionPanel.
unsubscribeObserver(MapOptionPanelObserver) Unsubscribe a MapOptionPanelObserver from this AbstractMapOptionPanel.

Fields

- `protected java.util.Set` **observers**

Constructors

- **AbstractMapOptionPanel**

  ```
  public AbstractMapOptionPanel()
  ```

  - **Description**

    Default constructor

Methods

- **getTileTypes**

  ```
  public java.util.Set getTileTypes()
  ```

  - **Description**

    Get the TileTypes.
  - **Returns** – the TileTypes.

- **notify**

  ```
  public void notify()
  ```

  - **Description**

    Notify all subscribed MapOptionPanelObservers about a change in this AbstractMapOptionPanel.

- **setTileTypes**

```
public void setTileTypes(java.util.Set tileTypes)
```

  - **Description**

    Set the TileTypes.

  - **Parameters**

    * `tileTypes` –

- **subscribeObserver**

```
public void subscribeObserver(MapOptionPanelObserver observer)
```

  - **Description**

    Subscribe a MapOptionPanelObserver to this AbstractMapOptionPanel.

  - **Parameters**

    * `observer` –

- **unsubscribeObserver**

```
public void unsubscribeObserver(MapOptionPanelObserver observer)
```

  - **Description**

    Unsubscribe a MapOptionPanelObserver from this AbstractMapOptionPanel.

  - **Parameters**

    * `observer` –

3.14.5 Class LeafletMap

An AbstractMap that uses the Leaflet API.

Declaration

```
public class LeafletMap
 extends View.Map.AbstractMap
```

Constructor summary

LeafletMap() Default constructor

Constructors

- **LeafletMap**

  ```
  public LeafletMap()
  ```

  - **Description**

    Default constructor

Members inherited from class AbstractMap

`View.Map.AbstractMap` (in 3.14.3, page 123)

- `public void` **addLayer**`(MapLayer layer)`
- `protected` **dataset**
- `public void` **displayLayer**`(MapLayer layer)`
- `public void` **hideLayer**`(MapLayer layer)`
- `public void` **mapOptionUpdate**`()`
- `public void` **notify**`()`
- `protected` **position**
- `public void` **removeLayer**`(MapLayer layer)`
- `public void` **sensorOptionUpdate**`()`
- `public void` **setPosition**`(Point point)`
- `public void` **setTimeStamp**`(java.util.Date timeStamp)`
- `public void` **setZoom**`(int zoom)`
- `public void` **subscribeObserver**`(MapObserver observer)`
- `public void` **timeOptionUpdate**`()`
- `protected` **timeStamp**
- `public void` **unsubscribeObserver**`(MapObserver observer)`
- `public void` **updateDataset**`(RequestCommand dataset)`
- `protected` **zoom**

3.14.6 Class MapLayer

A map layer that can be displayed on an AbstractMap.

Declaration

```
public class MapLayer
 extends java.lang.Object
```

Field summary

layers

Constructor summary

MapLayer() Default constructor

Method summary

getGrid() Get the Grid of this MapLayer.
setGrid(Grid) Set the grid of this MapLayer.

Fields

- `protected AbstractMap` **layers**

Constructors

- **MapLayer**

  ```
  public MapLayer()
  ```

  - **Description**

    Default constructor

Methods

- **getGrid**

  ```
  public Grid getGrid()
  ```

  - **Description**

    Get the Grid of this MapLayer.
  - **Returns** – the Grid of this MapLayer.

- **setGrid**

  ```
  public void setGrid(Grid grid)
  ```

  - **Description**

    Set the grid of this MapLayer.
  - **Parameters**
    - `grid` –

3.14.7 Class MapOptionPanel

An implementation of AbstractMapOptionPanel.

Declaration

```
public class MapOptionPanel
 extends View.Map.AbstractMapOptionPanel
```

Constructor summary

MapOptionPanel() Default constructor

Constructors

- **MapOptionPanel**

  ```
  public MapOptionPanel()
  ```

  - **Description**

    Default constructor

Members inherited from class AbstractMapOptionPanel

`View.Map.AbstractMapOptionPanel` (in 3.14.4, page 127)

- `public Set` **getTileTypes**`()`
- `public void` **notify**`()`
- `protected` **observers**
- `public void` **setTileTypes**`(java.util.Set` **tileTypes**`)`
- `public void` **subscribeObserver**`(MapOptionPanelObserver` **observer**`)`
- `public void` **unsubscribeObserver**`(MapOptionPanelObserver` **observer**`)`

3.14.8 Class TileType

The type of a tile.

Declaration

```
public class TileType
 extends java.lang.Object
```

Field summary

tileTypes

Constructor summary

TileType() Default constructor

Fields

- `protected AbstractMapOptionPanel` **tileTypes**

Constructors

- **TileType**

  ```
  public TileType()
  ```

  – **Description**

  Default constructor

3.15 Package View.SensorOption

Package Contents *Page*

Interfaces

Classes

3.15.1 Interface SensorOptionPanelObserver

An observer that is meant to observe changes in the SensorOptionPanel.

Declaration

```
public interface SensorOptionPanelObserver
```

Method summary

sensorOptionUpdate() Update the observer with the current SensorOptionPanel state.

Methods

- **sensorOptionUpdate**

  ```
  void sensorOptionUpdate()
  ```

- **Description**

  Update the observer with the current SensorOptionPanel state.

3.15.2 Class AbstractSensorOptionPanel

A panel for handling user input, that deals with setting a new ObservedProperty and notifying its observers about changes.

Declaration

```
public class AbstractSensorOptionPanel
 extends ViewComponent
```

All known subclasses

SensorOptionPanel (in 3.15.4, page 136)

Constructor summary

AbstractSensorOptionPanel() Default constructor

Method summary

getObservedProperties() Get the sensor types.
notify() Notify all subscribed SensorOptionPanelObservers about a change in this AbstractSensorOptionPanel.
setObservedProperties(Set) Set the sensor types.
subscribeObserver(SensorOptionPanelObserver) Subscribe a SensorOptionPanelObserver to this AbstractSensorOptionPanel.
unsubscribeObserver(SensorOptionPanelObserver) Unsubscribe a SensorOptionPanelObserver from this AbstractSensorOptionPanel.

Constructors

- **AbstractSensorOptionPanel**

  ```
  public AbstractSensorOptionPanel()
  ```

  - **Description**

    Default constructor

Methods

- **getObservedProperties**

  ```
  public java.util.Set getObservedProperties()
  ```

  - **Description**

    Get the sensor types.
  - **Returns** – the sensor types.

- **notify**

  ```
  public void notify()
  ```

  - **Description**

    Notify all subscribed SensorOptionPanelObservers about a change in this AbstractSensorOptionPanel.

- **setObservedProperties**

  ```
  public void setObservedProperties(java.util.Set observedProperties)
  ```

  - **Description**

    Set the sensor types.
  - **Parameters**
    - `observedProperties` –

- **subscribeObserver**

  ```
  public void subscribeObserver(SensorOptionPanelObserver observer)
  ```

  - **Description**

    Subscribe a SensorOptionPanelObserver to this AbstractSensorOptionPanel.
  - **Parameters**
    - `observer` –

- **unsubscribeObserver**

  ```
  public void unsubscribeObserver(SensorOptionPanelObserver observer)
  ```

- **Description**

  Unsubscribe a SensorOptionPanelObserver from this AbstractSensorOptionPanel.

- **Parameters**

  * `observer` –

3.15.3 Class ObservedProperty

The data type measured by a sensor.

Declaration

```
public class ObservedProperty
 extends java.lang.Object
```

Constructor summary

ObservedProperty() Default constructor

Constructors

- **ObservedProperty**

  ```
  public ObservedProperty()
  ```

  - **Description**

    Default constructor

3.15.4 Class SensorOptionPanel

An implementation of AbstractSensorOptionPanel.

Declaration

```
public class SensorOptionPanel
 extends View.SensorOption.AbstractSensorOptionPanel
```

Constructor summary

SensorOptionPanel() Default constructor

Constructors

- **SensorOptionPanel**

  ```
  public SensorOptionPanel()
  ```

  - **Description**

    Default constructor

Members inherited from class AbstractSensorOptionPanel

`View.SensorOption.AbstractSensorOptionPanel` (in 3.15.2, page 134)

- `public Set` **getObservedProperties**`()`
- `public void` **notify**`()`
- `public void` **setObservedProperties**`(java.util.Set` **observedProperties**`)`
- `public void` **subscribeObserver**`(SensorOptionPanelObserver` **observer**`)`
- `public void` **unsubscribeObserver**`(SensorOptionPanelObserver` **observer**`)`

3.16 Package View.SensorTable

Package Contents *Page*

Classes

3.16.1 Class AbstractSensorTable

A table that visualizes the data in its dataset and enables the selection of a Sensor by using its SensorID.

Declaration

```
public class AbstractSensorTable
 extends ViewComponent
```

All known subclasses

SensorTable (in 3.16.2, page 139)

Field summary

dataset
timeStamp

Constructor summary

AbstractSensorTable() Default constructor

Method summary

mapUpdate()
selectSensor(SensorID) Select a sensor in the dataset by using its SensorID.
sensorOptionUpdate() Update the observer with the current SensorOptionPanel state.
setTimeStamp(Date) Set a time stamp and display the data from the dataset at the specified point in time.
timeOptionUpdate() Update the observer with the current TimeOptionPanel state.
updateDataset(RequestCommand) Update the dataset of this AbstractSensorTable by giving it a new RequestCommand.

Fields

- `protected RequestCommand` **dataset**
- `protected java.util.Date` **timeStamp**

Constructors

- **AbstractSensorTable**

  ```
  public AbstractSensorTable()
  ```

  - **Description**

    Default constructor

Methods

- **mapUpdate**

  ```
  public void mapUpdate()
  ```

- **selectSensor**

  ```
  public void selectSensor(SensorID sensorId)
  ```

- **Description**

  Select a sensor in the dataset by using its SensorID.

- **Parameters**

  * `sensorId` –

- **sensorOptionUpdate**

  ```
  public void sensorOptionUpdate()
  ```

  - **Description**

    Update the observer with the current SensorOptionPanel state.

- **setTimeStamp**

  ```
  public void setTimeStamp(java.util.Date timeStamp)
  ```

  - **Description**

    Set a time stamp and display the data from the dataset at the specified point in time.

  - **Parameters**

    * `timeStamp` –

- **timeOptionUpdate**

  ```
  public void timeOptionUpdate()
  ```

  - **Description**

    Update the observer with the current TimeOptionPanel state.

- **updateDataset**

  ```
  public void updateDataset(RequestCommand dataset)
  ```

  - **Description**

    Update the dataset of this AbstractSensorTable by giving it a new RequestCommand.

  - **Parameters**

    * `dataset` –

3.16.2 Class SensorTable

An implementation of AbstractSensorTable.

Declaration

```
public class SensorTable
 extends View.SensorTable.AbstractSensorTable
```

Constructor summary

SensorTable() Default constructor

Constructors

- **SensorTable**

  ```
  public SensorTable()
  ```

  – **Description**

  Default constructor

Members inherited from class AbstractSensorTable

`View.SensorTable.AbstractSensorTable` (in 3.16.1, page 137)

- `protected` **dataset**
- `public void` **mapUpdate()**
- `public void` **selectSensor(**`SensorID` **sensorId)**
- `public void` **sensorOptionUpdate()**
- `public void` **setTimeStamp(**`java.util.Date` **timeStamp)**
- `public void` **timeOptionUpdate()**
- `protected` **timeStamp**
- `public void` **updateDataset(**`RequestCommand` **dataset)**

3.17 Package View.TimeOption

Package Contents *Page*

Interfaces

Classes

3.17.1 Interface TimeOptionPanelObserver

An observer that is meant to observe changes in the TimeOptionPanel.

Declaration

```
public interface TimeOptionPanelObserver
```

All known subinterfaces

RefreshContext (in 3.17.7, page 151)

All classes known to implement interface

RefreshContext (in 3.17.7, page 151)

Method summary

timeOptionUpdate() Update the observer with the current TimeOptionPanel state.

Methods

- **timeOptionUpdate**

  ```
  void timeOptionUpdate()
  ```

  - **Description**

    Update the observer with the current TimeOptionPanel state.

3.17.2 Class AbstractTimeOptionPanel

A panel for handling user input, that deals with timing options and notifying its observers about changes in its state.

Declaration

```
public class AbstractTimeOptionPanel
 extends ViewComponent
```

All known subclasses

TimeOptionPanel (in 3.17.9, page 155)

Field summary

loopTimeFrame
refreshConfig
timeStamp

Constructor summary

AbstractTimeOptionPanel() Default constructor

Method summary

getLoopTimeframe() Get the loop time frame.
getRefreshContext() Get the RefreshContext.
getTimeStamp() Get the time stamp.
notify() Notify all subscribed TimeOptionPanelObservers about a change in this AbstractTimeOptionPanel.
setLoopTimeFrame(TimeFrame) Set the start and end time point of the loop.
setTimeStamp(Date) Set the time stamp.
subscribeObserver(TimeOptionPanelObserver) Subscribe a TimeOptionPanelObserver to this AbstractTimeOptionPanel.
unsubscribeObserver(TimeOptionPanelObserver) Unsubscribe a TimeOptionPanelObserver from this AbstractTimeOptionPanel.

Fields

- `protected TimeFrame` **loopTimeFrame**
- `protected java.util.Date` **timeStamp**
- `protected RefreshConfiguration` **refreshConfig**

Constructors

- **AbstractTimeOptionPanel**

  ```
  public AbstractTimeOptionPanel()
  ```

  - **Description**

    Default constructor

Methods

- **getLoopTimeframe**

  ```
  public TimeFrame getLoopTimeframe()
  ```

  - **Description**

    Get the loop time frame.
  - **Returns** – the loop time frame.

- **getRefreshContext**

  ```
  public RefreshContext getRefreshContext()
  ```

  - **Description**

    Get the RefreshContext.
  - **Returns** – the RefreshContext.

- **getTimeStamp**

  ```
  public java.util.Date getTimeStamp()
  ```

  - **Description**

    Get the time stamp.
  - **Returns** – the time stamp.

- **notify**

  ```
  public void notify()
  ```

  - **Description**

    Notify all subscribed TimeOptionPanelObservers about a change in this AbstractTimeOptionPanel.

- **setLoopTimeFrame**

  ```
  public void setLoopTimeFrame(TimeFrame loopTimeFrame)
  ```

  - **Description**

    Set the start and end time point of the loop.
  - **Parameters**
    - `loopTimeFrame` –

- **setTimeStamp**

  ```
  public void setTimeStamp(java.util.Date timeStamp)
  ```

  - **Description**

    Set the time stamp.
  - **Parameters**
    - `timeStamp` –

- **subscribeObserver**

  ```
  public void subscribeObserver(TimeOptionPanelObserver observer)
  ```

  - **Description**

    Subscribe a TimeOptionPanelObserver to this AbstractTimeOptionPanel.
  - **Parameters**
    - `observer` –

- **unsubscribeObserver**

  ```
  public void unsubscribeObserver(TimeOptionPanelObserver observer)
  ```

  - **Description**

    Unsubscribe a TimeOptionPanelObserver from this AbstractTimeOptionPanel.
  - **Parameters**
    - `observer` –

3.17.3 Class HistoricalRefreshState

In this state the refresh function simulates the historical data mode. The timeStamp parameter isn't altered and the currently selected dataset entries stay the same.

Declaration

```
public class HistoricalRefreshState
 extends View.TimeOption.RefreshState
```

Constructor summary

HistoricalRefreshState() Default constructor

Method summary

continueRoutine(RefreshContext) Switch to loop mode.
liveDataMode(RefreshContext) Switch to live data mode.
refresh(Date) Returns the submitted time stamp without any change.

Constructors

- **HistoricalRefreshState**

  ```
  public HistoricalRefreshState()
  ```

  - **Description**

    Default constructor

Methods

- **continueRoutine**

  ```
  public void continueRoutine(RefreshContext context)
  ```

  - **Description**

    Switch to loop mode.
  - **Parameters**

    * `context` –

- **liveDataMode**

  ```
  public void liveDataMode(RefreshContext context)
  ```

  - **Description**

    Switch to live data mode.
  - **Parameters**

    * `context` –

- **refresh**

  ```
  public java.util.Date refresh(java.util.Date timeStamp)
  ```

  - **Description**

    Returns the submitted time stamp without any change.

  - **Parameters**

    * `timeStamp` –

  - **Returns** – the submitted time stamp without any change.

Members inherited from class RefreshState

`View.TimeOption.RefreshState` (in 3.17.8, page 153)

- `public void` **continueRoutine**`(RefreshContext` **context**`)`
- `public void` **historicalDataMode**`(RefreshContext` **context**`)`
- `public void` **liveDataMode**`(RefreshContext` **context**`)`
- `public Date` **refresh**`(java.util.Date` **timeStamp**`)`
- `public void` **stopRoutine**`(RefreshContext` **context**`)`

3.17.4 Class LiveRefreshState

In this state the refresh function simulates the live data mode. Depending on the RefreshConfiguration the refresh function fetches live data. The timeStamp parameter isn't altered.

Declaration

```
public class LiveRefreshState
 extends View.TimeOption.RefreshState
```

Constructor summary

LiveRefreshState() Default constructor

Method summary

historicalDataMode(RefreshContext) Switch to historical data mode.
refresh(Date) Fetch live data and return the most up-to-date time stamp.

Constructors

- **LiveRefreshState**

  ```
  public LiveRefreshState()
  ```

  - **Description**

    Default constructor

Methods

- **historicalDataMode**

  ```
  public void historicalDataMode(RefreshContext context)
  ```

  - **Description**

    Switch to historical data mode.

  - **Parameters**

    * `context` –

- **refresh**

  ```
  public java.util.Date refresh(java.util.Date timeStamp)
  ```

  - **Description**

    Fetch live data and return the most up-to-date time stamp.

  - **Parameters**

    * `timeStamp` –

  - **Returns** – the most up-to-date time stamp.

Members inherited from class RefreshState

`View.TimeOption.RefreshState` (in 3.17.8, page 153)

- `public void` **continueRoutine**`(RefreshContext` **context**`)`
- `public void` **historicalDataMode**`(RefreshContext` **context**`)`
- `public void` **liveDataMode**`(RefreshContext` **context**`)`
- `public Date` **refresh**`(java.util.Date` **timeStamp**`)`
- `public void` **stopRoutine**`(RefreshContext` **context**`)`

3.17.5 Class LoopRefreshState

In this state the refresh function simulates the loop data mode. Depending on the loopTimeFrame value and the RefreshConfiguration, the refresh method modifies the submitted timeStamp which can be submitted to other ViewComponents to iterate to the next dataset entries.

Declaration

```
public class LoopRefreshState
 extends View.TimeOption.RefreshState
```

Constructor summary

LoopRefreshState() Default constructor

Method summary

liveDataMode(RefreshContext) Switch to live data mode.
refresh(Date) Returns the submitted time stamp modified according to the RefreshConfiguration.
stopRoutine(RefreshContext) Switch to historical mode.

Constructors

- **LoopRefreshState**

  ```
  public LoopRefreshState()
  ```

  - **Description**

    Default constructor

Methods

- **liveDataMode**

  ```
  public void liveDataMode(RefreshContext context)
  ```

  - **Description**

    Switch to live data mode.
  - **Parameters**
    * `context` –

- **refresh**

  ```
  public java.util.Date refresh(java.util.Date timeStamp)
  ```

  - **Description**

    Returns the submitted time stamp modified according to the RefreshConfiguration.
  - **Parameters**
    * `timeStamp` –
  - **Returns** – the submitted time stamp modified according to the RefreshConfiguration.

- **stopRoutine**

  ```
  public void stopRoutine(RefreshContext context)
  ```

- **Description**

  Switch to historical mode.

- **Parameters**

  * `context` –

Members inherited from class RefreshState

`View.TimeOption.RefreshState` (in 3.17.8, page 153)

- `public void` **continueRoutine**`(RefreshContext` **context**`)`
- `public void` **historicalDataMode**`(RefreshContext` **context**`)`
- `public void` **liveDataMode**`(RefreshContext` **context**`)`
- `public Date` **refresh**`(java.util.Date` **timeStamp**`)`
- `public void` **stopRoutine**`(RefreshContext` **context**`)`

3.17.6 Class RefreshConfiguration

Encapsulates the preferences about the fetching of live data and the loop mode of historical data.

Declaration

```
public class RefreshConfiguration
 extends java.lang.Object
```

Constructor summary

RefreshConfiguration() Default constructor

Method summary

autoRefresh() In live mode return whether data should be fetched automatically or manually.
refreshInterval() Returns the interval in which automatic refreshes are made.
setAutoRefresh(boolean) In live mode set whether data should be fetched automatically or manually.
setRefreshInterval(Interval) Set the interval in which automatic refreshes are made.

Constructors

- **RefreshConfiguration**

  ```
  public RefreshConfiguration()
  ```

  - **Description**

    Default constructor

Methods

- **autoRefresh**

  ```
  public boolean autoRefresh()
  ```

  - **Description**

    In live mode return whether data should be fetched automatically or manually. In historic mode return whether in loop mode the time stamp should be refreshed automatically or manually.

  - **Returns** – in live mode whether data should be fetched automatically or manually and In historic mode whether in loop mode the time stamp should be refreshed automatically or manually.

- **refreshInterval**

  ```
  public float refreshInterval()
  ```

  - **Description**

    Returns the interval in which automatic refreshes are made.

  - **Returns** – the interval in which automatic refreshes are made.

- **setAutoRefresh**

  ```
  public void setAutoRefresh(boolean bool)
  ```

  - **Description**

    In live mode set whether data should be fetched automatically or manually. In historic mode set whether in loop mode the time stamp should be refreshed automatically or manually.

  - **Parameters**

    * `bool` –

- **setRefreshInterval**

  ```
  public void setRefreshInterval(Interval inv)
  ```

  - **Description**

    Set the interval in which automatic refreshes are made.

  - **Parameters**

    * `inv` –

3.17.7 Class RefreshContext

Encapsulates the logic of switching between historical and live data mode and starting and stopping the loop mode. Through LiveRefreshConfiguration it also encapsulates whether live data is fetched automatically or manually and in which interval.

Declaration

```
public class RefreshContext
 extends java.lang.Object implements TimeOptionPanelObserver
```

Constructor summary

RefreshContext() Default constructor

Method summary

continueRoutine() Continue the current routine.
getLoopTimeFrame() Get the loop time frame.
getRefreshConfig() Get the RefreshConfiguration.
historicalDataMode() Switch to historical data mode.
liveDataMode() Switch to live data mode.
refresh(Date) Refresh the submitted time stamp depending on the TimeStampState by returning a new time stamp.
setLoopTimeFrame(TimeFrame) Set the start and end time point of the loop.
setRefreshState(RefreshState) Set the current refresh state.
stopRoutine() Stop the current routine.
timeOptionUpdate() Update the observer with the current TimeOptionPanel state.

Constructors

- **RefreshContext**

  ```
  public RefreshContext()
  ```

  - **Description**

    Default constructor

Methods

- **continueRoutine**

  ```
  public void continueRoutine()
  ```

- **Description**

  Continue the current routine.

- **getLoopTimeFrame**

  ```
  public TimeFrame getLoopTimeFrame()
  ```

  - **Description**

    Get the loop time frame.

  - **Returns** – the loop time frame.

- **getRefreshConfig**

  ```
  public RefreshConfiguration getRefreshConfig()
  ```

  - **Description**

    Get the RefreshConfiguration.

  - **Returns** – the RefreshConfiguration.

- **historicalDataMode**

  ```
  public void historicalDataMode()
  ```

  - **Description**

    Switch to historical data mode.

- **liveDataMode**

  ```
  public void liveDataMode()
  ```

  - **Description**

    Switch to live data mode.

- **refresh**

  ```
  public java.util.Date refresh(java.util.Date timeStamp)
  ```

  - **Description**

    Refresh the submitted time stamp depending on the TimeStampState by returning a new time stamp.

- **Parameters**
  - `timeStamp` –
- **Returns** – the submitted time stamp altered depending on the TimeStampState.

- **setLoopTimeFrame**

```
public void setLoopTimeFrame(TimeFrame loopTimeFrame)
```

  - **Description**

    Set the start and end time point of the loop.
  - **Parameters**
    - `loopTimeFrame` –

- **setRefreshState**

```
public void setRefreshState(RefreshState refreshState)
```

  - **Description**

    Set the current refresh state.
  - **Parameters**
    - `refreshState` –

- **stopRoutine**

```
public void stopRoutine()
```

  - **Description**

    Stop the current routine.

- **timeOptionUpdate**

```
public void timeOptionUpdate()
```

  - **Description**

    Update the observer with the current TimeOptionPanel state.

3.17.8 Class RefreshState

A state

Declaration

```
public class RefreshState
 extends java.lang.Object
```

All known subclasses

LoopRefreshState (in 3.17.5, page 147), LiveRefreshState (in 3.17.4, page 146), HistoricalRefreshState (in 3.17.3, page 144)

Constructor summary

RefreshState() Default constructor

Method summary

continueRoutine(RefreshContext) Continue the current routine.
historicalDataMode(RefreshContext) Switch to historical data mode.
liveDataMode(RefreshContext) Switch to live data mode.
refresh(Date) Refresh the the submitted time stamp depending on the TimeStampState by returning a new time stamp.
stopRoutine(RefreshContext) Stop the current routine.

Constructors

- **RefreshState**

  ```
  public RefreshState()
  ```

  - **Description**

    Default constructor

Methods

- **continueRoutine**

  ```
  public void continueRoutine(RefreshContext context)
  ```

  - **Description**

    Continue the current routine.
  - **Parameters**

    * `context` –

- **historicalDataMode**

```
public void historicalDataMode(RefreshContext context)
```

- **Description**

  Switch to historical data mode.

- **Parameters**

  * `context` –

- **liveDataMode**

```
public void liveDataMode(RefreshContext context)
```

- **Description**

  Switch to live data mode.

- **Parameters**

  * `context` –

- **refresh**

```
public java.util.Date refresh(java.util.Date timeStamp)
```

- **Description**

  Refresh the the submitted time stamp depending on the TimeStampState by returning a new time stamp.

- **Parameters**

  * `timeStamp` –

- **Returns** – the most up-to-date time stamp.

- **stopRoutine**

```
public void stopRoutine(RefreshContext context)
```

- **Description**

  Stop the current routine.

- **Parameters**

  * `context` –

3.17.9 Class TimeOptionPanel

An implementation of AbstractTimeOptionPanel.

Declaration

```
public class TimeOptionPanel
 extends View.TimeOption.AbstractTimeOptionPanel
```

Constructor summary

TimeOptionPanel() Default constructor

Constructors

- **TimeOptionPanel**

  ```
  public TimeOptionPanel()
  ```

  - **Description**

    Default constructor

Members inherited from class AbstractTimeOptionPanel

`View.TimeOption.AbstractTimeOptionPanel` (in 3.17.2, page 142)

- `public TimeFrame` **getLoopTimeframe**`()`
- `public RefreshContext` **getRefreshContext**`()`
- `public Date` **getTimeStamp**`()`
- `protected` **loopTimeFrame**
- `public void` **notify**`()`
- `protected` **refreshConfig**
- `public void` **setLoopTimeFrame**`(TimeFrame` **loopTimeFrame**`)`
- `public void` **setTimeStamp**`(java.util.Date` **timeStamp**`)`
- `public void` **subscribeObserver**`(TimeOptionPanelObserver` **observer**`)`
- `protected` **timeStamp**
- `public void` **unsubscribeObserver**`(TimeOptionPanelObserver` **observer**`)`

3.18 Package View.Util

Package Contents *Page*

Classes

3.18.1 Class ClusterID

A Cluster Identifier.

Declaration

```
public class ClusterID
 extends View.Util.Identifier
```

Constructor summary

ClusterID() Default constructor

Constructors

- **ClusterID**

  ```
  public ClusterID()
  ```

  - **Description**

    Default constructor

Members inherited from class Identifier

`View.Util.Identifier` (in 3.18.3, page 158)

- `public boolean` **equals**`(Identifier` **other**`)`

3.18.2 Class Date

Represents a specific point in time.

Declaration

```
public class Date
 extends java.lang.Object
```

Constructor summary

Date() Default constructor

Constructors

- **Date**

  ```
  public Date()
  ```

  - **Description**

    Default constructor

3.18.3 Class Identifier

Represents an identifier made up of a String.

Declaration

```
public class Identifier
 extends java.lang.Object
```

All known subclasses

SensorID (in 3.18.5, page 160), ClusterID (in 3.18.1, page 157)

Constructor summary

Identifier() Default constructor

Method summary

equals(Identifier) Returns whether this identifier is equal to the submitted identifier or not.

Constructors

- **Identifier**

  ```
  public Identifier()
  ```

  - **Description**

    Default constructor

Methods

- **equals**

  ```
  public boolean equals(Identifier other)
  ```

  - **Description**

    Returns whether this identifier is equal to the submitted identifier or not.
  - **Parameters**
    - `other` –
  - **Returns** –

3.18.4 Class Point

A point representing a location in (x,y) coordinate space, specified in float precision.

Declaration

```
public class Point
 extends java.lang.Object
```

Constructor summary

Point() Default constructor

Method summary

getX() Returns the x coordinate of this point.
getY() Returns the y coordinate of this point.

Constructors

- **Point**

  ```
  public Point()
  ```

  - **Description**

    Default constructor

Methods

- **getX**

  ```
  public float getX()
  ```

  - **Description**

    Returns the x coordinate of this point.
  - **Returns** –

- **getY**

  ```
  public float getY()
  ```

  - **Description**

    Returns the y coordinate of this point.
  - **Returns** –

3.18.5 Class SensorID

A Sensor Identifier.

Declaration

```
public class SensorID
 extends View.Util.Identifier
```

Constructor summary

SensorID() Default constructor

Constructors

- **SensorID**

  ```
  public SensorID()
  ```

  - **Description**

    Default constructor

Members inherited from class Identifier

`View.Util.Identifier` (in 3.18.3, page 158)

- `public boolean` **equals**`(Identifier` **other**`)`

3.18.6 Class TimeFrame

A period of time, specified by a start and end date.

Declaration

```
public class TimeFrame
 extends java.lang.Object
```

Constructor summary

TimeFrame() Default constructor

Method summary

getEnd() Returns the end date of this time frame.
getStart() Returns the start date of this time frame.

Constructors

- **TimeFrame**

  ```
  public TimeFrame()
  ```

  - **Description**
    Default constructor

Methods

- **getEnd**

  ```
  public Date getEnd()
  ```

  - **Description**
    Returns the end date of this time frame.
  - **Returns** –

- **getStart**

  ```
  public Date getStart()
  ```

  - **Description**
    Returns the start date of this time frame.
  - **Returns** –

Class Hierarchy

Classes

- java.lang.Object
  - Download.DownloadID (in 3.20.2, page 178)
  - Download.DownloadState (in 3.20.3, page 179)
    - Download.AlterableDownloadState (in 3.20.1, page 176)
  - Export.AbstractExporter (in 3.19.2, page 164)
    - Export.FileExporter (in 3.19.6, page 170)
  - Export.CSVWriterStrategy (in 3.19.3, page 165)
  - Export.ExportProperties (in 3.19.4, page 167)
  - Export.ExportStreamGenerator (in 3.19.5, page 169)
  - Export.FileExtension (in 3.19.7, page 171)
  - Export.FileType (in 3.19.8, page 172)
  - Export.FileTypesUtility (in 3.19.9, page 173)
  - Export.NetCDFWriterStrategy (in 3.19.10, page 175)
  - ExportDownloadCommunication.HttpServlet (in 3.21.4, page 185)
    - ExportDownloadCommunication.DownloadServlet (in 3.21.1, page 181)
    - ExportDownloadCommunication.ExportServlet (in 3.21.2, page 182)
    - ExportDownloadCommunication.FileExtensionServlet (in 3.21.3, page 184)
    - ExportDownloadCommunication.StatusServlet (in 3.21.5, page 186)

Interfaces

- Export.FileWriterStrategy (in 3.19.1, page 163)

3.19 Package Export

Package Contents *Page*

Interfaces

Classes

3.19.1 Interface FileWriterStrategy

Interface for the FileWriterStrategy classes. Realization of a Strategy to be able to swap out the way a File has to be saved.

Declaration

`public interface FileWriterStrategy`

All known subinterfaces

NetCDFWriterStrategy (in 3.19.10, page 175), CSVWriterStrategy (in 3.19.3, page 165)

All classes known to implement interface

NetCDFWriterStrategy (in 3.19.10, page 175), CSVWriterStrategy (in 3.19.3, page 165)

Method summary

saveToFile(KStream, FilePath) Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.

Methods

- **saveToFile**

  ```
  void saveToFile(KStream stream, FilePath path)
  ```

  - **Description**

    Creates a File as specified by the FilePath and saves the Data from the provided Kafka-Stream into it.

  - **Parameters**

    * `stream` – is the KStream, that should be exported to a File.
    * `path` – Is the FilePath, where the new File should be created.

3.19.2 Class AbstractExporter

Abstract Exporter of Data to a File.

Declaration

```
public class AbstractExporter
 extends java.lang.Object
```

All known subclasses

FileExporter (in 3.19.6, page 170)

Field summary

properties Contains the Properties of an Export Request.

Constructor summary

AbstractExporter() Default constructor

Method summary

createFile() Generates the File with the desired Data.
createFileInformation() Creates Information for that Export.

Fields

- `public ExportProperties` **properties**
  - Contains the Properties of an Export Request.

Constructors

- **AbstractExporter**

  ```
  public AbstractExporter()
  ```

  - **Description**

    Default constructor

Methods

- **createFile**

  ```
  public void createFile()
  ```

  - **Description**

    Generates the File with the desired Data.

- **createFileInformation**

  ```
  public DownloadID createFileInformation()
  ```

  - **Description**

    Creates Information for that Export. These Information will be used to identifie a File for the WebGUI, that gets the created DownloadID.
  - **Returns** – Is the DownloadID for the started Export.

3.19.3 Class CSVWriterStrategy

Implementation of the FileWriterStrategy interface for CSV files.

Declaration

```
public class CSVWriterStrategy
 extends java.lang.Object implements FileWriterStrategy
```

Constructor summary

CSVWriterStrategy() Default constructor

Method summary

saveToFile(KStream, FilePath) Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.
saveToFile(KStream, FilePath) Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.

Constructors

- **CSVWriterStrategy**

  ```
  public CSVWriterStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **saveToFile**

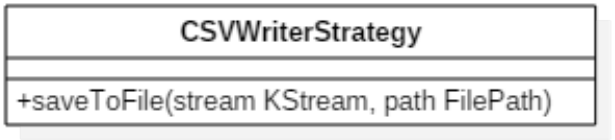
  ```
  public void saveToFile(KStream stream, FilePath path)
  ```

  - **Description**

    Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.

  - **Parameters**

    * `stream` – is the KStream, that should be exported to a File.
    * `path` – Is the FilePath, where the new File should be created.



- **saveToFile**

  ```
  public void saveToFile(KStream stream, FilePath path)
  ```

  - **Description**

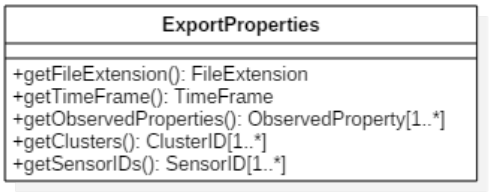
    Creates a File as specified by the FilePath and saves the Data from the provided Kafka-Stream into it.

  - **Parameters**

    * `stream` – is the KStream, that should be exported to a File.
    * `path` – Is the FilePath, where the new File should be created.

3.19.4 Class ExportProperties

Contains the Properties of an Export Request.



Declaration

```
public class ExportProperties
 extends java.lang.Object
```

Constructor summary

ExportProperties() Default constructor

Method summary

getClusters() Get the ClusterIDs that should be exported.
getFileExtension() Get the FileExtension for the Export File.
getObservedProperties() Get the ObservedProperties that should be exported.
getSensorIDs() Get the SensorIDs that should be exported.
getTimeFrame() Get the TimeFrame of the Data that should be exported.

Constructors

- **ExportProperties**

  ```
  public ExportProperties()
  ```

  - **Description**

    Default constructor

Methods

- **getClusters**

  ```
  public java.util.Set getClusters()
  ```

  - **Description**

    Get the ClusterIDs that should be exported. Always only exports a Groupd of Sensors or a Group of Clusters. The other Option is Empty.
  - **Returns** – The Clusters that should be taken in the Export.

- **getFileExtension**

  ```
  public FileExtension getFileExtension()
  ```

  - **Description**

    Get the FileExtension for the Export File.
  - **Returns** – The FileExtension for the File to export.

- **getObservedProperties**

  ```
  public java.util.Set getObservedProperties()
  ```

  - **Description**

    Get the ObsorvedProperties that should be exported.
  - **Returns** – The ObservedProperties that should be used for the export.

- **getSensorIDs**

  ```
  public java.util.Set getSensorIDs()
  ```

- **Description**

  Get the SensorIDs that should be exported. Always only exports a Groupd of Sensors or a Group of Clusters. The other Option is Empty.

- **Returns** – The SensorIDs of the Data that should be exported.

- **getTimeFrame**

```
public TimeFrame getTimeFrame()
```

  - **Description**

    Get the TimeFrame of the Data that should be exported.

  - **Returns** – The TimeFrame of the Data to be exported.

3.19.5 Class ExportStreamGenerator

Generates a Stream for the Export by asking for one at the PaVoS Core and Subscribing to it.

Declaration

```
public class ExportStreamGenerator
 extends java.lang.Object
```

Field summary

properties Contains the Properties of an Export Request.

Constructor summary

ExportStreamGenerator() Default constructor

Method summary

createExportStream() Asks for a KafkaStream and subscribes to it.

Fields

- `public ExportProperties` **properties**
  - Contains the Properties of an Export Request.

Constructors

- **ExportStreamGenerator**

  ```
  public ExportStreamGenerator()
  ```

  - **Description**

    Default constructor

Methods

- **createExportStream**

  ```
  public KStream createExportStream()
  ```

  - **Description**

    Asks for a KafkaStream and subscribes to it. Then gives it through to the needed part for the export.
  - **Returns** – Is a KStream of the Data that should be exported.

3.19.6 Class FileExporter

Exporter of Data from Kafka to a File.

Declaration

```
public class FileExporter
 extends Export.AbstractExporter
```

Constructor summary

FileExporter() Default constructor

Method summary

createFile() Generates the File with the desired Data.
createFileInformation() Creates Information for that Export.

Constructors

- **FileExporter**

  ```
  public FileExporter()
  ```

  - **Description**

    Default constructor

Methods

- **createFile**

  ```
  public void createFile()
  ```

  - **Description**

    Generates the File with the desired Data.

- **createFileInformation**

  ```
  public DownloadID createFileInformation()
  ```

  - **Description**

    Creates Information for that Export. These Information will be used to identifie a File for the WebGUI, that gets the created DownloadID.
  - **Returns** – Is the DownloadID for the started Export.

Members inherited from class AbstractExporter

`Export.AbstractExporter` (in 3.19.2, page 164)

- `public void` **createFile()**
- `public DownloadID` **createFileInformation()**
- `public` **properties**

3.19.7 Class FileExtension

Represents the FileExtension of a File. Is used to match the right FileFormat for an export or import.

Declaration

```
public class FileExtension
 extends java.lang.Object
```

Constructor summary

FileExtension() Default constructor

Constructors

- **FileExtension**

  ```
  public FileExtension()
  ```

  - **Description**

    Default constructor

3.19.8 Class FileType

Is used to store a FileExtension information and give the right FileWriter for this FileExtension.

Declaration

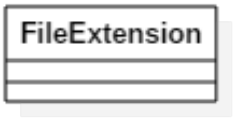
```
public class FileType
 extends java.lang.Object
```

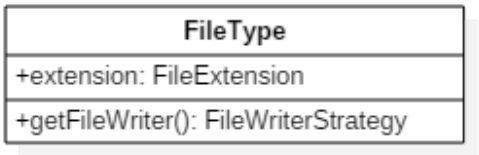
Field summary

extension The FileExtension is defining the FileType.

Constructor summary

FileType() Default constructor





Method summary

getFileWriter() Gives an instance of the implemented FileWriter that is associated with this FileType, thus this FileExtension.

Fields

- `public FileExtension` **extension**
  - The FileExtension is defining the FileType.

Constructors

- **FileType**

  ```
  public FileType()
  ```

  - **Description**

    Default constructor

Methods

- **getFileWriter**

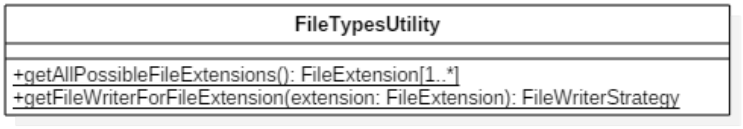
  ```
  public FileWriterStrategy getFileWriter()
  ```

  - **Description**

    Gives an instance of the implemented FileWriter that is associated with this FileType, thus this FileExtension. To do so it uses the static method getFileWriterForFileExtension from the FileTypesUtility class.
  - **Returns** – Is a new instance of an implementation of a FilWriterStrategy.

3.19.9 Class FileTypesUtility

Utility class that provides static methods to get all supported FileExtensions and one to get a new Instance of the FileWriter associated with a given FileExtension. If a new FileWriter is added to PaVoS, this class needs some changed to be able to return the new FileWriter.



Declaration

```
public class FileTypesUtility
 extends java.lang.Object
```

Constructor summary

FileTypesUtility() Default constructor

Method summary

getAllPossibleFileExtensions() Gives all supported FileExtensions in an ArrayList.
getFileWriterForFileExtension(FileExtension) Gives a new Instance of the FileWriter associated witha given FileExtension.

Constructors

- **FileTypesUtility**

  ```
  public FileTypesUtility()
  ```

  - **Description**

    Default constructor

Methods

- **getAllPossibleFileExtensions**

  ```
  public static java.util.Set getAllPossibleFileExtensions()
  ```

  - **Description**

    Gives all supported FileExtensions in an ArrayList.
  - **Returns** – Is an Array of the possible FileExtensions for an Export.

- **getFileWriterForFileExtension**

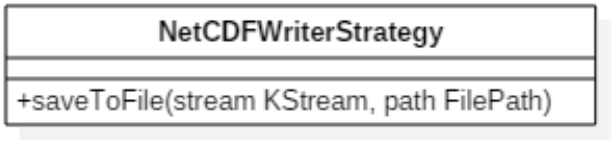
  ```
  public static FileWriterStrategy getFileWriterForFileExtension(
      FileExtension extension)
  ```

  - **Description**

    Gives a new Instance of the FileWriter associated witha given FileExtension.
  - **Parameters**
    - `extension` – Is the FileExtension for which a new instance of an Implementation of the FileWriterStrategy is wanted.
  - **Returns** – Is the instance of the implementation of a FileWriterStrategy.

3.19.10 Class NetCDFWriterStrategy

Implementation of the FileWriterStrategy interface for NetCDF files.



Declaration

```
public class NetCDFWriterStrategy
 extends java.lang.Object implements FileWriterStrategy
```

Constructor summary

NetCDFWriterStrategy() Default constructor

Method summary

saveToFile(KStream, FilePath) Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.
saveToFile(KStream, FilePath) Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.

Constructors

- **NetCDFWriterStrategy**

  ```
  public NetCDFWriterStrategy()
  ```

  - **Description**

    Default constructor

Methods

- **saveToFile**

  ```
  public void saveToFile(KStream stream,FilePath path)
  ```

  - **Description**

    Creates a File as specified by the FilePath and saves the Data from the provided KafkaStream into it.

- **Parameters**
  - `stream` – is the KStream, that should be exported to a File.
  - `path` – Is the FilePath, where the new File should be created.

- **saveToFile**

```
public void saveToFile(KStream stream, FilePath path)
```

  - **Description**

    Creates a File as specified by the FilePath and saves the Data from the provided Kafka-Stream into it.

  - **Parameters**
    - `stream` – is the KStream, that should be exported to a File.
    - `path` – Is the FilePath, where the new File should be created.

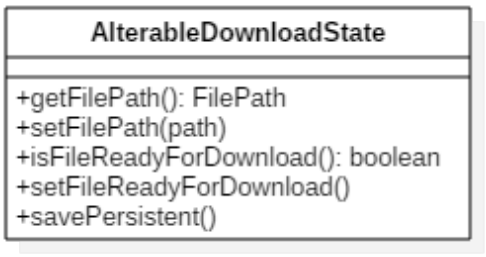
3.20 Package Download

Package Contents *Page*

Classes

3.20.1 Class AlterableDownloadState

Verifies for the State of a Download. Can also change it.



Declaration

```
public class AlterableDownloadState
 extends Download.DownloadState
```

Constructor summary

AlterableDownloadState() Default constructor

Method summary

getFilePath() Gives the FilePath associated with this DownloadID.
isFileReadyForDownload() Checks if a File is Ready to be downloaded.
savePersistent() Save the changed Data persistently.
setFilePath(void) Defines the FilePath for the DownloadID.
setFileReadyForDownload() Validate, that the File is ready to be downloaded.

Constructors

- **AlterableDownloadState**

  ```
  public AlterableDownloadState()
  ```

  - **Description**

    Default constructor

Methods

- **getFilePath**

  ```
  public FilePath getFilePath()
  ```

  - **Description**

    Gives the FilePath associated with this DownloadID.
  - **Returns** – The FilePath of the File for the Download.

- **isFileReadyForDownload**

  ```
  public boolean isFileReadyForDownload()
  ```

  - **Description**

    Checks if a File is Ready to be downloaded.
  - **Returns** – A boolean whether the file is downloadable or not.

- **savePersistent**

  ```
  public void savePersistent()
  ```

  - **Description**

    Save the changed Data persistently.

- **setFilePath**

  ```
  public void setFilePath(void path)
  ```

  - **Description**

    Defines the FilePath for the DownloadID.

  - **Parameters**

    * `path` – Is the FilePath to be set.

- **setFileReadyForDownload**

  ```
  public void setFileReadyForDownload()
  ```

  - **Description**

    Validate, that the File is ready to be downloaded.

Members inherited from class DownloadState

`Download.DownloadState` (in 3.20.3, page 179)

- `public` **downloadID**
- `public FilePath` **getFilePath()**
- `public boolean` **isFileReadyForDownload()**

3.20.2 Class DownloadID

Is an Identifier for a specific Download, so that the right file can be fount for a requeststed Download.

Declaration

```
public class DownloadID
 extends java.lang.Object
```

Constructor summary

DownloadID() Default constructor

Constructors

- **DownloadID**

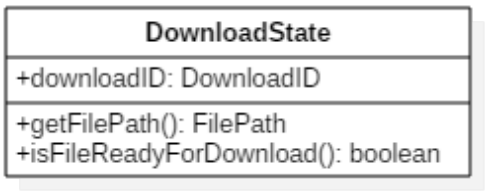
  ```
  public DownloadID()
  ```

  - **Description**

    Default constructor

3.20.3 Class DownloadState

Verifies for the State of a Download.



Declaration

```
public class DownloadState
 extends java.lang.Object
```

All known subclasses

AlterableDownloadState (in 3.20.1, page 176)

Field summary

downloadID Is an Identifier for a specific Download.

Constructor summary

DownloadState() Default constructor

Method summary

getFilePath() Gives the FilePath associated with this DownloadID.
isFileReadyForDownload() Checks if a File is Ready to be downloaded.

Fields

- `public DownloadID` **downloadID**
  - Is an Identifier for a specific Download.

Constructors

- **DownloadState**

  public DownloadState()

  - **Description**

    Default constructor

Methods

- **getFilePath**

  public FilePath getFilePath()

  - **Description**

    Gives the FilePath associated with this DownloadID.
  - **Returns** – The FilePath of the File for the Download.

- **isFileReadyForDownload**

  public boolean isFileReadyForDownload()

  - **Description**

    Checks if a File is Ready to be downloaded.
  - **Returns** – A boolean whether the file is downloadable or not.

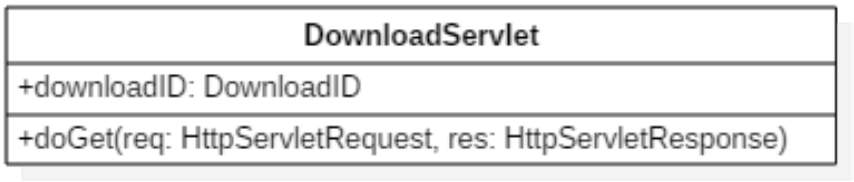
3.21 Package ExportDownloadCommunication

Package Contents *Page*

Classes

3.21.1 Class DownloadServlet

Servlet to let the WebGUI download a finished Export.



Declaration

```
public class DownloadServlet
 extends ExportDownloadCommunication.HttpServlet
```

Field summary

downloadID Is an Identifier for a specific Download.

Constructor summary

DownloadServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Handles a GET request by sending the desired File to the WebGUI.

Fields

- `public DownloadID` **downloadID**
  - Is an Identifier for a specific Download.

Constructors

- **DownloadServlet**

  ```
  public DownloadServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```

  - **Description**

    Handles a GET request by sending the desired File to the WebGUI.
  - **Parameters**
    - `req` – Is the HttpServletRequest.
    - `res` – Is the HttpServletResponse.

Members inherited from class HttpServlet

`ExportDownloadCommunication.HttpServlet` (in 3.21.4, page 185)

- `public void` **doGet**`(HttpServletRequest req, HttpServletResponse res)`

3.21.2 Class ExportServlet

HttpServlet to get a Dataexport request from the WebGUI.

Declaration

```
public class ExportServlet
 extends ExportDownloadCommunication.HttpServlet
```

Field summary

properties Contains the Properties of an Export Request.

Constructor summary

ExportServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Handles a GET request by starting the export of the desired Data.

Fields

- `public ExportProperties` **properties**
  - Contains the Properties of an Export Request.

Constructors

- **ExportServlet**

  ```
  public ExportServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```

  - **Description**

    Handles a GET request by starting the export of the desired Data. At the same time a DownloadID is sent back to the WebGUI, so that it can check for the File.

  - **Parameters**
    - `req` – Is the HttpServletRequest.
    - `res` – Is the HttpServletResponse.

Members inherited from class HttpServlet

`ExportDownloadCommunication.HttpServlet` (in 3.21.4, page 185)

- `public void` **doGet**`(HttpServletRequest` **req**`, HttpServletResponse` **res**`)`

3.21.3 Class FileExtensionServlet

Servlet, to let the WebGUI ask for the available FileExtensions for the Export.

Declaration

```
public class FileExtensionServlet
 extends ExportDownloadCommunication.HttpServlet
```

Constructor summary

FileExtensionServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Handles a GET request by sending Information about the available FileExtensions.

Constructors

- **FileExtensionServlet**

  ```
  public FileExtensionServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```

- **Description**

  Handles a GET request by sending Information about the available FileExtensions.

- **Parameters**
  - `req` – Is the HttpServletRequest.
  - `res` – Is the HttpServletResponse.

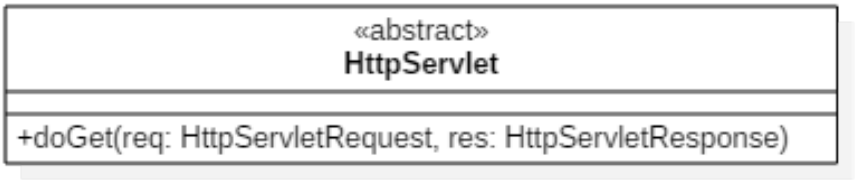
Members inherited from class HttpServlet

`ExportDownloadCommunication.HttpServlet` (in 3.21.4, page 185)

- `public void` **doGet**`(HttpServletRequest req, HttpServletResponse res)`

3.21.4 Class HttpServlet

Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site. (javax.servlet.http.HttpServlet)



Declaration

```
public class HttpServlet
 extends java.lang.Object
```

All known subclasses

StatusServlet (in 3.21.5, page 186), FileExtensionServlet (in 3.21.3, page 184), ExportServlet (in 3.21.2, page 182), DownloadServlet (in 3.21.1, page 181)

Constructor summary

HttpServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Called by the server (via the service method) to allow a servlet to handle a GET request.

Constructors

- **HttpServlet**

```
public HttpServlet()
```

  - **Description**
    Default constructor

Methods

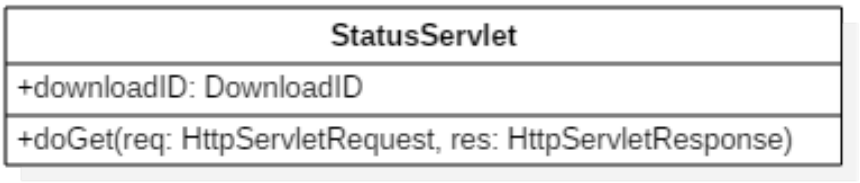
- **doGet**

```
public void doGet(HttpServletRequest req,HttpServletResponse res)
```

  - **Description**
    Called by the server (via the service method) to allow a servlet to handle a GET request.
  - **Parameters**
    * `req` – Is the HttpServletRequest.
    * `res` – Is the HttpServletResponse.

3.21.5 Class StatusServlet

Servlet to let the WebGUI check if a Download is ready.



Declaration

```
public class StatusServlet
 extends ExportDownloadCommunication.HttpServlet
```

Field summary

downloadID Is an Identifier for a specific Download.

Constructor summary

StatusServlet() Default constructor

Method summary

doGet(HttpServletRequest, HttpServletResponse) Handles a GET request by checking the availability of the desired download.

Fields

- `public DownloadID` **downloadID**
  - Is an Identifier for a specific Download.

Constructors

- **StatusServlet**

  ```
  public StatusServlet()
  ```

  - **Description**

    Default constructor

Methods

- **doGet**

  ```
  public void doGet(HttpServletRequest req,HttpServletResponse res)
  ```

  - **Description**

    Handles a GET request by checking the availability of the desired download.
  - **Parameters**
    - `req` – Is the HttpServletRequest.
    - `res` – Is the HttpServletResponse.

Members inherited from class HttpServlet

`ExportDownloadCommunication.HttpServlet` (in 3.21.4, page 185)

- `public void` **doGet(`HttpServletRequest` req, `HttpServletResponse` res)**