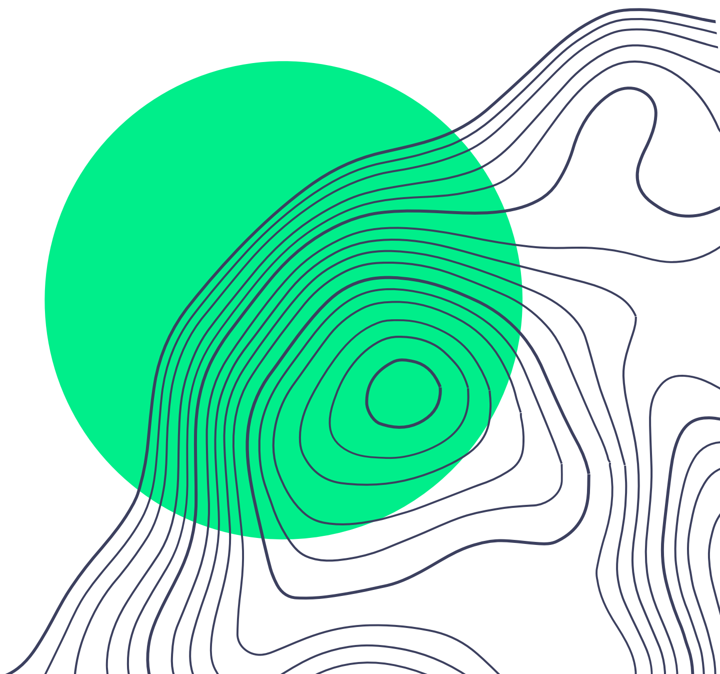
Next generation integration software

Manual

gravity.integration

gravity-integration.com

TABLE OF CONTENTS



1. CONTEXT

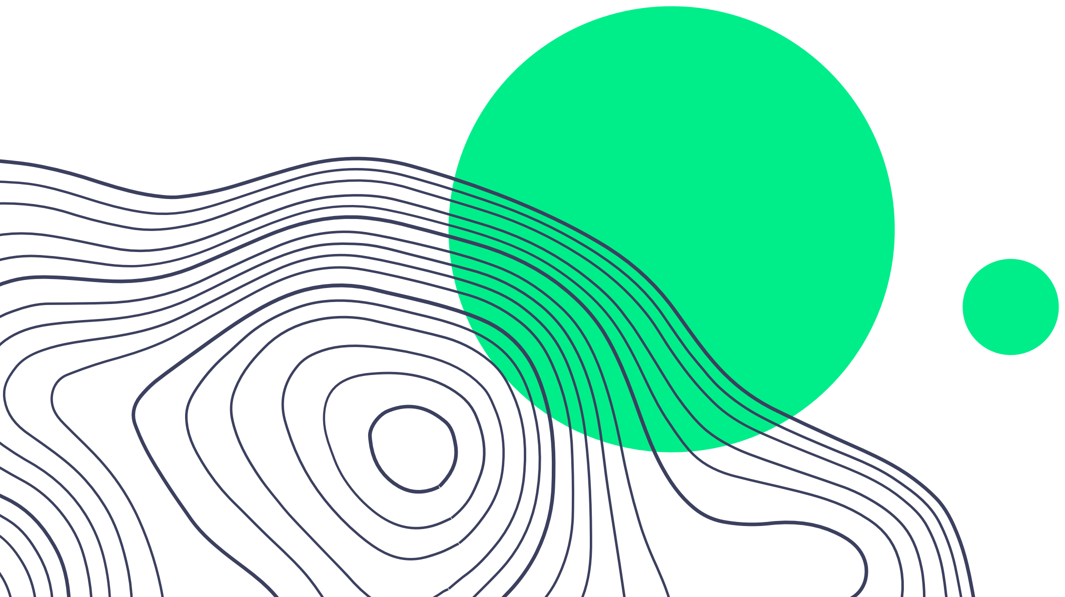
IT support of an enterprise is developing in two directions: technological advancement is increasing and the range and functional complexity of software is constantly changing.

The reflexion above generates the following problem: how to increase IT involvement in running a business in a graduate way, without freezing functional solutions, but to react flexibly on new processes and modern, more efficient technologies.

We have no doubts, that constant software exchange, as it happens in case of one software operating all areas, or being dependent on only one IT supplier, generates huge amount of work and financial resources.

In our opinion the most effective IT strategy, therefore the most scaled and the cheapest, is to use an "open" and multilayer software, capable to communicate with other domain solutions, which can be integrated in any way possible. Thanks to this the effectiveness of the used software is not limited by the weakest element effectiveness, but is a sum of potentials of each and every element of the IT system.

Let us use the following metaphor: individual IT solutions are areas of a brain, which are responsible for a – sometimes narrow - range of "services" or functions. However when connected with a network of neurons (which creates the nervous system), they have a potential that multiply exceeds the sum of potentials of every individual area of the brain.



2. THE PURPOSE OF GRAVITY SOFTWARE

GRAVITY is a software that supports the integration of many different development environments. In GRAVITY environment you can prepare projects of data extraction, data transfer, integrate data from other IT environments, process, standardize and validate the data.

GRAVITY is a data bus IT environment, where by using available operators, you can model data transfer process and create transforming mechanisms, which are also beyond the imagination of GRAVITY developers.

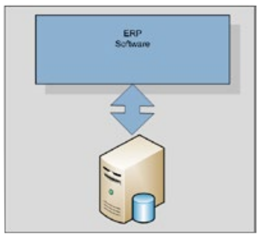
We suggest to imagine GRAVITY as a huge data bus, designed freely, capable of branching out in any direction, with terabytes of data rushing in every direction. Data, that during the transfer, can be controlled, validated, processed.

Referring to readers imagination and the metaphor quoted in the Context chapter, if a complete IT system of an enterprise is “a brain”, than GRAVITY can become a “neuron” or “neurons network” transferring the information between the areas of “the brain”.

3. GRAVITY AS A LAYER OF AN IT SYSTEM

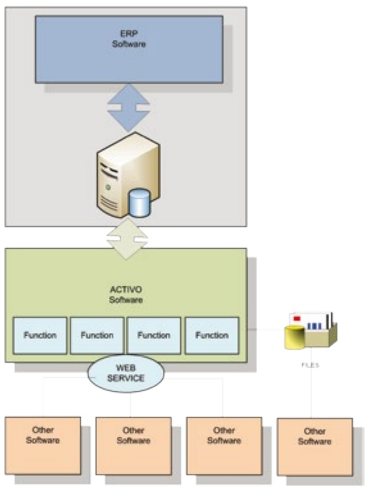
GRAVITY is an environment that facilitates delayering of complex IT systems. To picture the influence of an IT system delayering on growth potential, we present the following diagrams.

3.1 SOLUTION DIAGRAM WITH MINIMAL GROWTH POTENTIAL



- Software is integrated with the database (software contains all access logic and data handling)
- Use of data included in the database leads only through ERP system functions.
- Even the smallest expansion of ERP system functions requires the exchange of whole ERP software.
- Big dependency on one IT supplier (ERP system supplier)

SOLUTION DIAGRAM WITH GREAT GROWTH POTENTIAL



- Once built data access layer can be used multiple times for different applications.
- When developing specialized software it will be possible to focus on the interface: GRAVITY layer will be responsible for data extraction, validation, transfer.
- There is a division of competences: people with various qualifications will take care of the project in GRAVITY and the project of creating andedicated application.

CONCLUSION

i If an IT system of the enterprise is multi-layer, it is then stronger, more efficient and more evolutionary than IT solutions with fewer numbers of layers.

4. WHO IS GRAVITY SOFTWARE FOR

GRAVITY is an IT environment for:

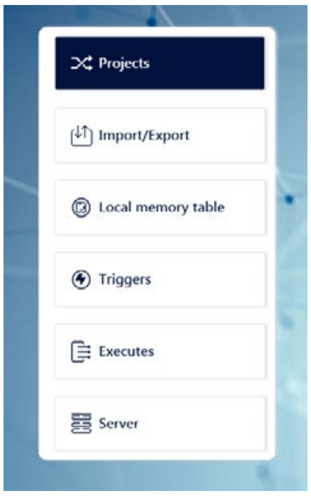
IT INTEGRATORS, COMPANIES AND SERVICES

- Software developers do not need to create a software layer responsible for integration (integration layer can be modeled in GRAVITY)
- Software developers can reuse multiple times an GRAVITY application in various enterprises.
- Integration layer modeled in GRAVITY will allow effective exchange of front-end and back-end applications;

ENTERPRISES

- IT services can use GRAVITY as an internal data bus, connecting ERP and domain solutions.

5. GRAVITY ENVIRONMENT FUNCTIONAL DESCRIPTION



GRAVITY Main Menu consists of the following options.

Below we present to you a detailed description of each option in terms of purpose and application.

5.1 PROJECTS

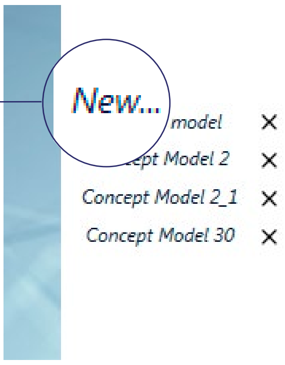
In this option you will create GRAVITY application. You can declare any number of projects (GRAVITY applications), for various usage. Individual projects can be independent, however there is a possibility to bind them together, for example the result of one project can be on bus input for other GRAVITY project designed by you.

Creating an GRAVITY project is simple and intuitive. It is enough that you drag and drop it on a working field, and lead a bus between the operators. You can ramify the data transfer bus, you can multiply the operators and lead the processed data to specified locations (bases, tables, mail etc.)

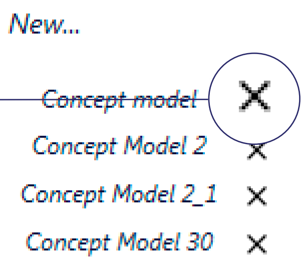
The range of functional operators, which you can use during configuring GRAVITY project is presented in DESCRIPTION OF GRAVITY OPERATORS chapter.

5.1.1 CREATING NEW GRAVITY PROJECT

If you want to create new GRAVITY application, use NEW option



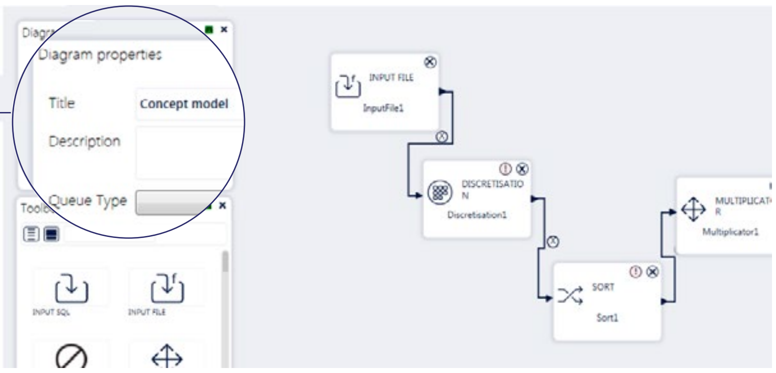
5.1.2 DELETING GRAVITY PROJECT



If you want to delete GRAVITY application, use X option.

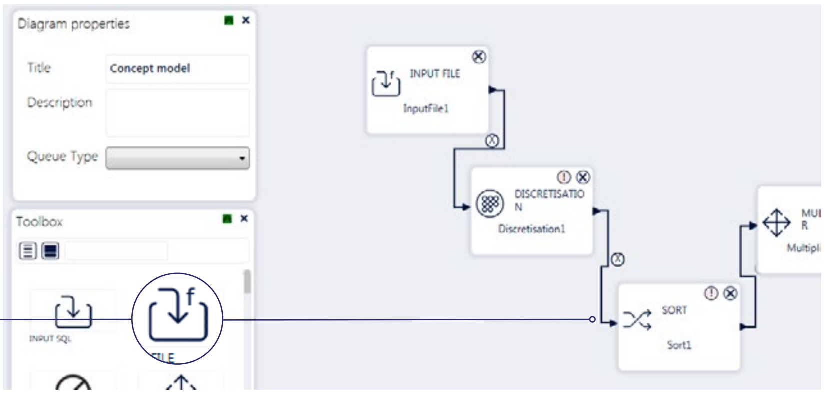
detailed description in terms of process queuing you can find in **GRAVITY PROCCES CALL**

5.1.3 EDITING GRAVITY APPLICATION FEATURES



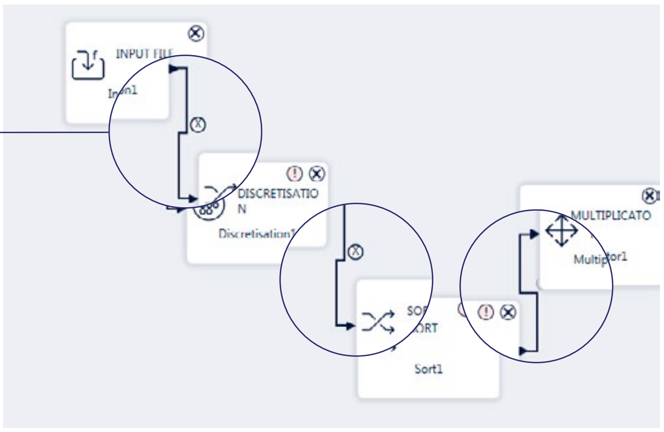
In the area DIAGRAM PROPERTIES GRAVITY application example you declare the title, project description and queuing method of the running process

5.1.4 OPERATOR SELECTION



Selected operator needs to be caught on TOOLBOX area, and dropped onto the sheet.

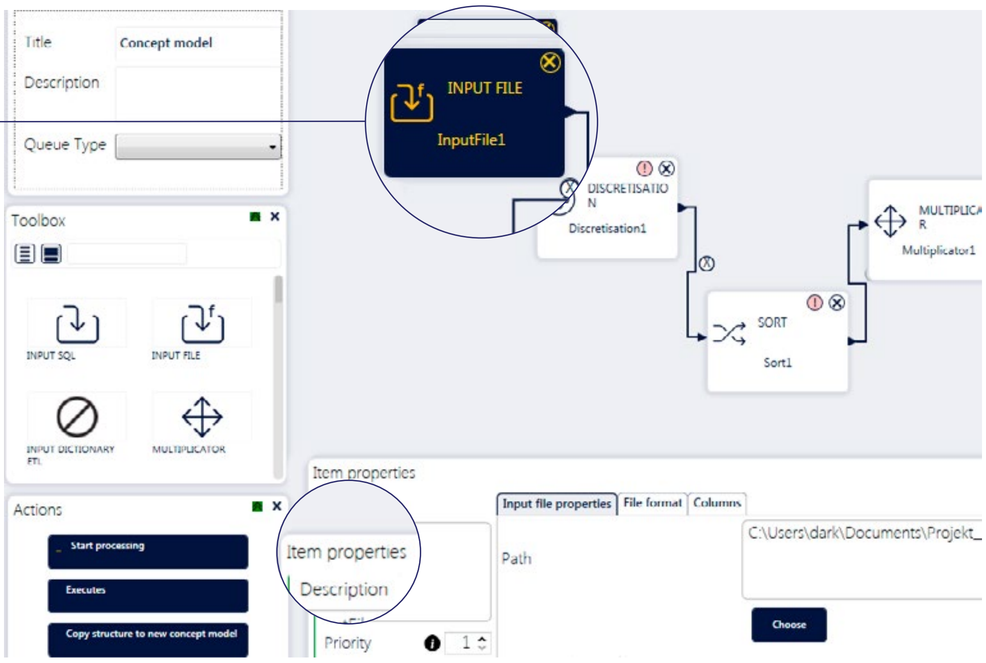
5.1.5 RUNNING A DATASTREAM BUS



We draw a data bus by connecting output of one operator with an input of next operator using mouse pointer. GRAVITY application example.

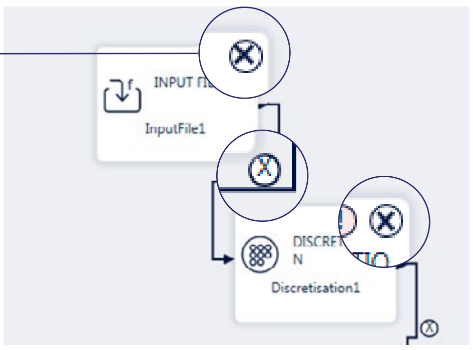
5.1.6 OPERATOR CONFIGURATION

Each operator has a set of specific parameters, which you fill in the ITEM PROPERTIES area. The selection of an operator to configure is performed by clicking the mouse pointer on its area.



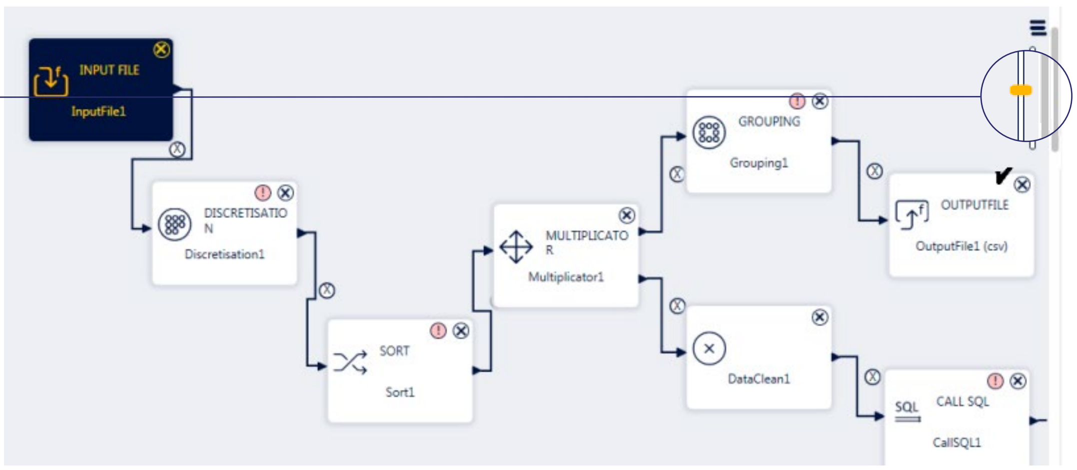
5.1.7 DELETING OPERATOR AND BUS

Each operator and bus can be deleted – select the delete icon.



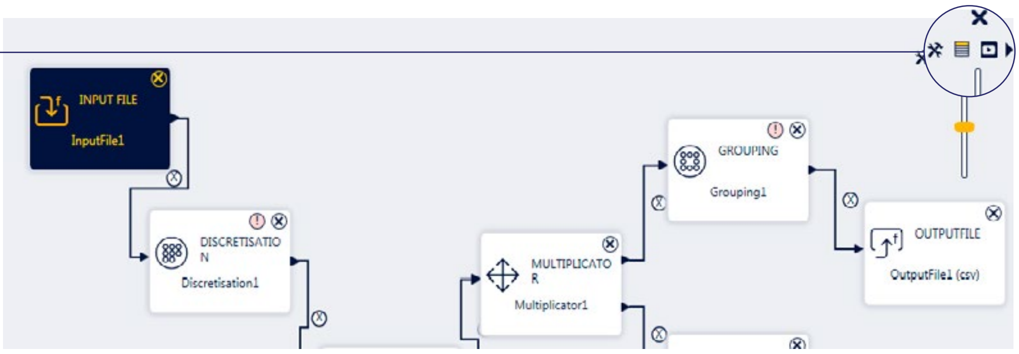
5.1.8 SHEET SCALING

For comfortable viewing, a sheet can be zoomed in and out.



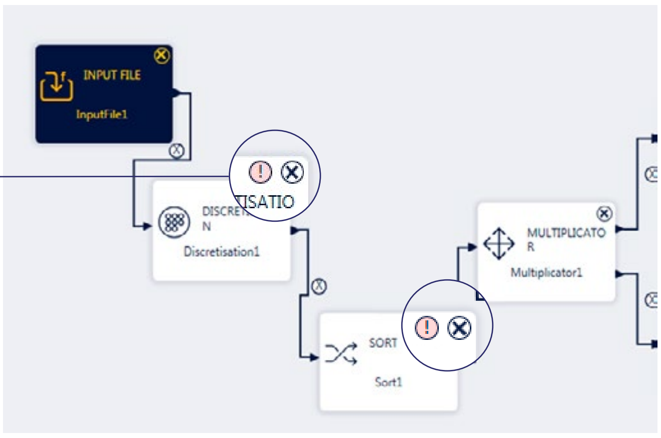
5.1.9 SWITCHING OFF WORKING AREAS

For comfortable viewing, you can switch off individual working areas (DIAGRAM PROPERTIES, TOOLBOX, ACTIONS, ITEM PROPERTIES).



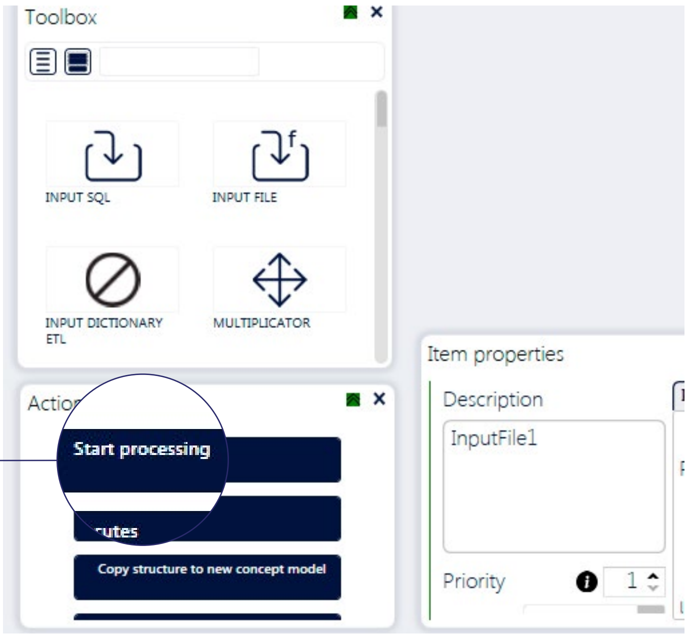
5.1.10 INCOMPLETE PROJECT SIGNAL

If an operator is not completely configured, "incomplete" icon appears on the sheet. Such project cannot be executed.

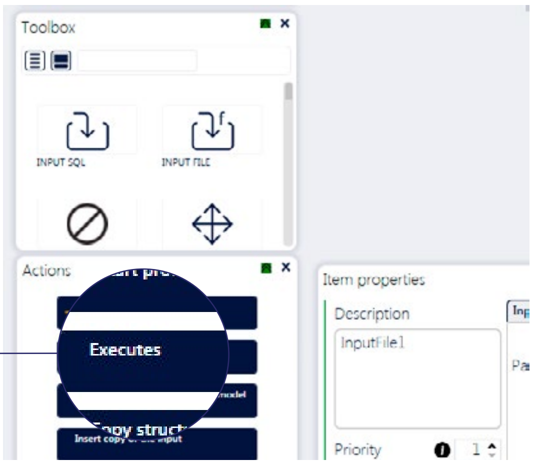


5.1.11 MANUAL PROJECT EXECUTION.

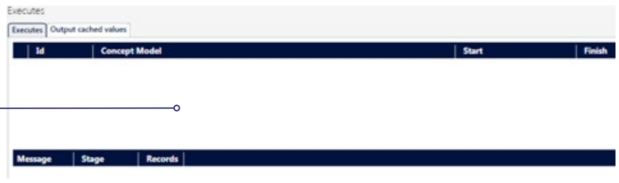
In ACTIONS area GRAVITY application example you can manually execute GRAVITY application. (detailed description in terms of GRAVITY application execution you can find in GRAVITY PROCCES CALL)



5.1.12 VIEWING OF ALL GRAVITY PROJECT CALLS

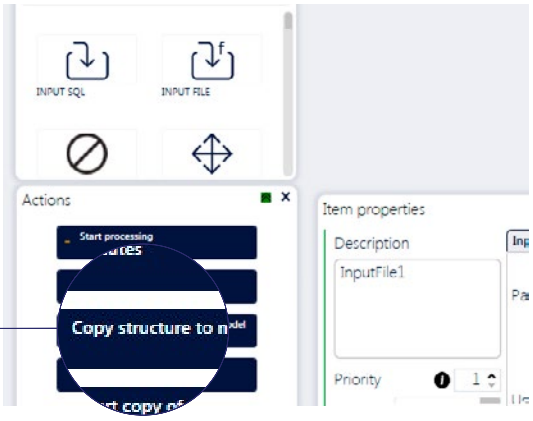


In ACTIONS area GRAVITY application example you can reach the index of all application calls,



with information on time of execution, results, warnings and errors (see picture below)

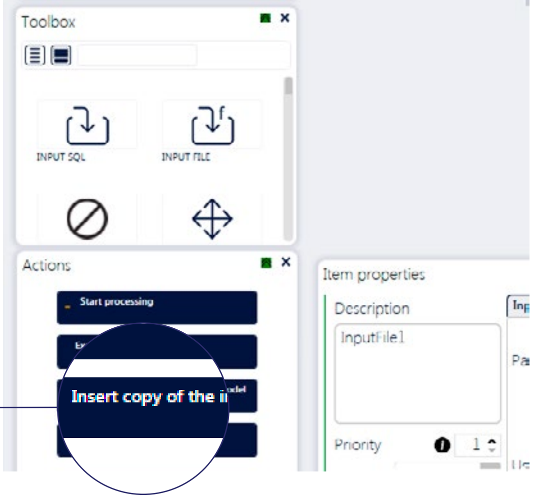
5.1.13 PROJECT SHEET COPYING



In ACTIONS area GRAVITY application example you can create new GRAVITY application by copying the current application.

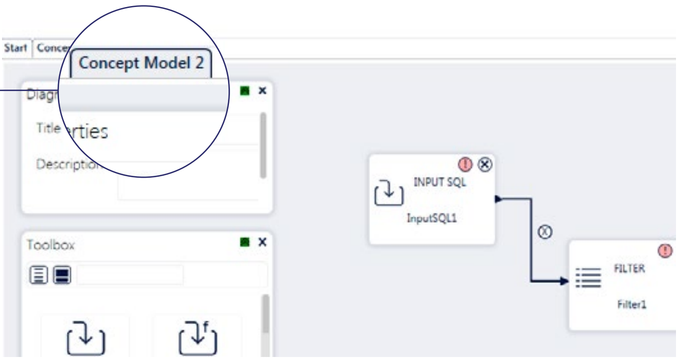
5.1.14 COPYING A CONFIGURED INPUT OPERATROR FROM ANOTHER SHEET

5.1.15 SIMULTANEOUS VIEW OF MULTIPLE GRAVITY PROJECTS

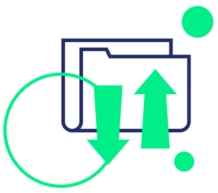


in ACTIONS area GRAVITY application example you can copy an INPUT operator from a selected different application.

5.2 IMPORT/EXPORT



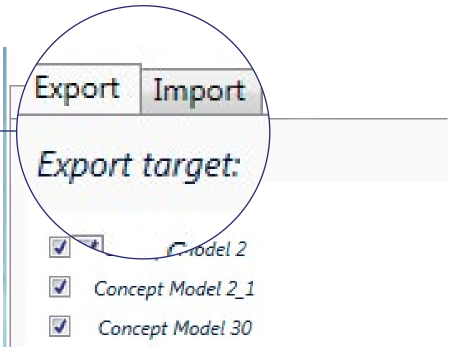
Each project is opened in a new tab and you can easily move between the projects by selecting a proper tab GRAVITY application example



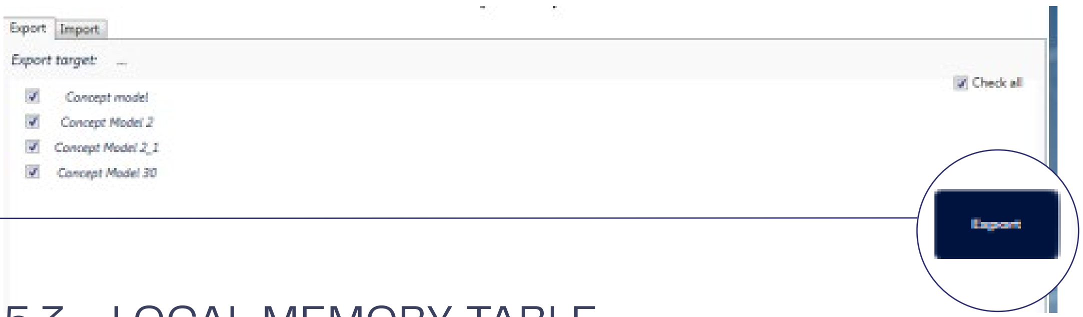
5.2.1 EXPORT

This function of the software allows you to export everything you have created, and import what has been created by others. The option enables cooperation between multiple creators.

5.2.2 IMPORT

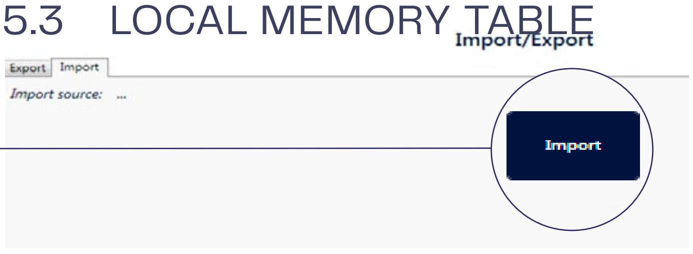


Files with applications selected for export will be stored in a location selected by you GRAVITY application example. Each file is in XML format.



Exporting action is initiated by EXPORT option GRAVITY application example

5.3 LOCAL MEMORY TABLE



If you want to import GRAVITY application, select an XML file containing application definition, (which for example was earlier exported from GRAVITY) and run IMPORT option GRAVITY application example

An option, that will allow you to see data placed in the memory and executed by GRAVITY during current session (that is since the last start of GRAVITY environment). This data can be used by external systems, while access to them is provided by WEB SERVICE function.

5.4 CALLS

The chapter "GRAVITY PROCESS CALL" describes in details how to execute GRAVITY process (application)

Let us focus then, that GRAVITY processes can be executed as follows:

- manually
- automatic by other GRAVITY application
- automatic by SCHEDULER function.
- automatic by database notification
- as a WEB SERVICE function via external software

Inside CALLS option you can configure the following GRAVITY call mechanisms:

- automatic by other GRAVITY application.
- automatic by SCHEDULER function.
- automatic by database notification.
- as a WEB SERVICE function via external software.

If you want to learn more on methods od using the options above – read the ACTIVE PROCESS CALL chapter

5.5 EXECUTES

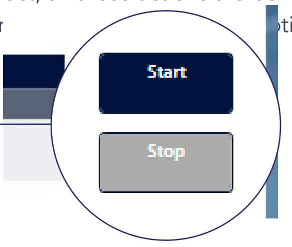
Preview of index of all GRAVITY applications executes, with detailed indication of application name, moment of execution, result, errors and messages.

5.6 SERVER

WEB SERVICE server. In CALLS options we have described multiple ways to run GRAVITY processes (applications). One of the way is to induce GRAVITY process, as a WEB SERVICE function using CALLS action.

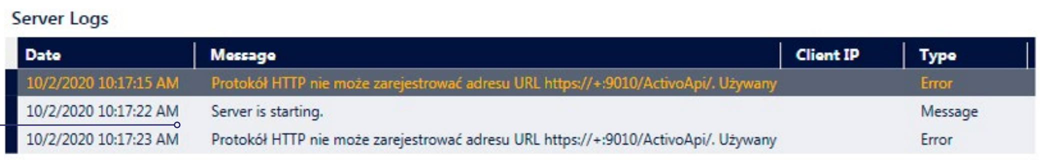
Please note, that you can define any number of CALLS actions, so you can define any number of GRAVITY applications and execute via external software as WEB SERVICE function.

In fact, all those actions are being trasferred through one WEB SERVICE function server, that is being cor tion: you have to configure port, user and password.



The server is run and stopped using START / STOP options – see picture below.

All server-related events are registered and presented on the list – see picture below.


Server Logs

Date	Message	Client IP	Type
10/2/2020 10:17:15 AM	Protokół HTTP nie może zarejestrować adresu URL https://+:9010/ActivoApi/. Używany		Error
10/2/2020 10:17:22 AM	Server is starting.		Message
10/2/2020 10:17:23 AM	Protokół HTTP nie może zarejestrować adresu URL https://+:9010/ActivoApi/. Używany		Error

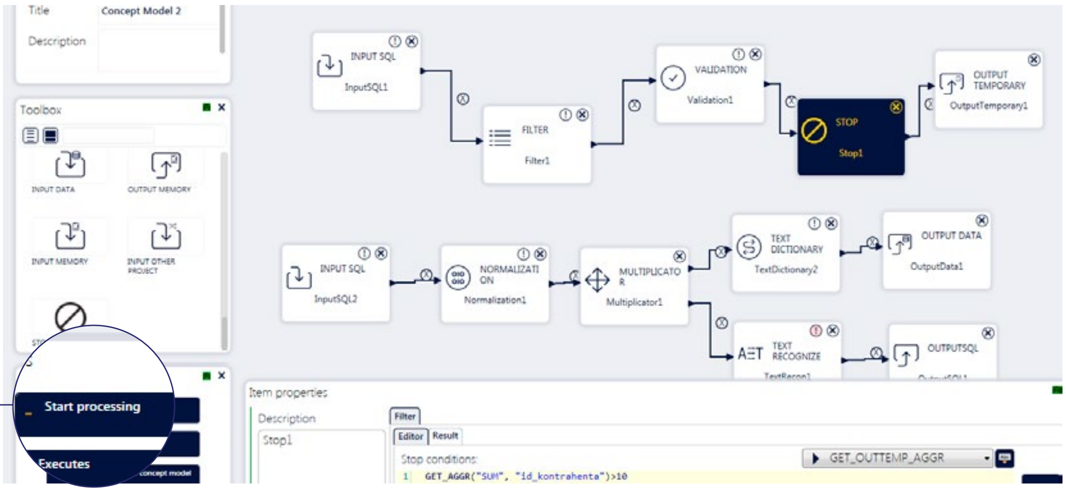
In terms of WEB SERVICE function execution, you probably came up with a question: how can an external software communicate with GRAVITY process? Be sure to read detailed description for INPUT DATA and OUTPUT DATA operators (chapter DESCRIPTION OF GRAVITY OPERATORS). Those operators are responsible for input and output data transfer.

6. GRAVITY PROCESS CALL

METHODS TO CALL GRAVITY PROCESS

- manually
- automatic by other GRAVITY application
- automatic by Scheduler
- automatic by database notification
- as a WEB SERVICE function via external software

6.1 MANUAL CALL

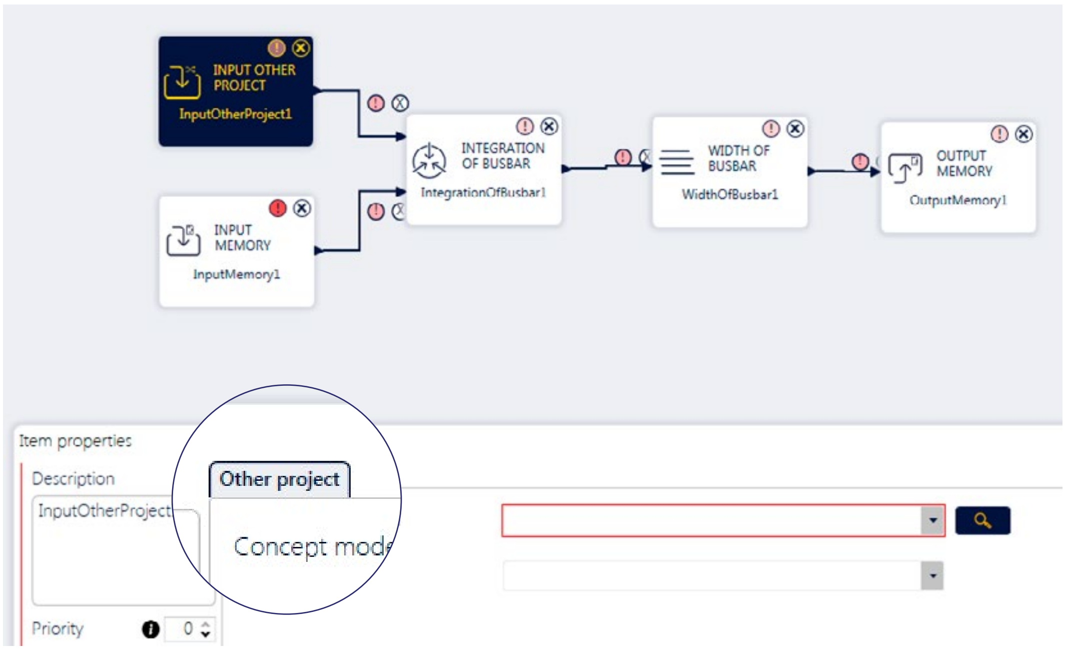


```
GET_AGGR("SUM", "id_kontrahenta")>10
```

On the picture above we have pointed out an option running a process, available in each application window. Each process action is remembered and available for inspection in EXECUTES option.

6.2 AUTOMATIC CALL BY OTHER GRAVITY APPLICATION

GRAVITY application can be executed by other GRAVITY application. On the illustration below you have an example of process diagram, which had INPUT OTHER PROJECT operator used (more information on parameters of this operator will be provided to you in GRAVITY OPERATORS DESCRIPTION chapter). The use od such operator causes, that other GRAVITY application associated (see operator parameters) with this operator is executed first.

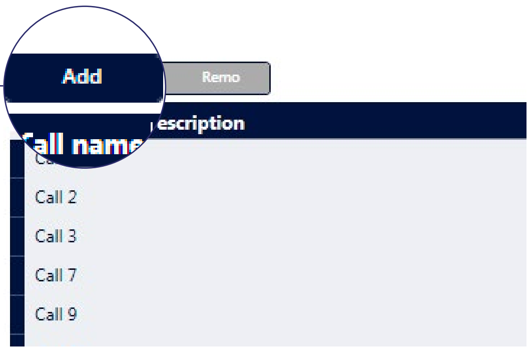


6.3 AUTOMATIC CALL BY SCHEDULER FUNCTION

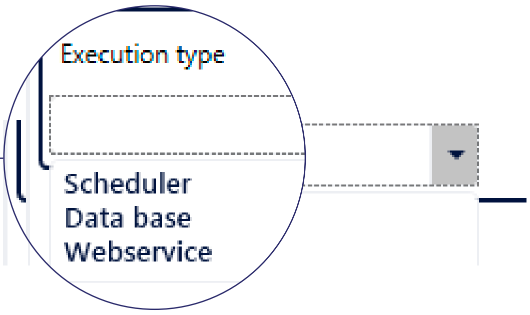
If you want to configure automatic execution using SCHEDULER, you need to select CALLS option in the Main Menu.

When deciding on time execution mechanism, you select SCHEDULER option from the list.

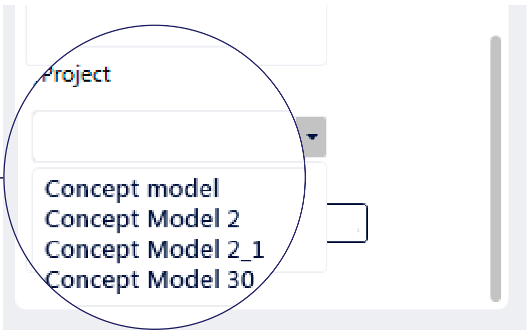
By choosing ADD, you can add new automatic GRAVITY application execution configuration. After choosing ADD option, you need to select the execution type from the list (see picture below)



In the PROPERTIES area you also need to choose GRAVITY application (project), which from now on will be associated with the configured call.



In order to finish the configuration, you will have to select a method to identify execute time.



In order to finish the configuration, you will have to select a method to identify execute time (see picture below). You can choose from:

- Specyfic date

Execute will take place only once, in the selected date and time.

- Every few minutes

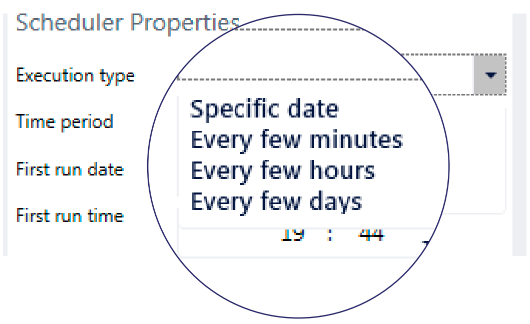
Execute will take place for the first time in the selected date and time, and will continue every declared number of minutes.

- Every few hours

Execute will take place for the first time in the selected date and time, and will continue every declared number of hours.

- Every few days

Execute will take place for the first time in the selected date and time, and will continue every declared number of days.

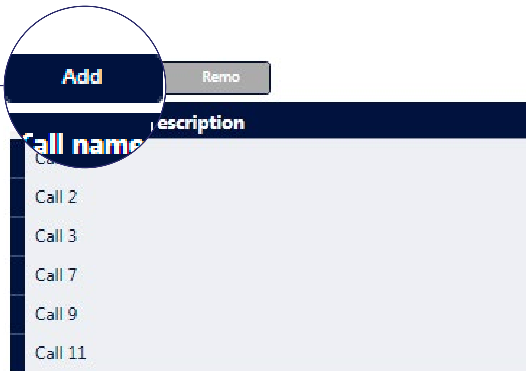


If you have a configured SCHEDULER type CALL action, GRAVITY software needs to remain up and running, thanks to this the system will start associated GRAVITY application (process) in the specific time or time intervals.

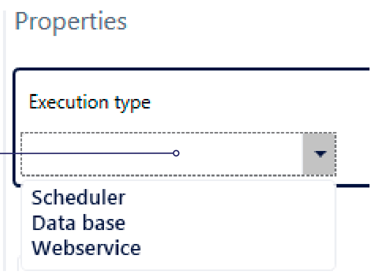
6.4 AUTOMATIC CALL BY DATABASE NOTIFICATION

If you want to configure application execution by database notification mechanism (or: based on a signal coming from a database), you need to select CALLS option in the Main Menu.

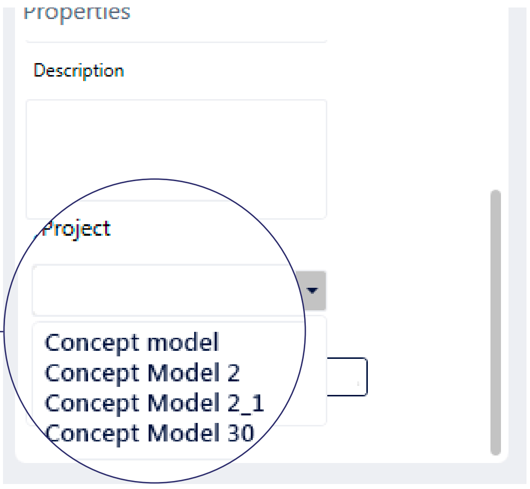
By choosing ADD you can add new automatic GRAVITY application configuration GRAVITY application example.

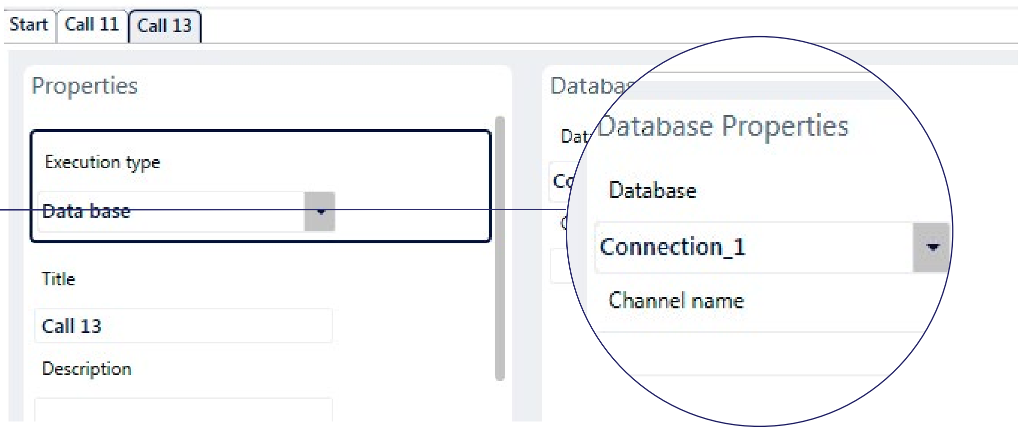


After choosing ADD option – you need to select the type of automatic call from the list (see picture below)



In the PROPERTIES area, you need to choose GRAVITY application (project), which from now on will be associated with the configured call (see picture below)





When deciding on database notification mechanism, you choose the option DATABASE from the list

In order to finish the configuration, you will also have to choose the connection with the database (selection from the list of configured connections – see picture below)

When deciding on execution mechanism by database notification, you choose DATABASE option from the list.

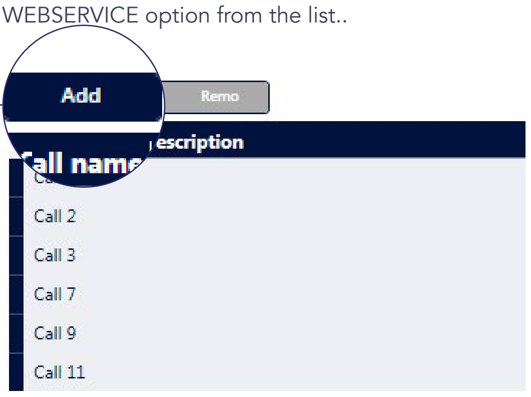
In order to finish the configuration, you will also have to choose the connection with the database (selection from the list of configured connections – see picture above) and a unique channel name to listen the database (the channel has to be declared earlier in the database and assigned to a certain event, which triggers the database notification action)

If you have a configured DATABASE type CALL action, GRAVITY software needs to remain up and running. GRAVITY is connected to the database and listens. If on a selected channel a signal appears (which means an event assigned to channel in the database occurred), the associated GRAVITY application is executed.

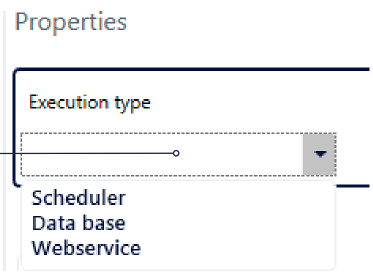
6.5 EXECUTION AS A WEB SERVICE FUNCTION VIA EXTERNAL SOFTWARE.

If you want to configure GRAVITY application execution via external software as WEB SERVICE function, you need to select CALLS option in the Main Menu.

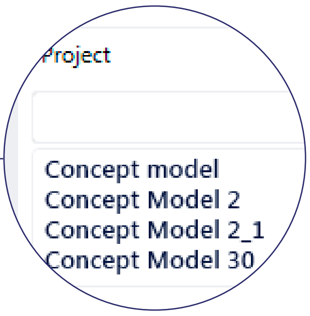
When deciding on mechanism of executing GRAVITY application as WEB SERVICE function, you choose WEBSERVICE option from the list..



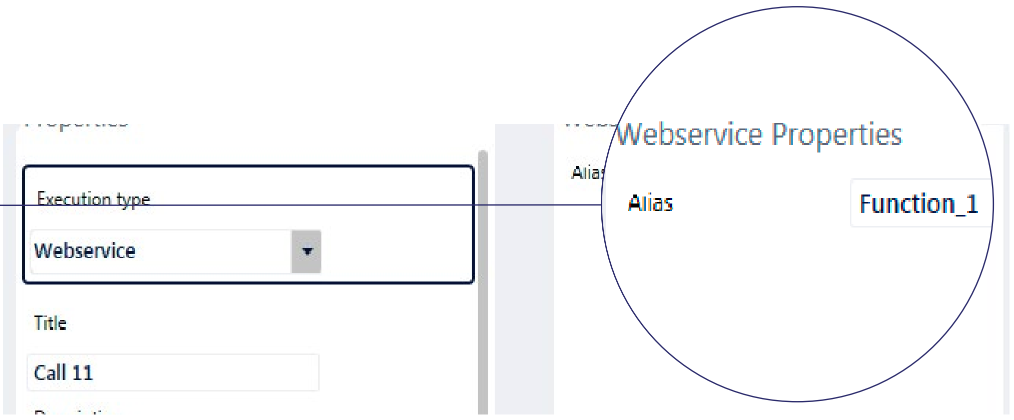
By selecting ADD option you can add new configuration of automatic GRAVITY application execution GRAVITY application example.



After choosing ADD option, you need to select the type of automatic execution from the list.



In the PROPERTIES area you also need to indicate GRAVITY application (project), which from now on will be associated with the configured execution.


Execution type: Webservice

Title: Call 11

Webservice Properties

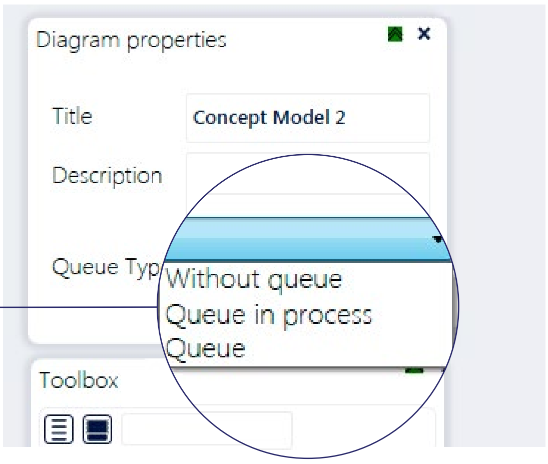
Alias: Function_1

To complete the configuration, you also need to create a unique action name (ALIAS field – see picture below)

If you have a configured WEBSERVICE type CALL action, you still need to configure the WEB SERVICE server (SERVER option). All WEBSERVICE type CALL actions will be available for the external software throughout this server. Action identification is based on a unique ALIAS name. Execution of each action results in launching the GRAVITY application (project) associated to this action.

6.5.1 GRAVITY PROCESSES QUEUING MANAGEMENT

The same or different GRAVITY applications can be executed multiple times within a short period of time. To manage those processes, you have a possibility to declare the method of task processing queuing.



In each process, in the DIAGRAM PROPERTIES area GRAVITY application example you can find QUEUE TYPE parameter. You can decide on the following actions

- Without queue – process executed immediately
- Queue in process – process being queued within the same executions of the same application.
- Queue – process queued.

7. TYPES OF GRAVITY OPERATORS

Do perceive GRAVITY application as a map of buses connecting multiple converters, which we call "operators". GRAVITY has a group of converters (operators), which allow to acquire an external data stream. The data stream can further circulate around the buses, be transformed, connected, identified, cleaned, branched, kept in memory for fast access, sent outside using special output operators. The size of the bus as well as number of operators used in the application can be unlimited. You also have at your disposal data stream flow triggering mechanisms: from automatic to manual mechanisms. In this chapter we describe functional actions of each operator.

To make your journey through GRAVITY world easier – we grouped the operators following the purpose principle.

7.1 INPUT OPERATORS

- INPUT DATA
- INPUT FILE
- INPUT MEMORY
- INPUT OTHER PROJECT
- INPUT SQL

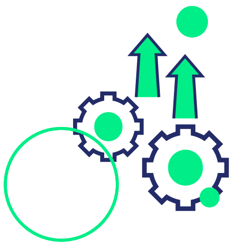
7.2 OUTPUT OPERATORS

- OUTPUT DATA
- OUTPUT FILE
- OUTPUT MEMORY
- OUTPUT PDF
- OUTPUT POST
- OUTPUT SQL
- OUTPUT TEMPORARY

7.3 PROCESSING OPERATORS

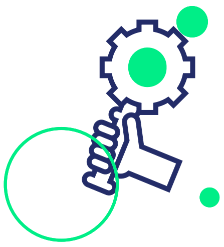
- BUSBAR
- COMPUTING
- DATA CLEAN
- DISCRETISATION
- FILTER
- GROUPING
- IF
- INTEGRATION OF BUSBAR
- MERGE OF BUSBAR
- MULTIPLICATOR

- NORMALIZATION
- OCR
- SORT
- STANDARIZATION
- STANDARIZATION OUTPUT
- STOP
- TEXT DICTIONARY
- TEXT RECOGNIZE
- WIDTH OF BUSBAR
- VALIDATION



7.4 OUTSIDE PROCESSING OPERATORS

- CALL EXE
- CALL FUNCTION DLL
- CALL SQL
- CALL WEB SERVICE



7.5 SELF-DESIGNED OPERATORS

Operators can also be created completely by yourself. Self-designed operators are present in the system on the same rules as built-in operators. Creating an operator is all about declaring the name and transferring code in GRAVITY environment. While coding you have access to data stream on the input bus and you point out the method of output stream transferring. The code is automatically compiled while calling GRAVITY application.

8. DESCRIPTION OF GRAVITY OPERATORS

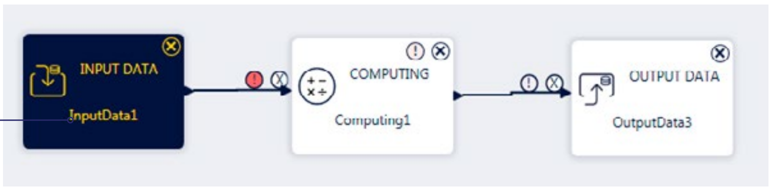
8.1 INPUT OPERATORS

8.1.1 INPUT DATA

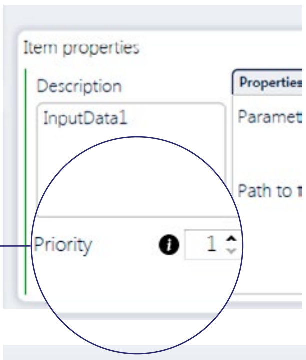
In context of GRAVITY application execution as WEB SERVICE function via an external software you have at your disposal two operators communicating with the interior of the project: INPUT DATA and OUTPUT DATA

INPUT DATA is an operator downloading data stream from a program executing GRAVITY project, as WEB SERVICE function, transferred to the project in JSON protocol. INPUT DATA can also set global parameter values present in GRAVITY application.

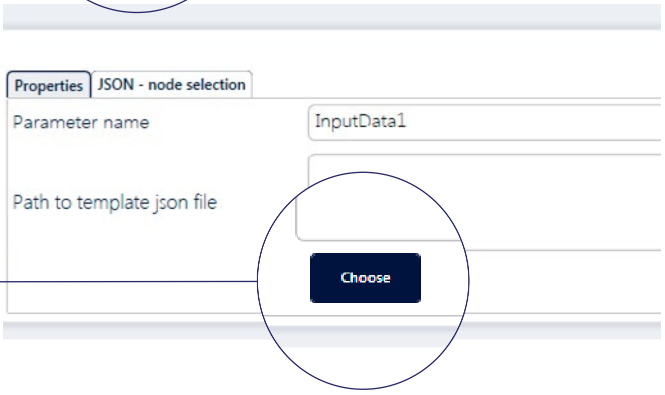
Because INPUT DATA is an input operator, and multiple input operators can exist inside the project, you can declare the order of initial operators execution by editing PRIORITY parameter.



GRAVITY project example using INPUT DATA operator.



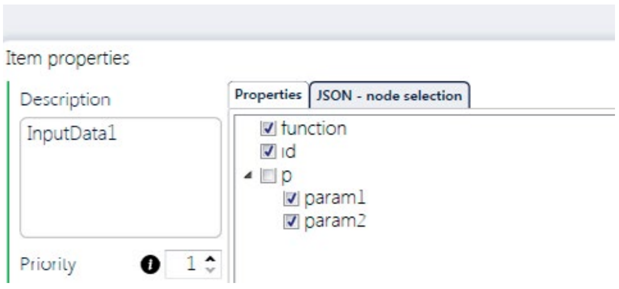
Because INPUT DATA is an input operator, and multiple input operators can exist inside the project, you can declare the order of initial operators execution by editing PRIORITY parameter GRAVITY application example



Parameter name: InputData1

Path to template json file

The first step is to select an example JSON format file GRAVITY application example. The chosen file will be a input data structure template.



Basing on the template file, GRAVITY will create a list of potential JSON format elements. By selecting, you declare a precise structure of data transferred to GRAVITY by a system, that executes GRAVITY application as WEB SERVICE function.

Below JSON format of data transferred while calling WEB SERVICE by an external software

- call_alias – WEB SERVICE name (see: CALLS option, execution_type = Webservice)
- process_id – non-obligatory session variable, given optionally by executing program, used during pagination (data packing)
- get_data – list of OUTPUT DATA indicators for data download (one project can have multiple OUTPUT DATA operators)
    - » records_count – number of records inside the package (page), no parameter means resignation form packing.
    - » description - OUTPUT DATA indicator name
    - » offset – from which record should the system start paginating.
- parameters
    - » global input parameters: if you define global GRAVITY application parameters earlier, than pointing out their unique name inside JSON file together with the value
    - » input parameters for specified INPUT DATA indicator (identification of INPUT DATA operator by name): whole data stream transferred in this method can be found on output bus of INPUT DATA indicator selected by name.

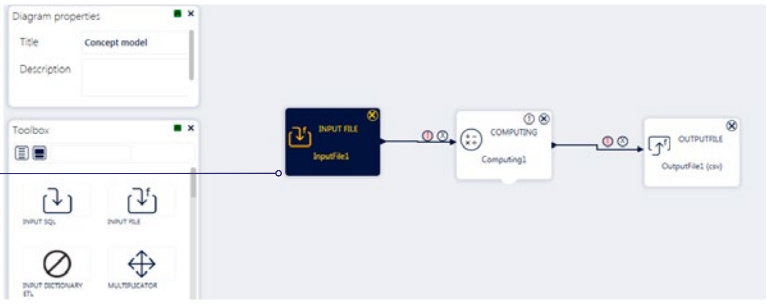
Example of JSON format (data transferred during WEB SERVICE call)

```
{
          „call_alias”: „test_index”,
          „process_id”: „327”,
          „get_data”: [
                    {
                    «description»: «OutputData1»,
                    „records_count”: „10”,
                    „offset”: „10”
          }
],
„parameters”: {
          „id”: „4”,
          „id2”: „1000”,
          «input_data_1»: [
                    {
                    „function”: „z22”,
          „id”: „84576”
          },
          {
                    „function”: „p23”,
          „id”: „31”
                              }
                    ]
          }
}
```

8.1.2 INPUT FILE

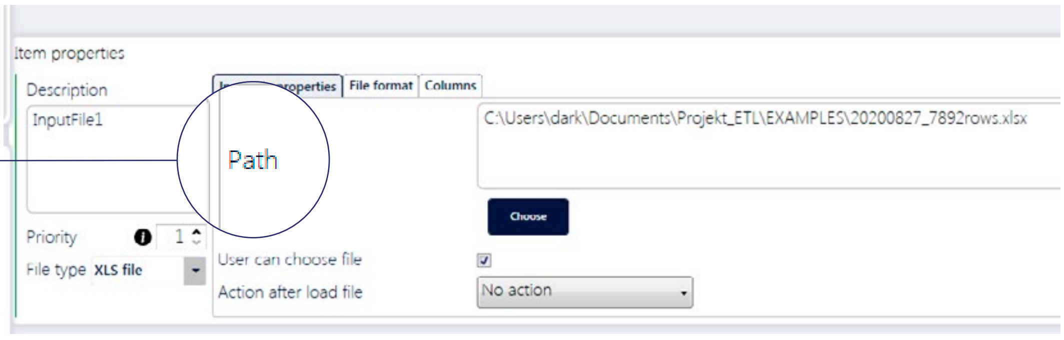
INPUT FILE operator enables data download from a file. On input of the operator a file with data is selected, on the output – there is a bus with data stream.

INPUT FILE PARAMETERS SETTING



GRAVITY project example using INPUT FILE operator

On INPUT FILE PROPERTIES, you will set up necessary parameters that allow to identify a file downloaded by the operator.

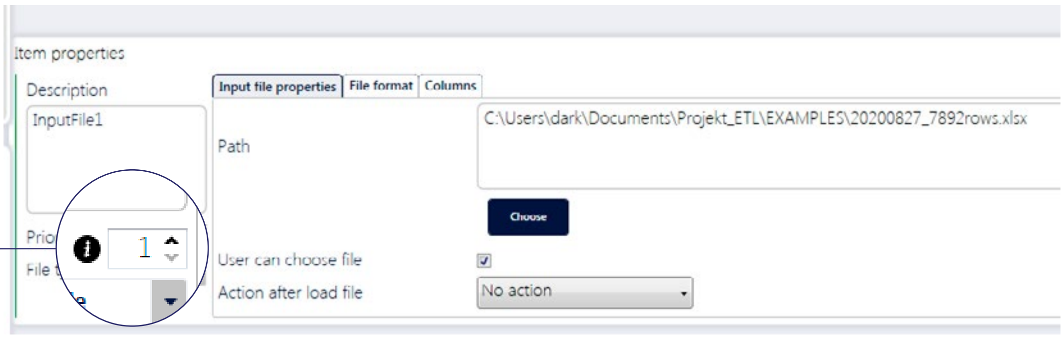


Item properties
Description: InputFile1
Priority: 1
File type: XLS file
Path: C:\Users\dark\Documents\Projekt_ETL\EXAMPLES\20200827_7892rows.xlsx
User can choose file: ✓
Action after load file: No action

On picture above we marked PATH parameter: this is were you select file location.

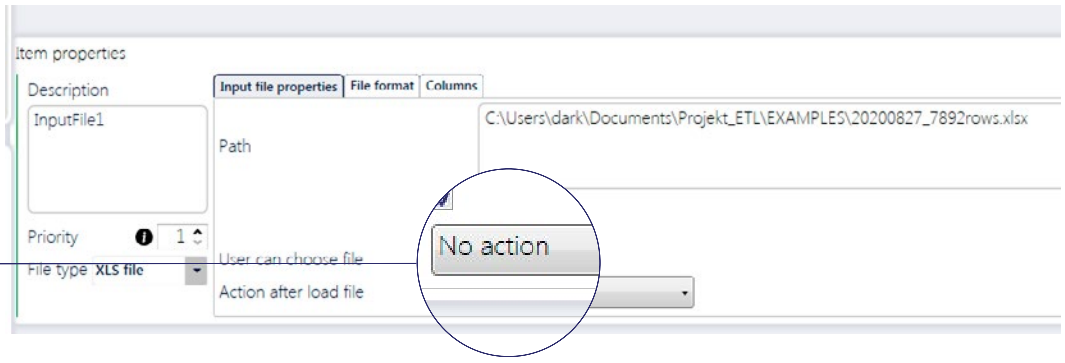
File selection can be done in a more sophisticated way. FILE RECOGNITION option allows the following procedure variants:

- ONLY PATTERN ….. GRAVITY downloads a file clearly chosen in PATH parameter.
- USER CAN CHOOSE…. The user selects the file himself during action (the action only has sense, when we call the project manually)
- OLDEST IN CHOOSEN PATH…..GRAVITY automatically selects the oldest file from location (PATH)



Item properties
Description: InputFile1
Priority: 1
Path: C:\Users\dark\Documents\Projekt_ETL\EXAMPLES\20200827_7892rows.xlsx
User can choose file: ✓
Action after load file: No action

Because INPUT FILE is an initial operator, and multiple initial operators can exist inside the project, you can declare the order of initial operators execution by editing PRIORITY parameter GRAVITY application example


Item properties

Description: InputFile1

Priority: 1

File type: XLS file

Input file properties | File format | Columns

Path: C:\Users\dark\Documents\Projekt_ETL\EXAMPLES\20200827_7892rows.xlsx

User can choose file

Action after load file: No action

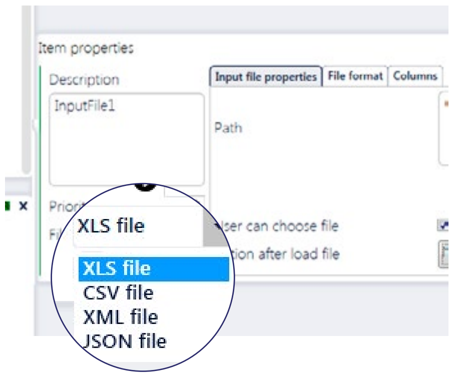
After downloading the file for processing, the processing itself may require different actions.

GRAVITY gives you the following opportunities

- NO ACTION … File is being download and remains in its location
- CHANGE FILE NAME…. File remains in its location, however its name is changed

Note: new filename can be designed using variables (variables range available in context help)

- MOVE FILE……… File is being moved to different location
- REMOVE FILE….. File is being deleted after download

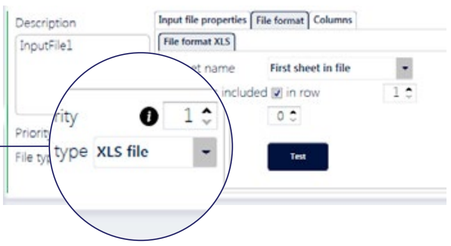


GRAVITY pozwala na pobieranie plików w następujących formatach:

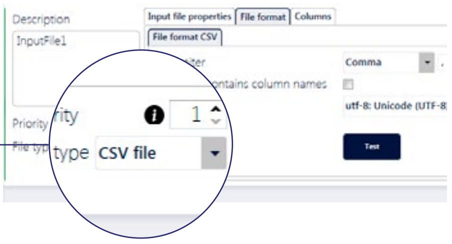
- XLS
- CSV
- XML
- JSON

INPUT FILE FORMAT CONFIGURATION

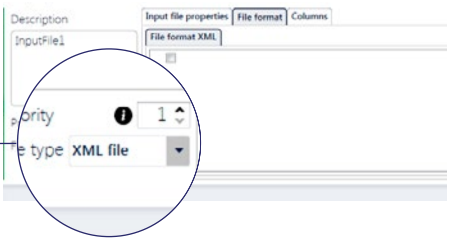
Depending on the format of downloaded file, user is obliged to set different parameters



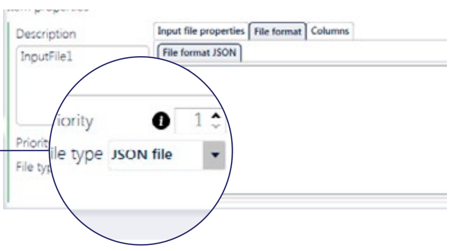
XLS file format configuration on the picture above.


Description: InputFile1
Input file properties | File format | Columns
File format CSV
Delimiter: Comma
Header contains column names: ☐
utf-8: Unicode (UTF-8)
Priority: 1
File type: CSV file

CSV file format configuration on the picture above.

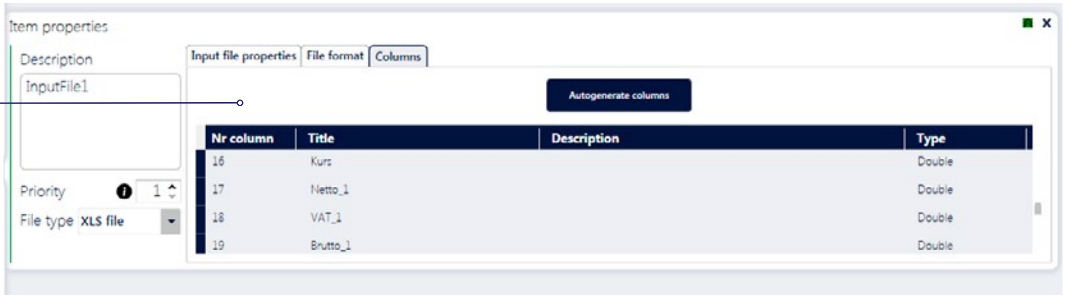

Description: InputFile1
Input file properties | File format | Columns
File format XML
Priority: 1
File type: XML file

XML file format configuration on the picture above. (GRAVITY enables the use of special editor to configure XML file)


Description: InputFile1
Input file properties | File format | Columns
File format JSON
Priority: 1
File type: JSON file

JSON file format configuration on the picture above. (GRAVITY enables the use of special editor to configure JSON file)

INPUT FILE COLUMN DECLARATION

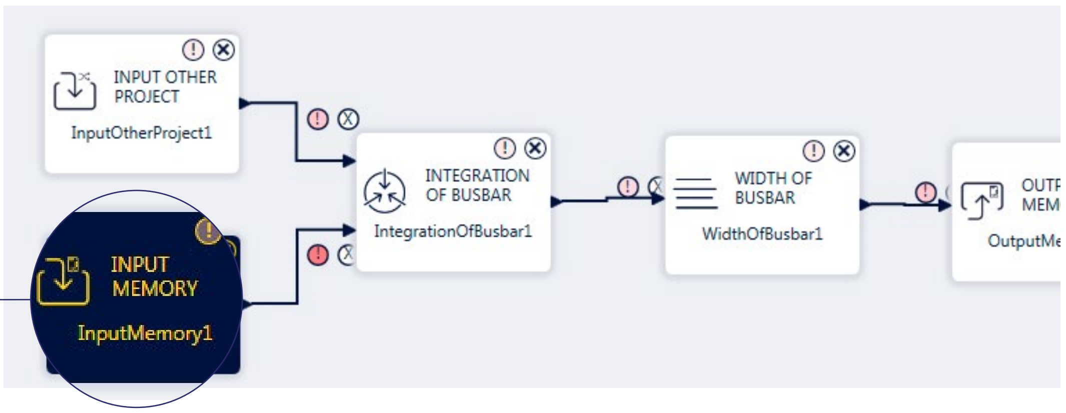

Item properties
Description: InputFile1
Priority: 1
File type: XLS file
Input file properties | File format | Columns

Nr column	Title	Description	Type
16	Kurs		Double
17	Netto_1		Double
18	VAT_1		Double
19	Brutto_1		Double

If the operator is already connected to the bus, you will be able do recognize the columns automatically.

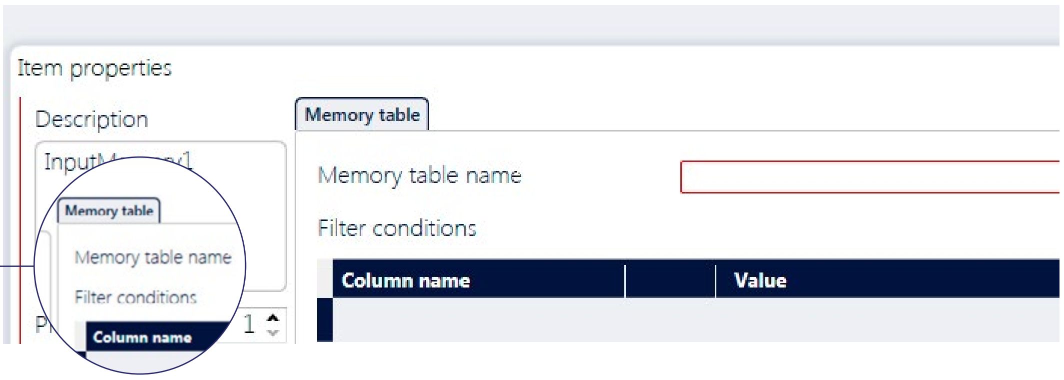
8.1.3 INPUT MEMORY

INPUT MEMORY operator introduces new data stream to an output bus. Data stream originates from the memory data file, built by OUTPUT MEMORY operator, inside the same or other GRAVITY application.



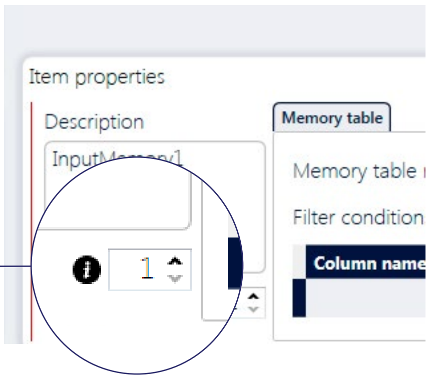
Example of GRAVITY project using OUTPUT MEMORY operator.

OPERATOR CONFIGURATION



You have to select a unique name of memory set.

Note that you don›t need to download all the data gathered inside the memory set, but download only a part of it, using filtering conditions.

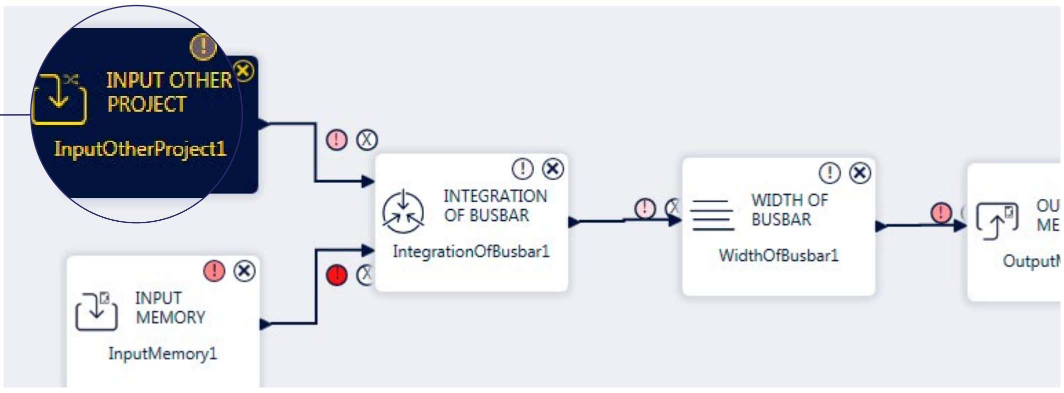


Because INPUT MEMORY is an initial operator, and multiple initial operators can exist inside the project, you can declare the order of initial operators execution by editing PRIORITY parameter GRAVITY application example

8.1.4 INPUT OTHER PROJECT

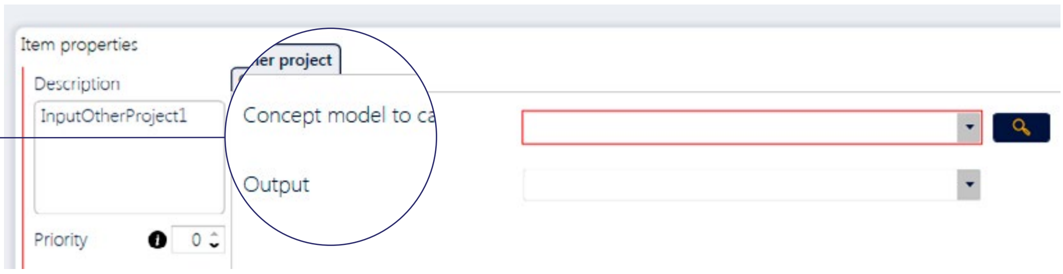
INPUT OTHER PROJECT operator allows inputing to the data bus, data stream which is the result of processing of different GRAVITY project.

OPERATOR CONFIGURATION



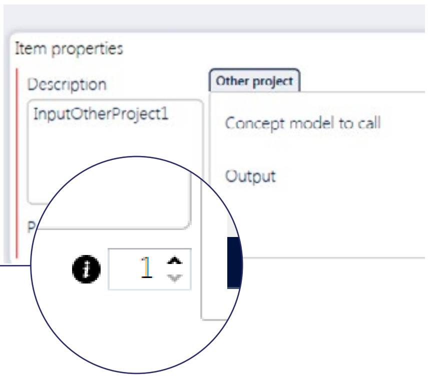
Example of GRAVITY project using INPUT OTHER PROJECT operator.

In INPUT OTHER PROJECT operator you need to declare othe GRAVITY project.



You need to select OUTPUT type operator, which is the source of external project data.

This project will be automatically processed during the implementation of actual project, in order to obtain initial data stream.

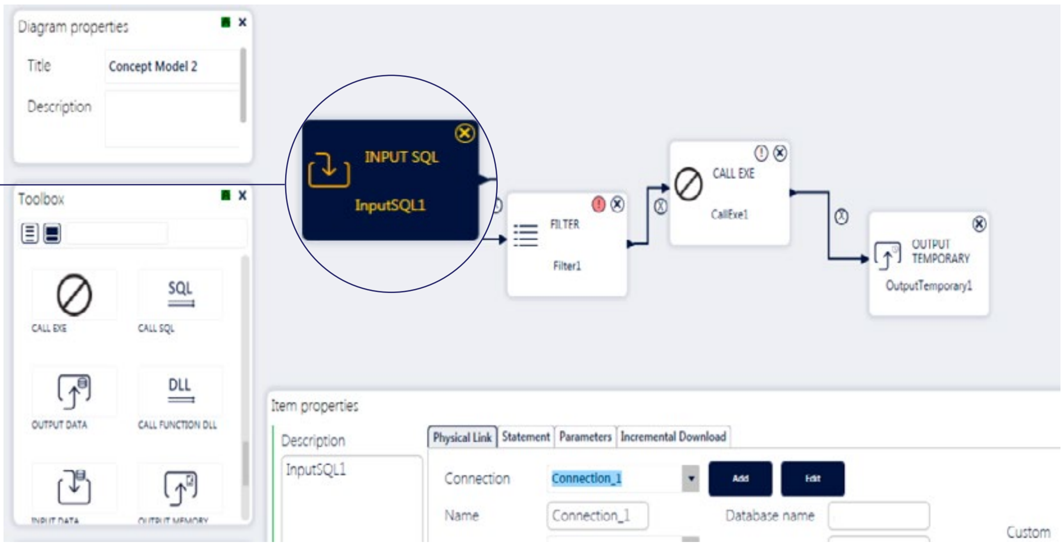


You can declare the order of initial operators execution by editing PRIORITY parameter

Because INPUT OTHER PROJECT is an initial operator, and multiple initial operators can exist inside the project

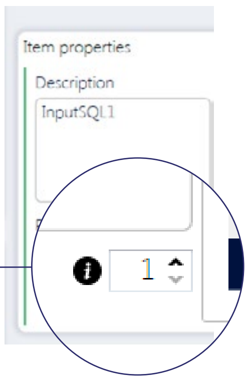
8.1.5 INPUT SQL

INPUIT SQL allows downloading the data from external data source (SQL base)



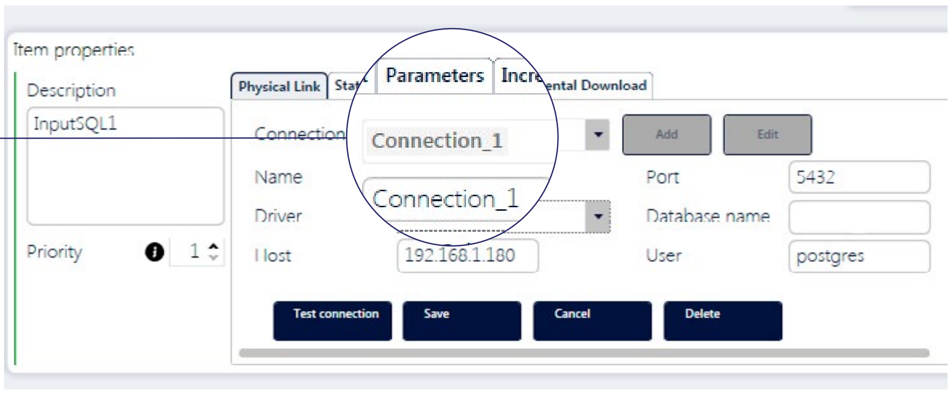
Projekt GRAVITY z użyciem operatora INPUT SQL

PARAMETER SETTING: PRIORITY

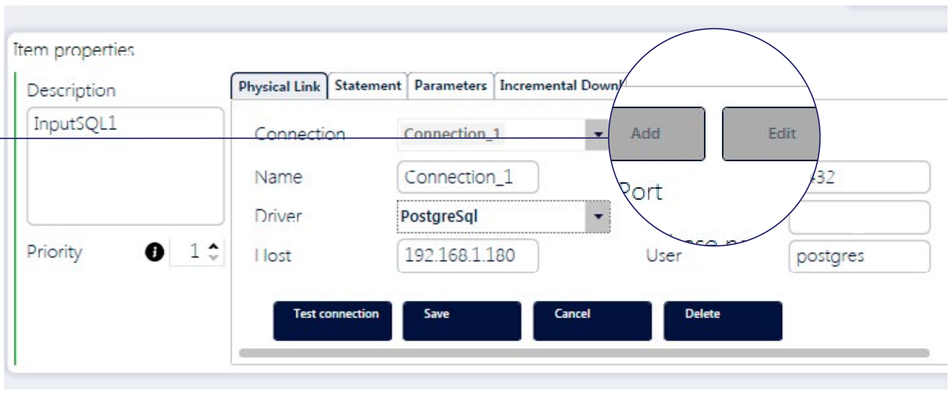


If GRAVITY project includes multiple input operators, you can set the order of processing using PRIORITY parameter.

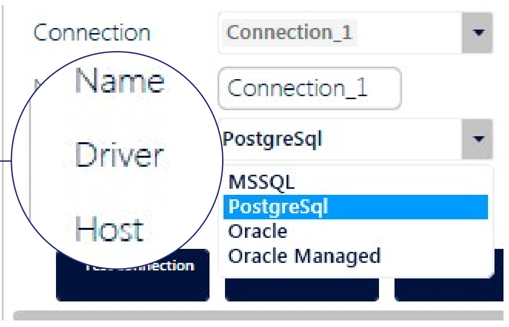
PARAMETER GROUP SETTING: PHYSICAL LINK



If anytime earlier a connection with SQL base was configured – they can be chosen by pointing the selection on the list.

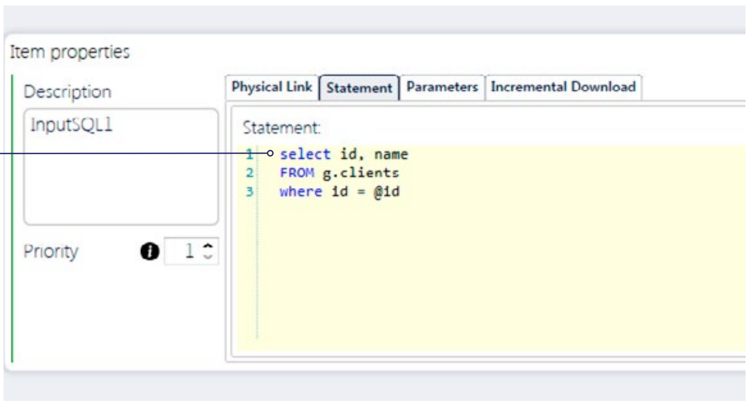


In case you want to configure new connection or edit the old one, you have to use options presented on the picture above (ADD or EDIT).



While editing the connection parameters, you can choose drivers from the list – see picture above

PARAMETER SETTING: STATEMENT



Item properties

Description: InputSQL1

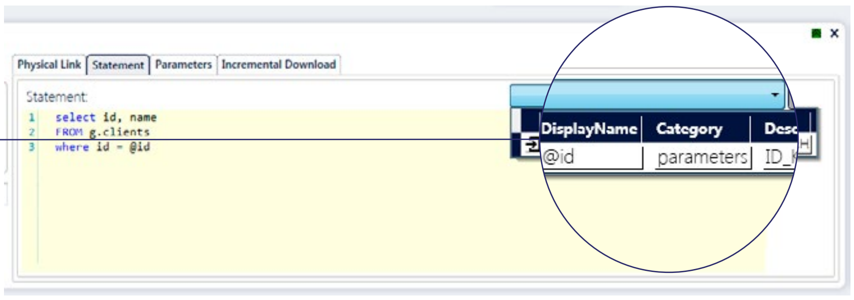
Priority: 1

Physical Link | Statement | Parameters | Incremental Download

Statement:

```
select id, name
FROM g.clients
where id = @id
```

The picture above presents the edit of SELECT statement.



Physical Link | Statement | Parameters | Incremental Download

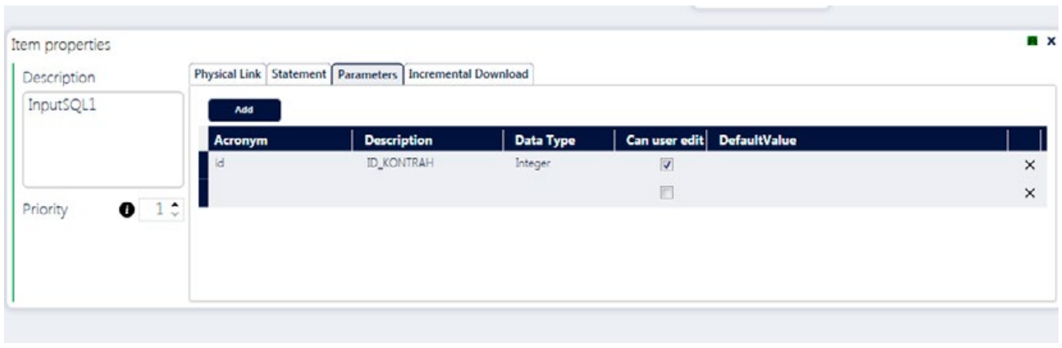
Statement:

```
select id, name
FROM g.clients
where id = @id
```

DisplayName	Category	Desc
@id	parameters	ID_

In SELECT statement you can use parameters defined in PARAMETERS tab. You can make the selection of the parameter by pointing it out on the list – see picture above

PARAMETER GROUP SETTING: PARAMETERS



Item properties

Description: InputSQL1

Priority: 1

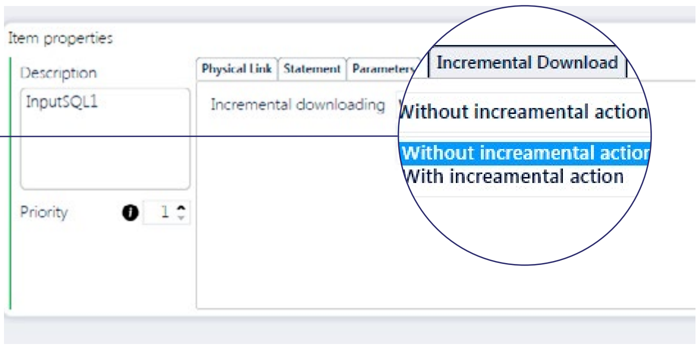
Physical Link | Statement | Parameters | Incremental Download

Add

Acronym	Description	Data Type	Can user edit	DefaultValue
id	ID_KONTRAH	Integer	✓	
			☐	

You can declare any number of parameters, which can (but don't have to) be edited while being processed by the user, and are available for your SQL expression for each project operator.

PARAMETER GROUP SETTING: INCREMENTAL DOWNLOAD



Item properties

Description: InputSQL1

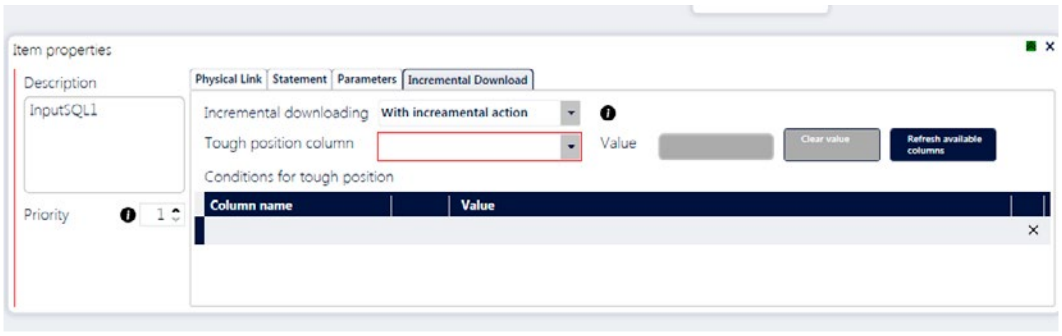
Priority: 1

Physical Link | Statement | Parameters | Incremental Download

Incremental downloading: Without increamental action

Without increamental action
With increamental action

Data download can take place in an incremental way: all you need to do is to select INCREMENTAL DOWNLOADING = WITH INCREMENTAL ACTION on the list


Item properties

Description: InputSQL1

Priority: 1

Incremental downloading: With increamental action

Tough position column:

Value:

Conditions for tough position

Column name	Value

If you decide to use incremental method, you need to select key column (tough position column). Additionally, you can select conditions to be fulfilled, so that the system takes the value of the column as the last one, collected after finishing processing. The recently received value can be cleaned using CLEAR VALUE option.

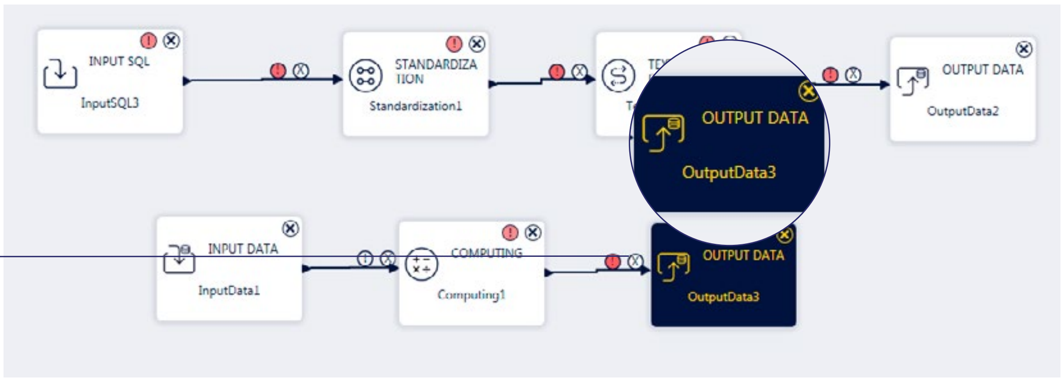
Declaration of incremental method is not visible in STATEMENT option as an SQL code. SQL code, responsible for the mechanism stated above arises automatically (is being generated by GRAVITY) while downloading the data, narrowing the downloaded data stream.

8.2 OUTPUT OPERATORS

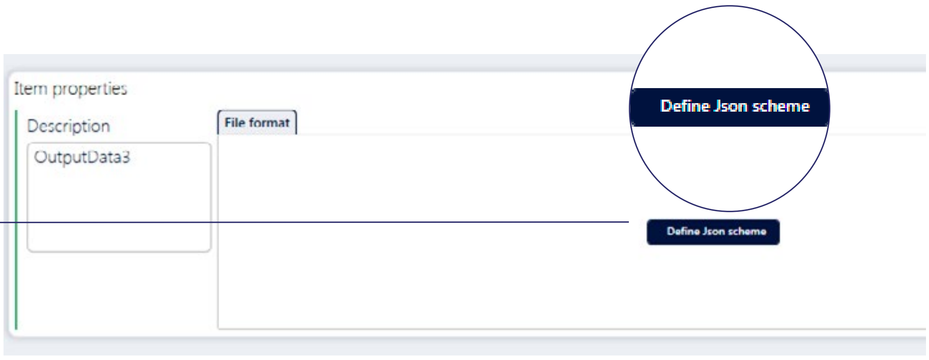
8.2.1 OUTPUT DATA

In context of GRAVITY application execution as WEB SERVICE function via an external software, you have at your disposal two operators communicating with the interior of the project: INPUT DATA and OUTPUT DATA

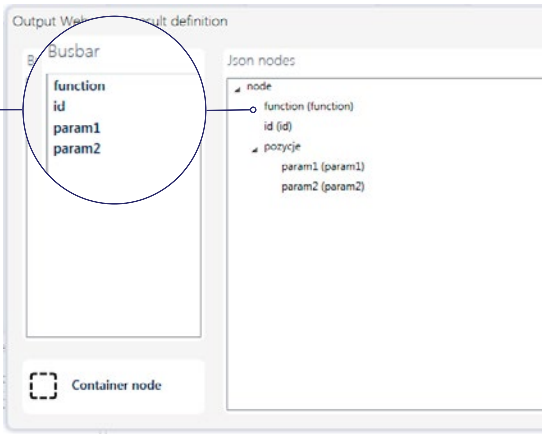
OUTPUT DATA is an operator that transfers the data stream as a result of GRAVITY action processing, to a program executing the project, as WEB SERVICE function. Data stream is in JSON format.



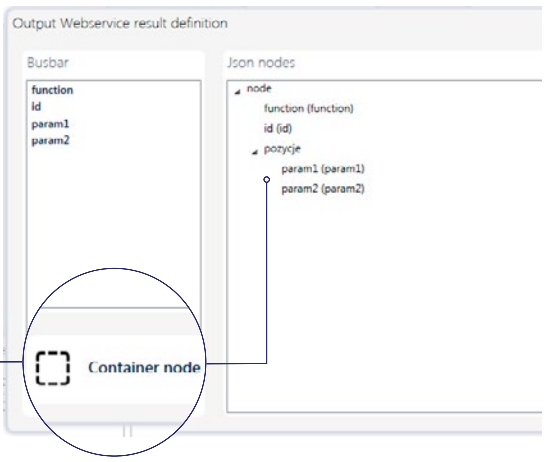
Example of GRAVITY project using OUTPUT DATA operator.



In the first step you need to select DEFINE JSON option GRAVITY application example, which will help you configure the format of data stream in an automatic and clear way.



You build the format by dragging and dropping the columns of input signal onto the area of JSON format GRAVITY application example



You can also create depth levels by dragging and dropping CONTAINER NODE area onto JSON format area GRAVITY application example

Below JSON format of data transferred as a result, while calling WEB SERVICE by an external program.

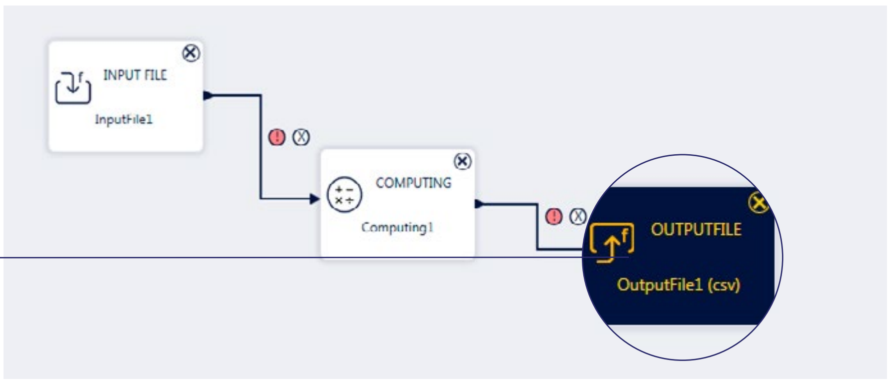
- error_code – error number (0 – no error)
- exeptions – errors table (only for the use of EXCEPTION class C#);
- data – Data table from all OUTPUT DATA operators, declared during WEB SERVICE function execution. (see: INPUT DATA operator)
  - » [operator name]
  - » (data dable from the selected OUTPUT DATA operator)

Przykładowy format JSON (strumień danych jako rezultat):

```
“ErrorCode”: 0,
“Exceptions”: [],
“Data”: [
{“OutputDataIndeksy”: {
“data”: [
{“id_index”: “4”,
“index”: “Product4”,
“name”: “Name_ Product 4”,
“jm”: “kg”
},
{
“id_index”: “6”,
“index”: “ Product 6”,
“name”: “name_ Product6 “,
“jm”: “kg”
}
]
}
},
{
“OutputData1”: {
“node”: [
{
“index”: Product5 “,
“name”: “ name_Product 5”
},
{
“index”: “ Product10 “,
“name”: “name_ Product10 “
},
{
“index”: “ Product 11”,
“name”: “ name_Product11 “}
]
}}
```

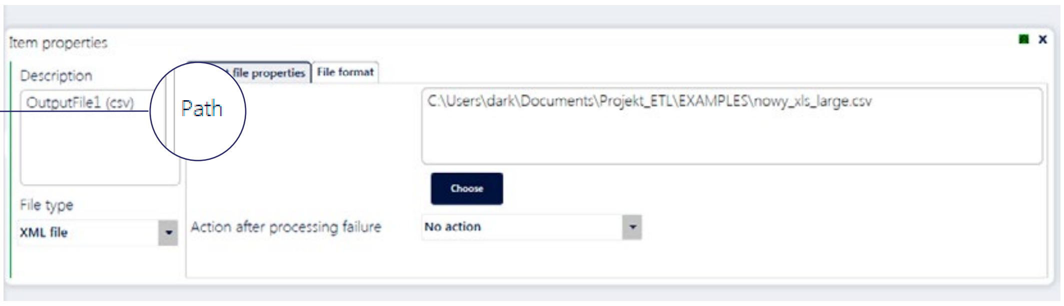
8.2.2 OUTPUT FILE

OUTPUT FILE operator allows creation of data file. On the input of an operator is a data stream, which as a result of operator action is being transformed into a file, in a format selected by the user.



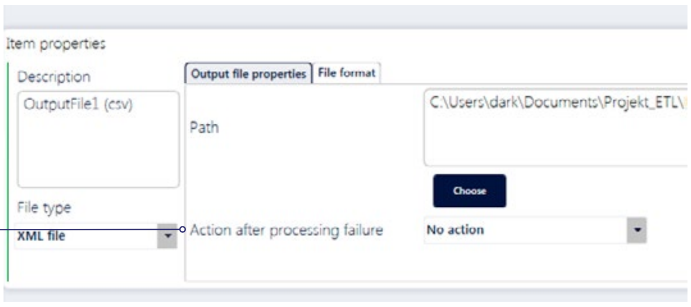
Example of GRAVITY project using OUTPUT FILE operator.

OUTPUT FILE PARAMETERS SETTING



C:\Users\dark\Documents\Projekt_ETL\EXAMPLES\nowy_xls_large.csv

In the OUTPUT FILE PROPERTIES tab you can set necessary parameters of file generated by the operator



C:\Users\dark\Documents\Projekt_ETL\

On the picture above we marked PATH parameter: here you choose location or file name.

As a standard, processing creates new file or overwrites it, if an identical file is found in the current location.

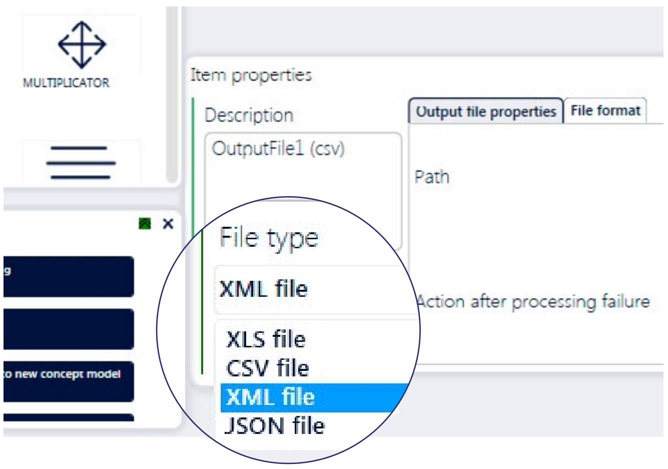
In the option shown above, you can design action variant in case of processing failure.

You can declare the following actions:

- NO ACTION – processing does not generate a new file
- CREATE OR OVERWRITE EMPTY FILE – Generated file will be empty
- REMOVE OLD FILE – If a file with the same name already exists in the current location – it will be deleted

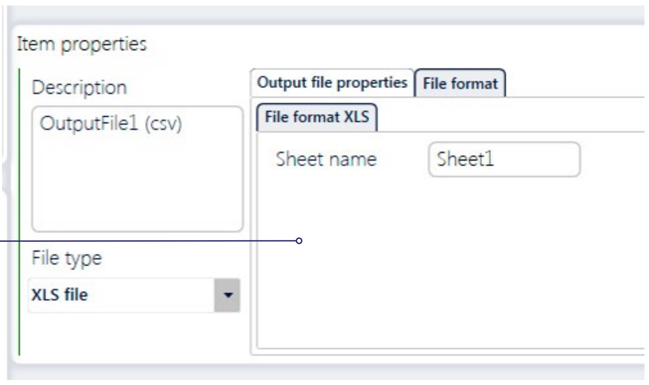
GRAVITY allows to generate files in the following formats :

- XLS
- CSV
- XML
- JSON



OUTPUT FILE FORMAT CONFIGURATION

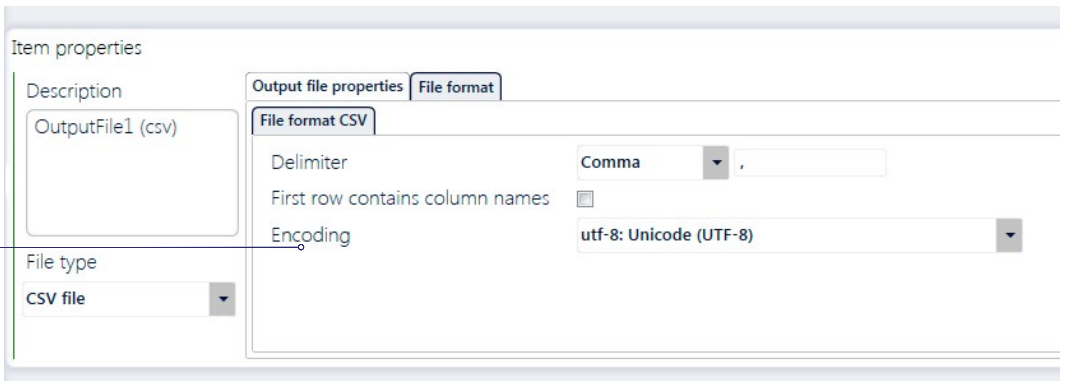
Depending on the format of output file, the user is obliged to set various parameters – see picture below.



Sheet name: Sheet1

File type: XLS file

XML file configuration on the picture above..



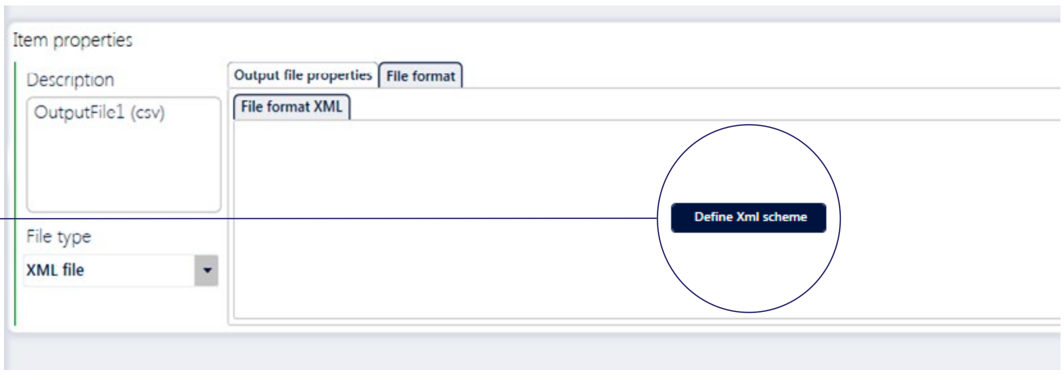
Delimiter: Comma ,

First row contains column names

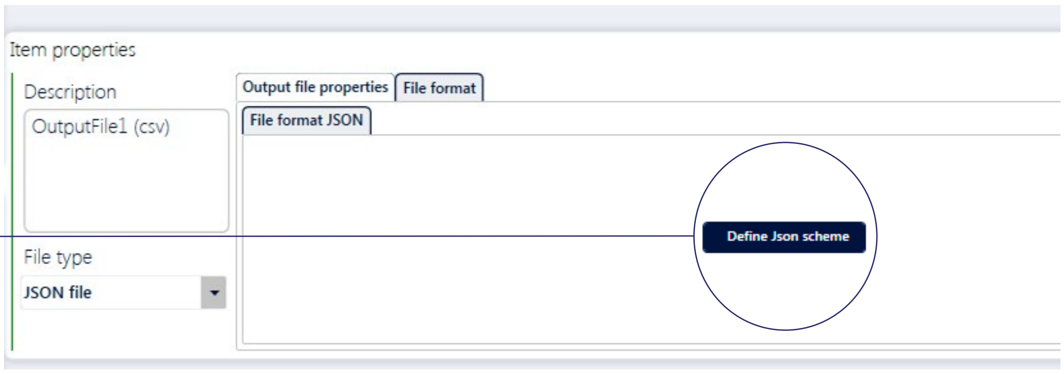
Encoding: utf-8: Unicode (UTF-8)

File type: CSV file

XML file configuration



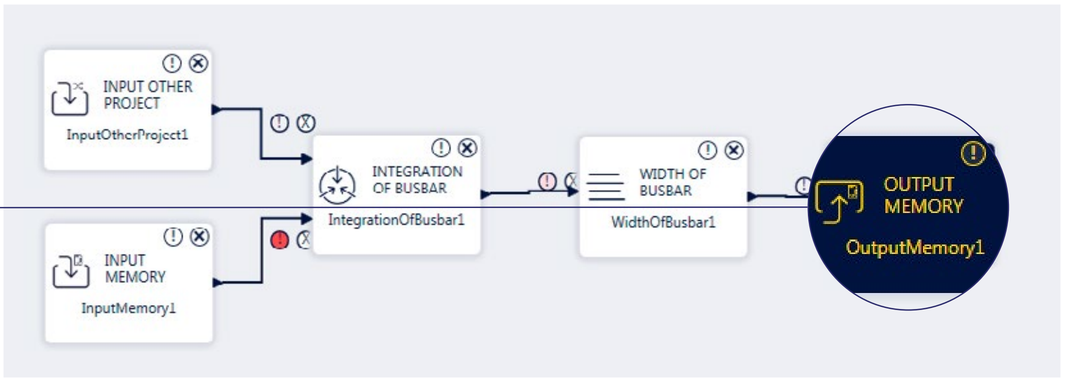
XML file configuration on the picture above. (GRAVITY gives you possibility to use special editor for XML file configuration)



JSON file configuration on the picture above. (GRAVITY gives you possibility to use special editor for JSON file configuration)

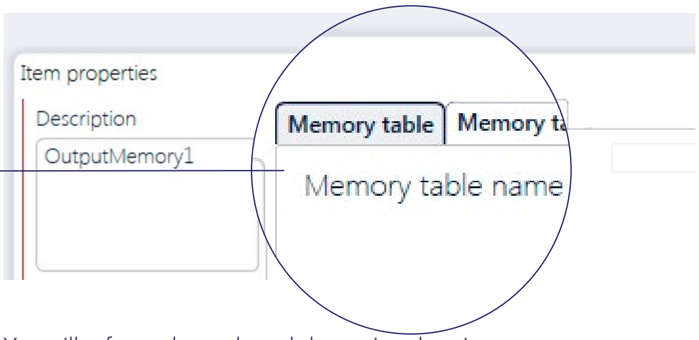
8.2.3 OUTPUT MEMORY

OUTPUT MEMORY operator gathers data, which come from the input bus data stream. The gathered data can be used in the same GRAVITY application or in other GRAVITY applications.



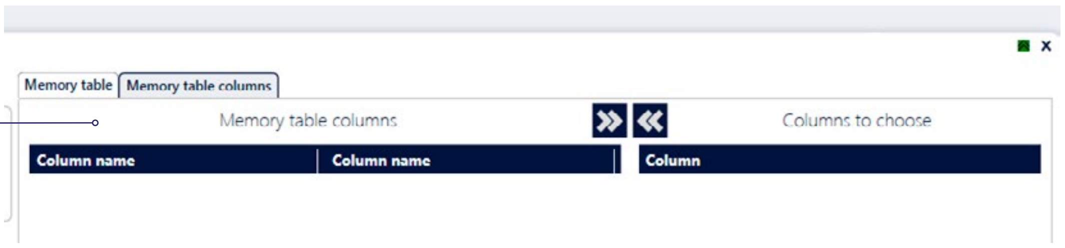
Example of GRAVITY project with OUTPUT MEMORY operator usage.

OPERATOR CONFIGURATION



You need to setup a unique name for the data gathered in the memory

You will refer to the gathered data using the given name.



The range of the gathered data depends on the configuration. You can select any number of columns from the bus to the operator (drag and drop action)

8.2.4 OUTPUT PDF

OUTPUT PDF operator enables formatting the data from output bus in such way, that the result of operator action will be one object in a PDF document form. PDF document can contain any data located on the input bus. PDF document form is being configured while creating GRAVITY application.

The operator enables creating in GRAVITY complete reporting system, printouts, statements in a format adjusted to the needs of the user. Reports are available for each external software as WEB SERVICE functions.

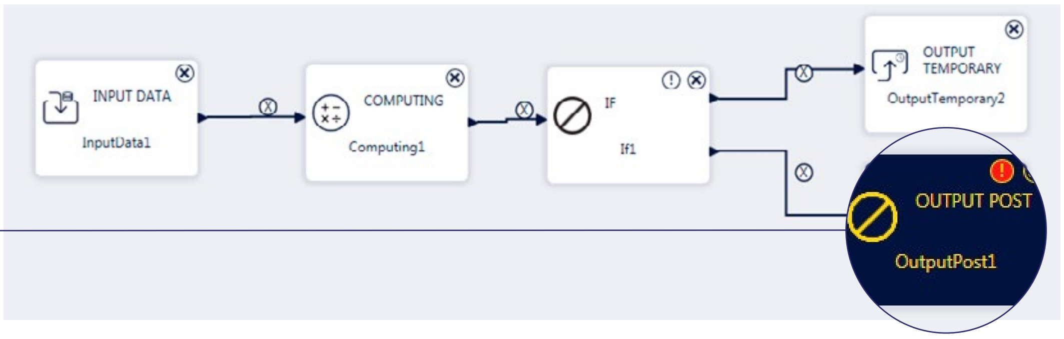
Detailed description of report construction (so OUTPUT PDF indicator configuration) can be found in an independent GRAVITY REPORT manual.

8.2.5 OUTPUT POST

OUTPUT POST gives opportunity to transform the signal on input bus, to an information sent by e-mail.

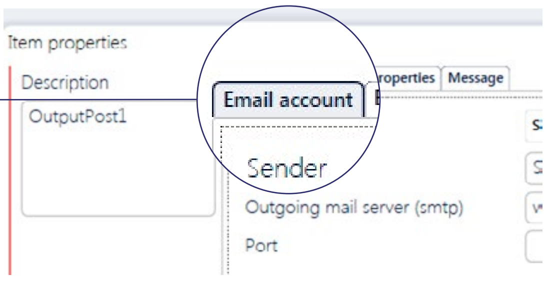
Recipients of the mail can be set basing on the information contained in the input bus or permanently defined. Input data stream can be gathered in a file and sent by e-mail as an attachment.

Mail can be edited for each input data stream record, or grouped for declared operator key designer.



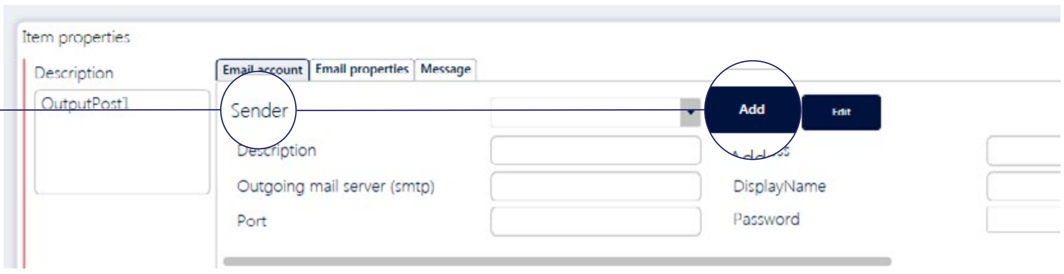
Example of GRAVITY project using OUTPUT POST operator.

OPERATOR COFIGURATION: EMAIL ACCOUNT

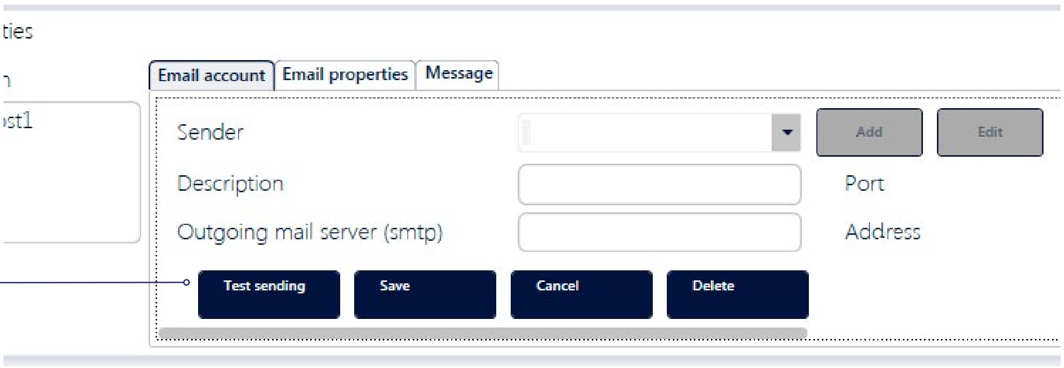


In the EMAIL ACCOUNT tab GRAVITY application example e-mail account necessary to send mail correspondence

You can select the mailbox from the list (if you configured it earlier in GRAVITY mailboxes) or declare a new account GRAVITY application example.



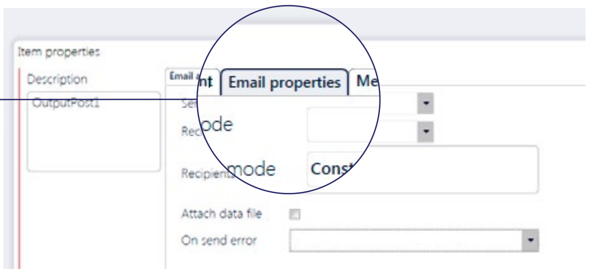
On the picture above you can see which parameters have to be filled when configuring the mailbox. Note, that you have TEST SENDING option available, which will allow you to evaluate if data you inputed is correct.



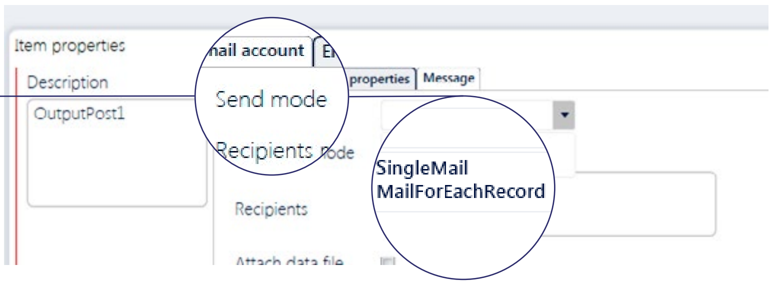
Note, that you have TEST SENDING option available, which will allow you to evaluate if data you inputed is correct.

OPERATOR CONFIGURATION: EMAIL PROPERTIES

In the EMAIL PROPERTIES tab GRAVITY application example you will declare the recipients of the mail, attached files and the system behavior in case of an error.



In the first step – define the way you want to send mail correspondence.

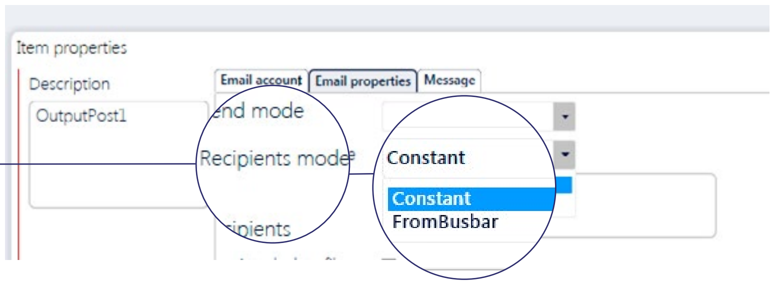


SEND MODE option:

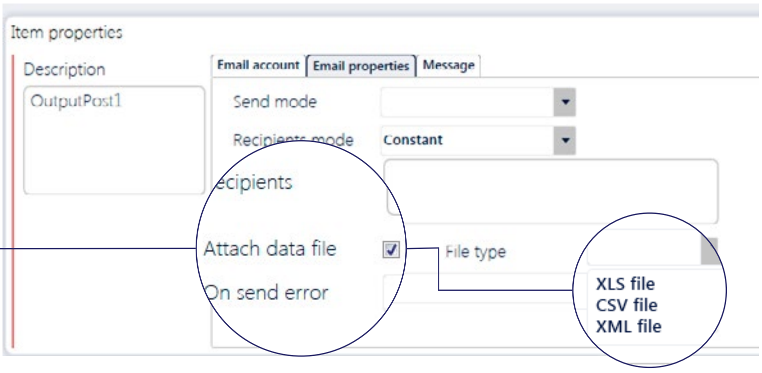
- one mail for whole stream (SEND MODE = SingleMail)
- for each data stream record you want to send an independent e-mail (SEND MODE= MailForEachRecord)

Those are two completely different strategies of understanding OUTPUT POST operator process, both ways can offer you priceless services for independent processes.

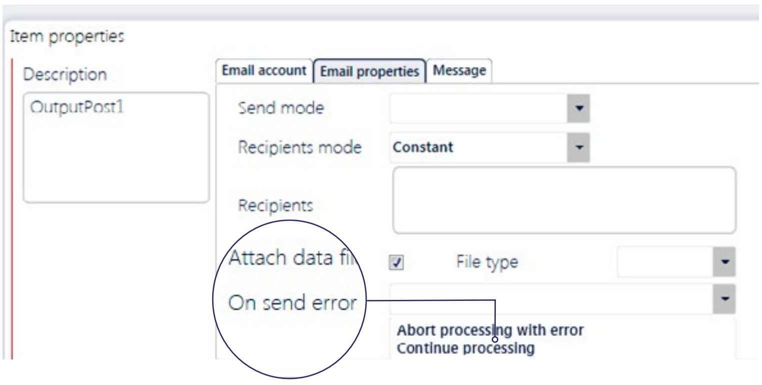
When using SEND MODE= MailForEachRecord you can for example automatically generate, within the designated by you time intervals, without involvement of your employees, balances for all your contractors, while using SEND MODE =SingleMail you can prepare a report based on a diagnosed situation, detected automatically (internal alert)



You can send mail to selected constant recipients (RECIPIENTS MODE = CONSTANS; option – see picture above) or to an address which can be found on the input data bus (RECIPIENTS MODE = FromBusbar).



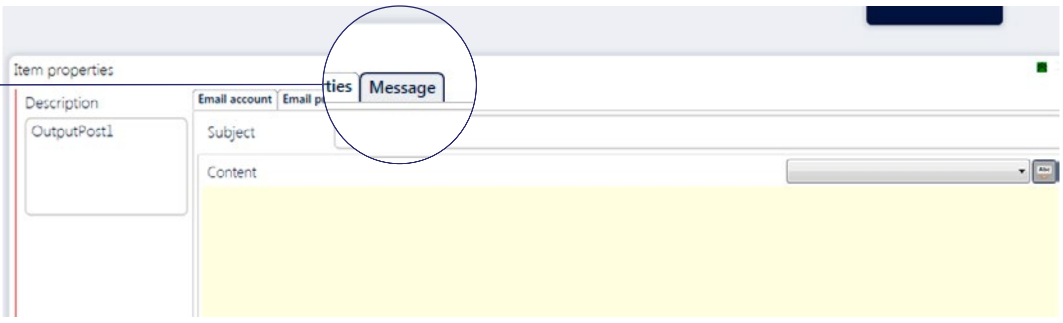
You can add a processed data stream to your e-mail, in a format selected by you GRAVITY application example. In that case you need to mark ATTACH DATA FILE option and select one of file formats accepted by GRAVITY: xls, xml or csv.



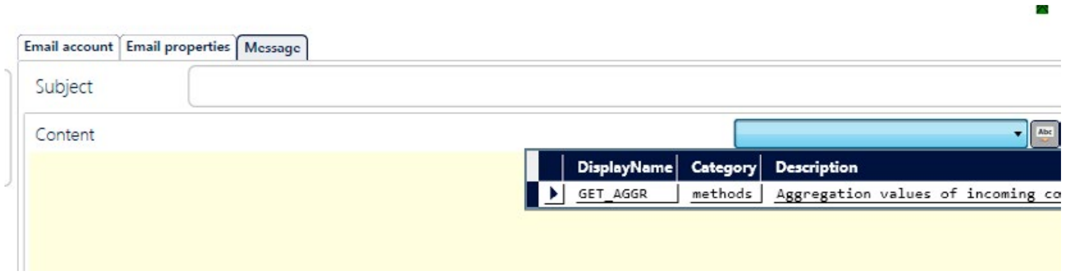
While data processing in OUTPUT POST operator, GRAVITY can face a dilemma in which way to process data in case of an error. You can design it, setting ON SEND ERROR option onto one to two possible variants:

- ABORT PROCESSING WITH ERROR
  process is stopped and all the operations performed so far are withdrawn – such process is considered as non-existent)
- CONTINUE PROCESSING

OPERATOR CONFIGURATION: MESSAGE



In the MESSAGE tab GRAVITY application example, you will define the title and text of your e-mail correspondence.

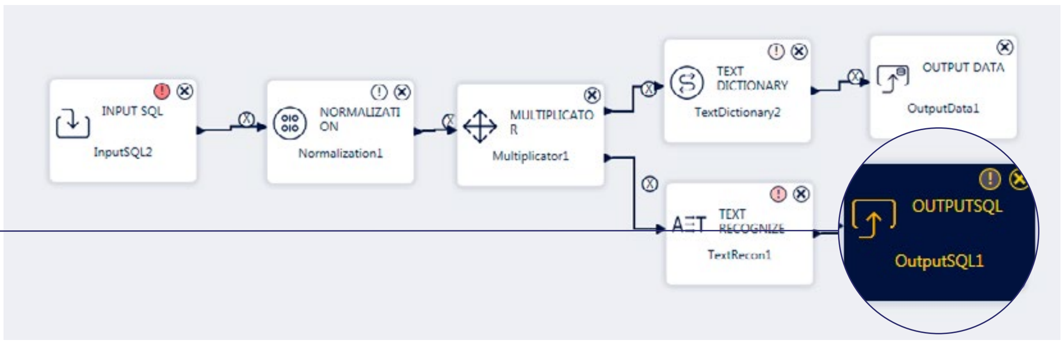


DisplayName	Category	Description
GET_AGGR	methods	Aggregation values of incoming co

Within the content you can use GET_AGGR parameter (adding up selected bit on input bus), thanks to which you can refer to the sums inside the input data stream.

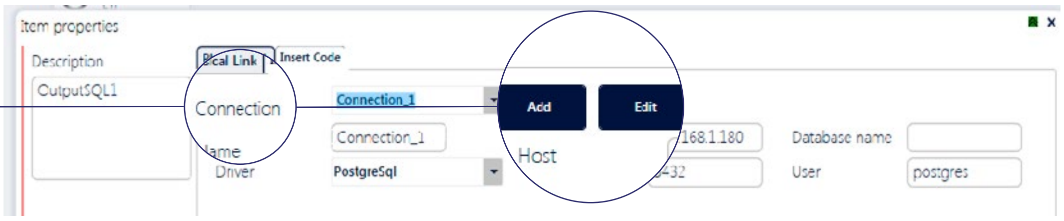
8.2.6 OUTPUT SQL

OUTPUT SQL operator is responsible for introducing the input bus data stream to a selected SQL base.



Example of GRAVITY project with OUTPUT SQL operator usage.

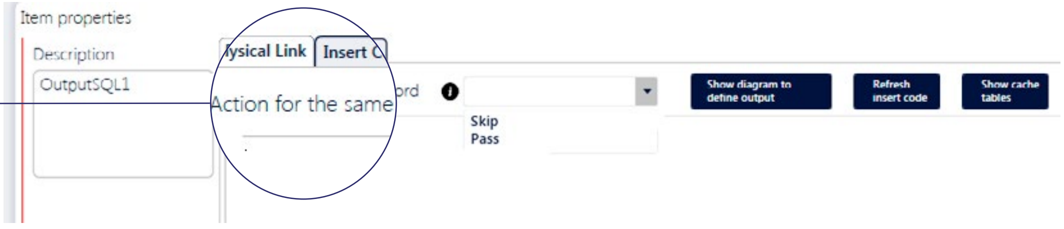
BASE CONNECTION CONFIGURATION



In the first step you need to select a connection to a chosen base. Your choice is selected from the list – if you configured such connections earlier – or define a new one (ADD or EDIT options – see picture above).

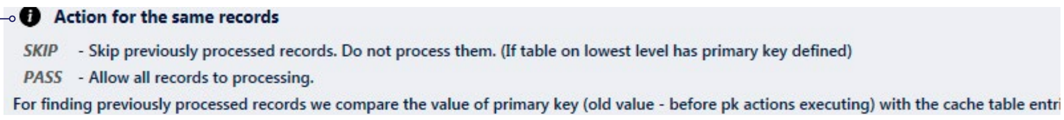
WARNING! You are selecting a target base, which you intend to introduce the data stream to..

ACTION DECLARATION IN CASE OF DUPLICATE DETECTION



Decide, what the system shoud do in case of detecting duplicated data in the target base.

You have two options available in case of detection of duplicated records in the target base while adding the data.



Action for the same records

SKIP - Skip previously processed records. Do not process them. (If table on lowest level has primary key defined)

PASS - Allow all records to processing.

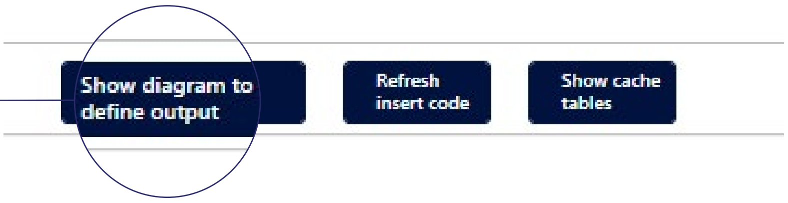
For finding previously processed records we compare the value of primary key (old value - before pk actions executing) with the cache table entri

If you select PASS option, then process behavior for duplicates depends on action pointed in SHOW DIAGRAM TO DEFINE OUTPUT option. In SQL ACTIONS option you can define:

- DELETE OLD RECORD (Detected record will be deleted, and a new record will be added) or
- UPDATE OLD RECORD (Detected record will be overwritten with a new record)

DATA STREAM ADDING CODE DECLARATION

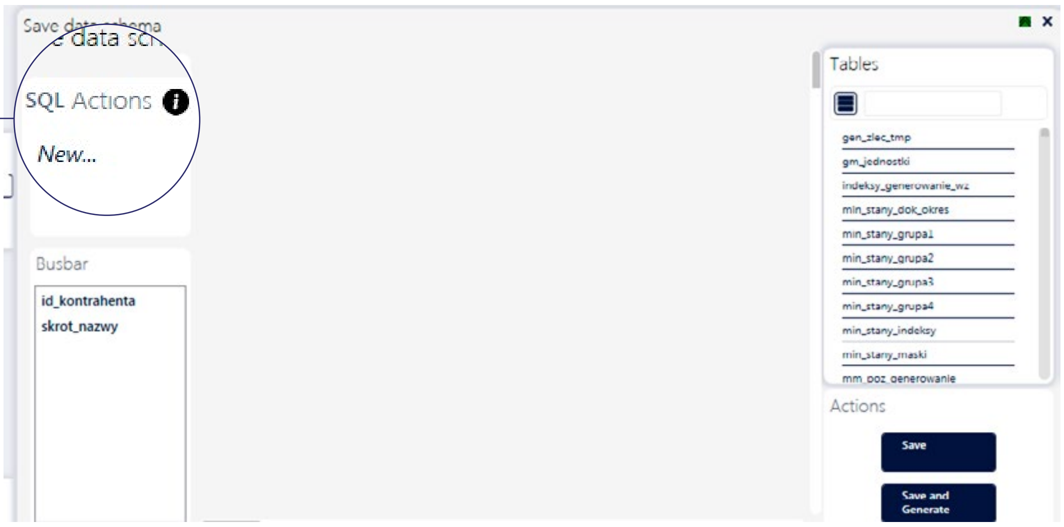
Data stream adding code is generated automatically, basing on a diagram configured by you. (see oprion SHOW DIAGRAM TO DEFINE OUTPUT.



Automatically generated code has XML format and can be also modified by you manually. If you want to return to a template code, created in the diagram, it is enough to start REFRESH INSERT CODE option)

DATA STREAM ADDING CODE DECLARATION: ADDITIONAL ACTIONS

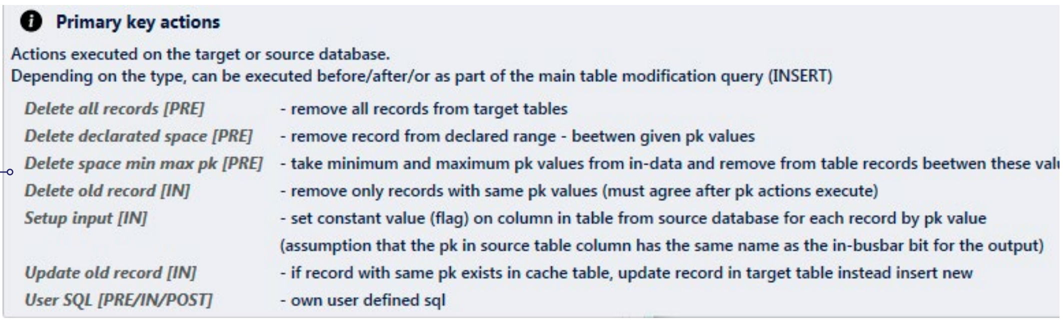
After selecting SHOW DIAGRAM TO DEFINE OUTPUT option, you will see a window just like on the picture below.



In the first step you need to think, which actions you want to execute, before the process of adding data stream to the base. The illustration above shows selected NEW option, which will allow you adding actions:

- PRE type, which are actions before adding data stream to the base
- POST type, which are actions after adding data stream to the base
- IN type, actions while adding, for example system behavior in case of detecting an identical record (according to PRIMARY KEY) in the target database.

If you need detailed help in terms of specific actions functions, run HELP option – you will receive window with explanations



Primary key actions

Actions executed on the target or source database.
Depending on the type, can be executed before/after/or as part of the main table modification query (INSERT)

Delete all records [PRE]	- remove all records from target tables
Delete declarated space [PRE]	- remove record from declared range - beetwen given pk values
Delete space min max pk [PRE]	- take minimum and maximum pk values from in-data and remove from table records beetwen these val
Delete old record [IN]	- remove only records with same pk values (must agree after pk actions execute)
Setup input [IN]	- set constant value (flag) on column in table from source database for each record by pk value (assumption that the pk in source table column has the same name as the in-busbar bit for the output)
Update old record [IN]	- if record with same pk exists in cache table, update record in target table instead insert new
User SQL [PRE/IN/POST]	- own user defined sql

DELETE ALL RECORDS

Before adding the data stream (PRE type action), target table will be cleaned of all the records.

DELETE ALL RECORDS FOR SOURCE

Before adding the data stream (PRE type action), target table will be cleaned of all the records, but only in terms of those records, which were standardized on the base of the idicated source. The source is identified basing on the name (constans) or selected bit value (columns) of input bus.

DELETE DECLARDED SPACE

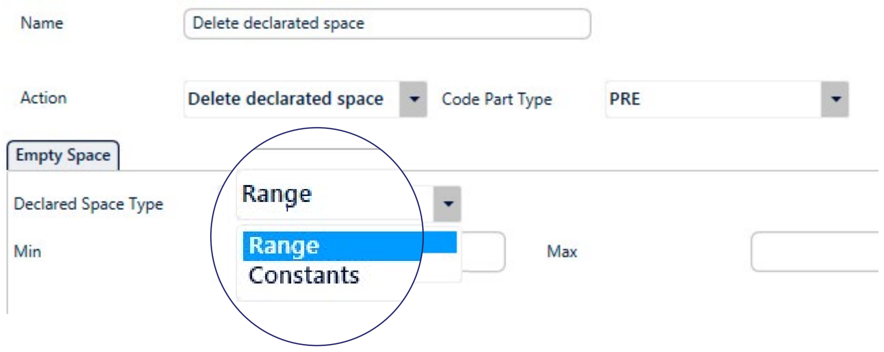
Before adding the data stream (PRE type action), target table will be cleaned of all the records in the declared space.

You have two methods to select which records range should be deleted (see picture below)

- RANGE
- CONSTANS

RANGE setting enables selecting the space in terms of PRIMARY KEY column value. The records within this space will be deleted.

You will use the option when in one table you collect records from multiple sources, and having only one column at your disposal, you solved the problem of unique ID by creating numerical spaces for different sources (for example: source 1: space from 1 -> 1 000 000, source 2: space 1 000 001 -> 2 000 000 etc.)



Name	Delete declarated space		
Action	Delete declarated space	Code Part Type	PRE
Declared Space Type	Range		
Min		Max	

DELETE SPACE MIN MAX PK

Before adding the data stream (PRE type action), target table will be cleaned of all the records in the declared space. Space for deletion will be determined in the area between the minimal and the maximum PRIMARY KEY value (declaration of PK key – refer to description TARGET TABLE AND ADDING RULES INDICATION)

Warning! If the columns belonging to PK has been standardized, (so had the value altered), GRAVITY will automatically detect it and identifies the proper record without any mistakes

DELETE OLD RECORD

Specific action (twin to UPDATE OLD RECORD action), executed for each data stream record (in this case – it is IN type action). If you set up an input filter ACTION FOR THE SAME RECORD = PASS (see description above), now you can select the method which GRAVITY process takes care of potential duplicate. If you select DELETE OLD RECORD action – current record will be deleted, so that the new record (potential duplicate) could be added without any obstacles.

Warning: if a unique data stream ID has been standardized (so it had the value altered), GRAVITY will automatically detect it and identifies the proper record without any mistkaes.

If you selected ACTION FOR THE SAME RECORD = SKIP filter, then DELETE OLD RECORD action will not be executed.

UPDATE OLD RECORD

Specific action (twin to DELETE OLD RECORD action), executed for each data stream record (in this case – it is IN type action). If you set up an input filter ACTION FOR THE SAME RECORD = PASS (see description above), now you can select the method which GRAVITY process takes care of potential duplicate. If you select UPDATE OLD RECORD action – current record will be overwritten by a new record (potential duplicate).

Warning: if a unique data stream ID has been standardized (so it had the value altered), GRAVITY will automatically detect it and identifies the proper record without any mistkaes.

If you selected ACTION FOR THE SAME RECORD = SKIP filter, then UPDATE OLD RECORD action will not be executed.

SETUP INPUT

IN type action (so performed multiple times, after each processing of following record in OUTPUT SQL operator), that gives the opportunity to edit a record downloaded to the bus, by any of INPUT SQL type operators. You will use this type of action if you want to, for example, setup download flags from the source.

Configuration range:

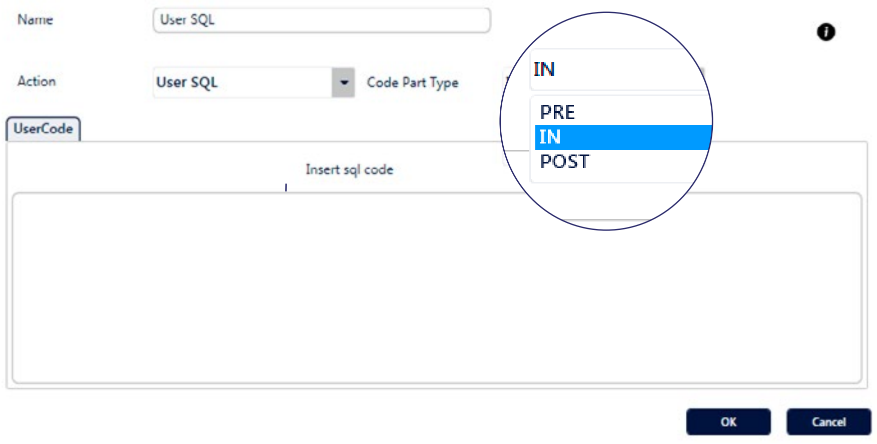
- INPUT……. Select INPUT type operator, where you download data for the bus from.
- INPUT TABLE…….. Select source table from INPUT operator, which you want to edit
- COLUMN FOR UPDATE…….. Select column from source table, which will be edited (for example flag setup location)
- VALUE FOR UPDATE…….. Select editable value (for example: flags)
- ID SOURCE RECORD……...Select record identification method (for example for flag setup); you select a unique column and column from input bus to OUTPUT SQL operator.

SETUP INPUT operator gives wide range of interactive with source data. After downloading the records, you can mark, which records has been downloaded. Thanks to this, you can perform data downloads in a clearly incremental way (setting a download condition to a selected flag value in INPUT operator)

USER SQL

Action, that allows you to act on your own. You can create any SQL code and point out a moment of its application (see picture below):

- PRE (before data stream adding action)
- POST (after data stream adding action)
- IN (during data stream adding action. Warning! The code is induced multiple times, after adding of each data stream record)

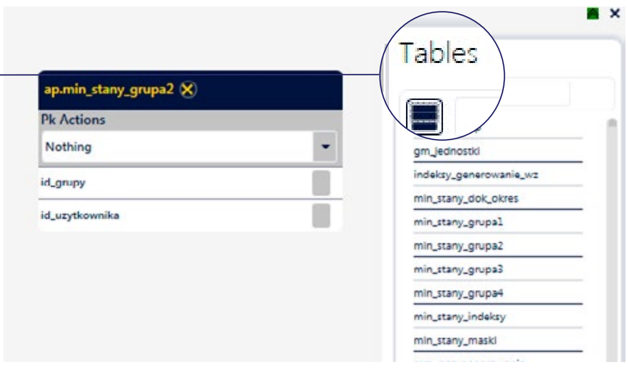


While writing your own SQL code, you can use signal values OUTPUT SQL input bus. You refer to the data stream column value using leading mark '@' and name of the bit (column) of input data stream (for example @id).

DATA STREAM ADDING CODE DECLARATION: SELECTING TARGET TABLE AND RULES OF DATA STREAM INPUTING.

STEP 1

On the right side of the window you can see all the tables located in the target database, pointed in PHYSICAL LINK (see picture below)



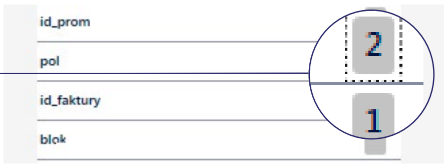
The selection of target table is performed through dragging and dropping the proper table into the working area.

On the working area you can drop few tables as target tables, however you must link them in a parent - child relation. In case you drop many tables independently (without setting the relation), the process will be stopped during processing.

After pointing target table, you can proceed to declare rules of data stream input in the table.

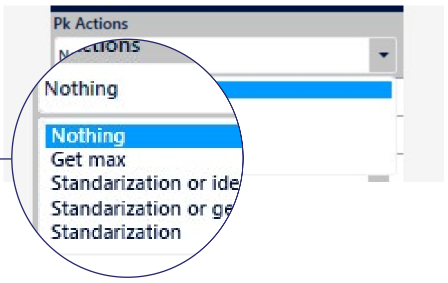
STEP 2

Now point PRIMARY KEY fields (PRIMARY KEY is a series of columns uniquely identifying the record), by marking of the selected columns (see picture below)



STEP 3

Now you should determine, how the process should act towards PRIMARY KEY fields during record addition (option available in the table header bar – see picture below)



You can choose between the following actions for PRIMARY KEY fields:

- NOTHING
- GET MAX
- STANDARIZATION OR IDENTITY
- STANDARIZATION OR GET MAX
- STANDARIZATION

Before we will describe in details each of the actions separately, we first want to create a standardization definition, in terms of OUTPUT SQL operator

What is standardization?

Standardization is a process of changing the value of a field (column) to a standard value, basing on a list matching current value with a standard value

In the context of OUTPUT SQL operator – standardizing list arises automatically – ad hoc – during the ongoing process through OUTPUT SQL operator, in case, when we face the situation of altering the value of PRIMARY KEY fields (columns)

Standardizing list is created by GRAVITY automatically and stored in the taget base.

Standardizing list is created for target table.

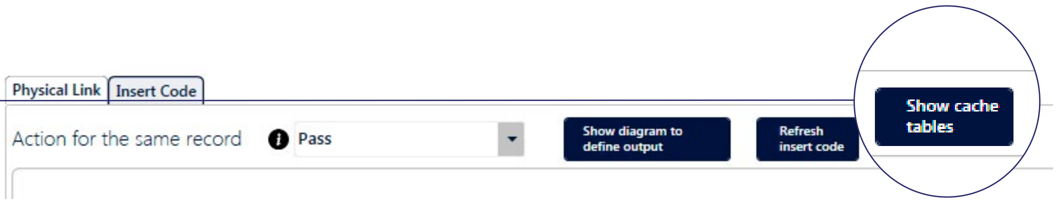
Why do we standardize?

Data stream stored in the target table, can have its origin in multiple independent data sources. In terms of this fact, to scale all the data in one database, you will have to make a decision on changing the values of unique fields (uniquely identifying), because in different sources the values can be the same.

How does the standardization work?

Technically speaking, it is an automatic action of creating recoding list. The recoding list (standardizing dictionary) is connected to the target table. Recoding connects source ID (source bit) from one side, and field values of old PK (unique IDs from the source) with unique IDs newly created in the target table.

If you want to see how the records look like in standardizing dictionaries, select SHOW CACHE TABLES option – see picture below.



If you already know what standardization is, you need to read the below usage description of each action against PRIMARY KEY field

NOTHING

PRIMARY KEY fields (uniquely identifying) remain unchanged

GET MAX

The process calculates new value for the last PRIMARY KEY field by incrementation of last record in target table. Standardizing dictionary is not built.

STANDARIZATION OR IDENTITY

In the first step – the process validates, if the fields were subject to earlier standardization. The control relies on examination of presence primal (source) Primary Key fields values in standardization dictionary. If the value is present, there is a change of a source value to standardized value (so a value from standardization dictionary). If the source value is not present in the standardization dictionary – ID given by the target base is downloaded and new value in standardization dictionary is added.

If you select this action – you will have to point out the source ID, pointing out a proper bit of input bus (if you have such bit) or entering own name, used always for this source of data.

STANDARIZATION OR GET MAX

In the first step – the process validates, if the fields were subject to earlier standardization. The control relies on examination of presence primal (source) Primary Key fields values in standardization dictionary. If the value is present, there is a change of a source value to standardized value (so a value from standardization dictionary). If the source value is not present in the standardization dictionary, GET MAX action takes place (see: GET MAX action description) and new value in standardization dictionary is added.

If you select this action – you will have to point out the source ID, pointing out a proper bit of input bus (if you have such bit) or entering own name, used always for this source of data.

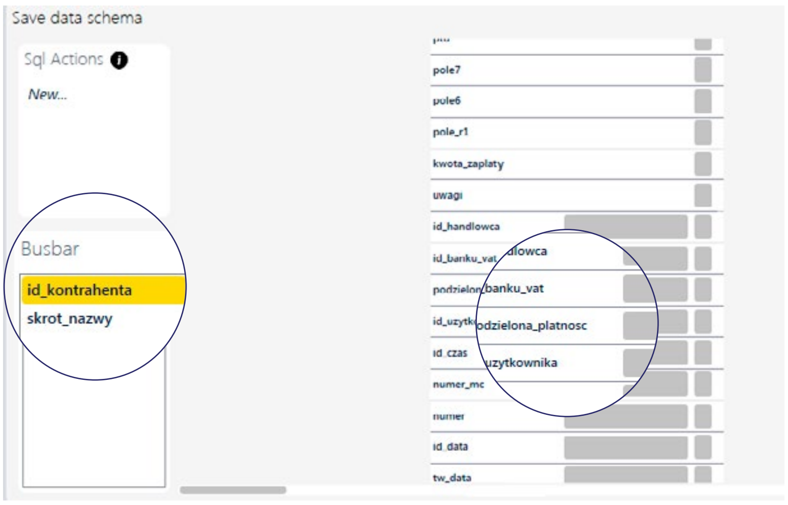
STANDARIZATION

Specific action. While proceeding the system checks if standardization dictionary already has PRIMARY KEY value. If not, then system fills standardization dictionary and adds a record. Otherwise (Primary Key field value already is in standardization dictionary) system does not add a new record, but uses UPDATE command, against identical record which was added earlier.

STEP 4

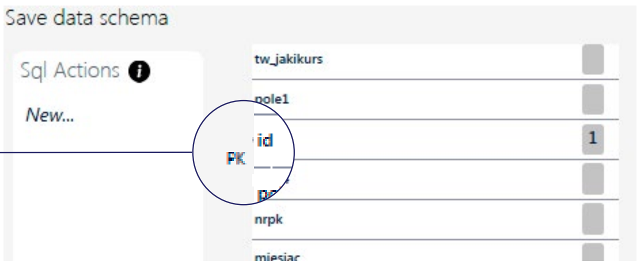
Now you should point out, how he input signal on the input bus translates to specific fields (columns) of target table. You can perform it as follows:

- Select input bus column (see picture below – marked area on your left hand side)
- System will automatically point out which fields are possible to compare by graying out (see picture below – area marked on you right hand side); those field are selected basing on compatibility of types of data.
- Choose table field by clicking on the proper column of target table.
- Select the next input bus column and repeat the actions

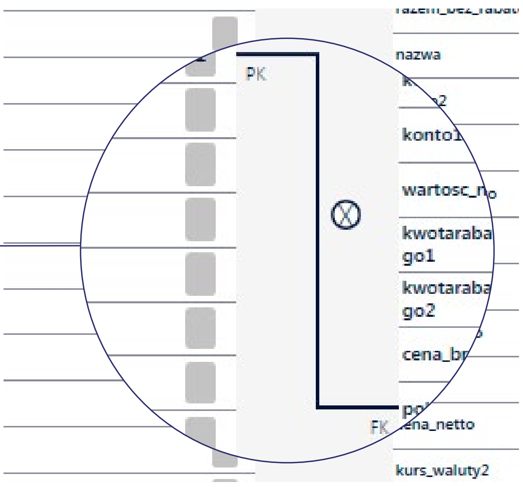


STEP 5

If you want, that the input bus data stream operated also on secondary tables against promary table, you can perform it in the next step. Select related table and place it on the working area – action similar as in case of main table: you need to drag and drop the related table from the list of available tables (list in the area on the left side) to the working area.



In the next step – choose in the "parent" table a connecting column by clicking the mouse – you will get a PK icon by this column (see picture below)



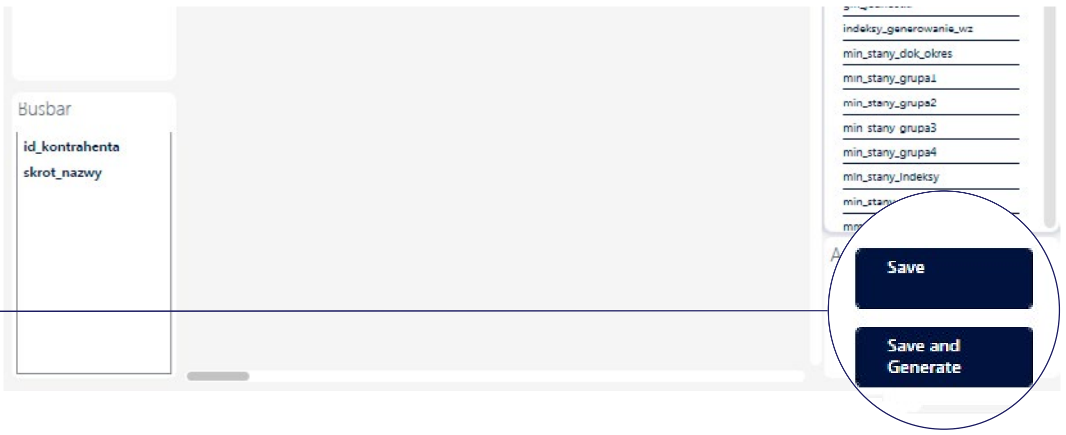
Now you need to connect it with the "child" table by pointing out connective column. You will get a line symbolizing the relation – as shown on the picture below.

In case of standarization of the connective fields, the system will automatically take care for field standardization, also in the "child" tables.

You can repeat this step multiple times, with any number of depth levels.

DATA STREAM ADDING CODE DECLARATION: SAVING THE CONFIGURATION

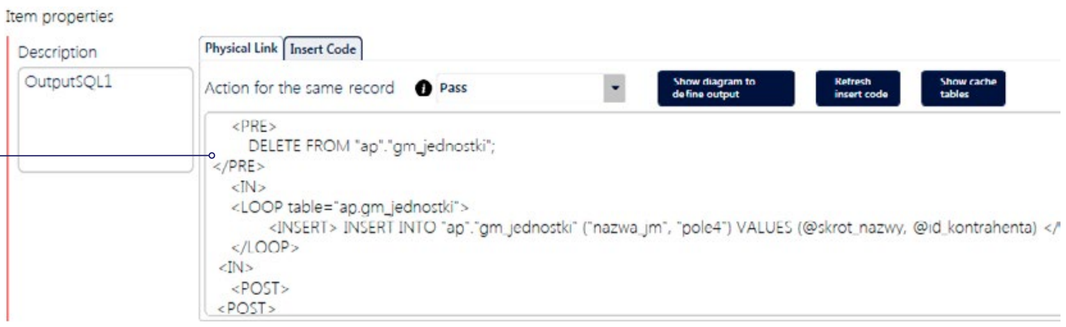
If you have experience – you an edit it, however it is not necessary: GRAVITY generates it independently,



If you set up all the actions and declared data stream input tables, you need to save your diagram

based on declared actions.

We have described above, how to use OUTPUT SQL operator. We do realize, that it is one of the most



Action for the same record Pass

```
  <PRE>
    DELETE FROM "ap"."gm_jednostki";
</PRE>
  <IN>
  <LOOP table="ap.gm_jednostki">
       <INSERT> INSERT INTO "ap"."gm_jednostki" ("nazwa_jm", "pole4") VALUES (@skrot_nazwy, @id_kontrahenta) </
  </LOOP>
 <IN>
  <POST>
 <POST>
```

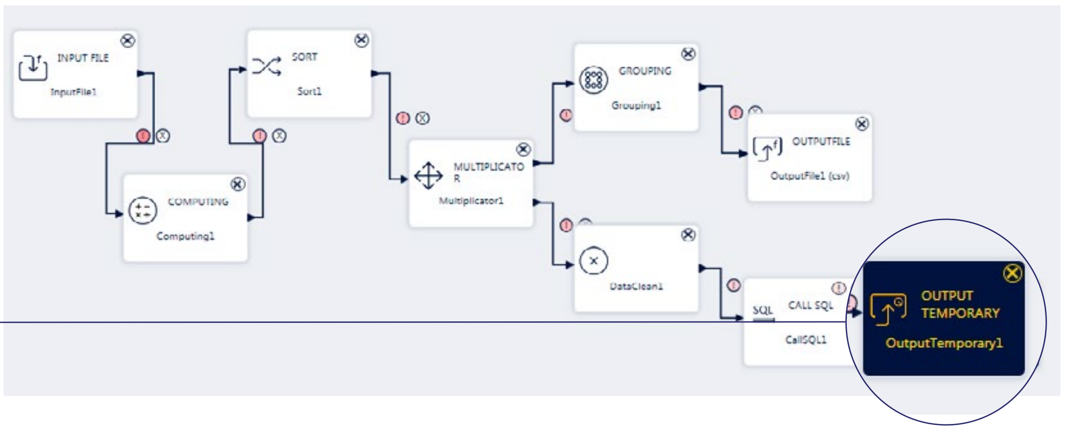
During saving GRAVITY automatically generates code in XML format. You can view it in INSERT CODE tab

complex GRAVITY components, therefore – for better picture of OUTPUT SQL operator action – we encourage you to get to know with examples (see chapter APPLICATION EXAMPLES). OUTPUT SQL operator, apart from regular context examples, we have devoted a separate paragraph USE OF OUTPUT SQL OPERATOR)

8.2.7 OUTPUT TEMPORARY

OUTPUT TEMPORARY operator has two applications:

- GRAVITY application blind data processing
- gathering of input data stream to be used by STANDARDIZATION operator

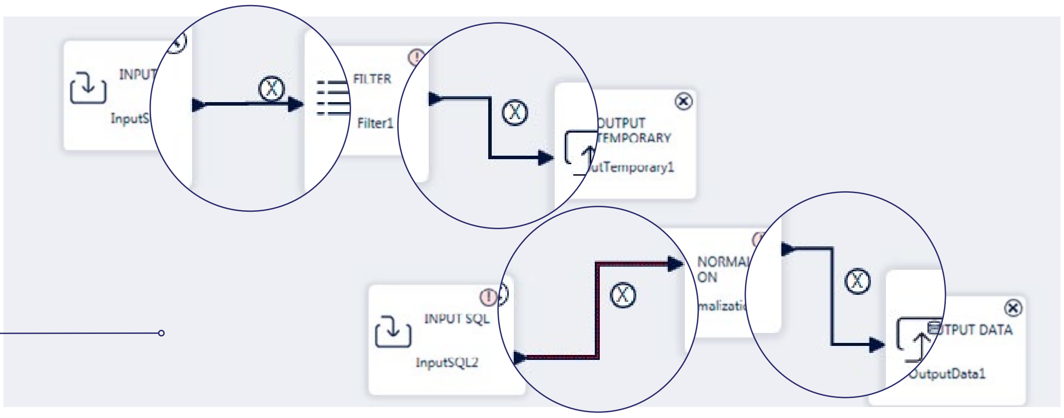


Example of GRAVITY project using OUTPUT TEMPORARY operator.

8.3 PROCESSING OPERATORS

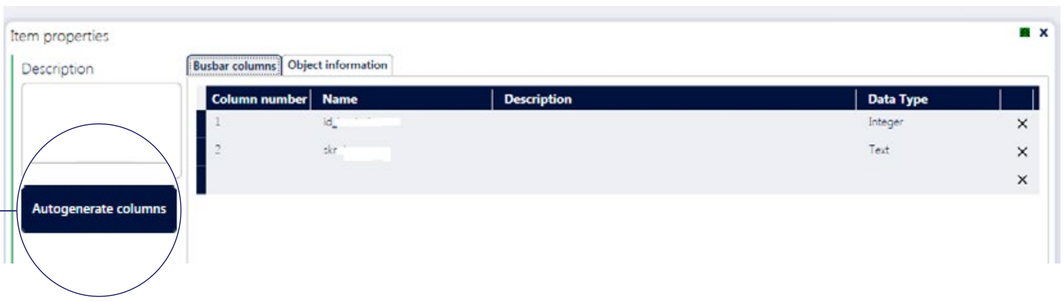
8.3.1 BUSBAR

BUSBAR is a bus connecting two operators. Processed data streams flow on the buses.



Buses example.

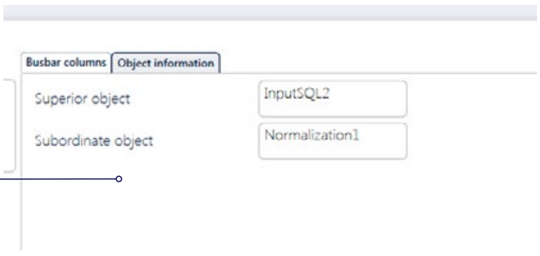
BUS CONFIGURATION



Column number	Name	Description	Data Type
1	id_		Integer
2	skr		Text

You can manually input the desctiptions to the columns, or once again generate columns basing on the source operator (

The essential characteristic of a bus is its width, so the columns transporting data in the data stream. After connecting two operators, the bus is automatically configured based on the source operator signal.



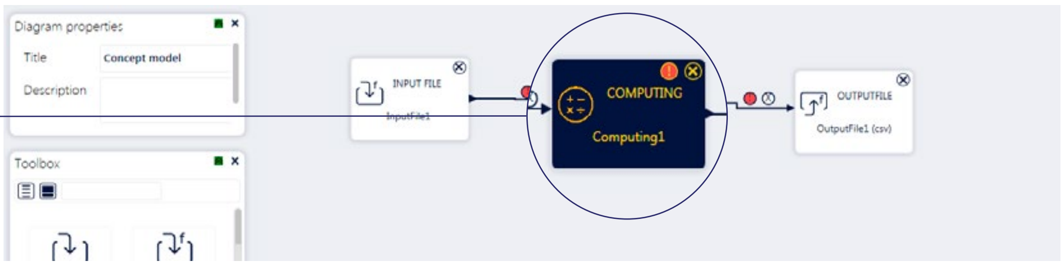
Superior object	InputSQL2
Subordinate object	Normalization1

In the OBJECT INFORMATION tab you can see the names of source operator and target operator

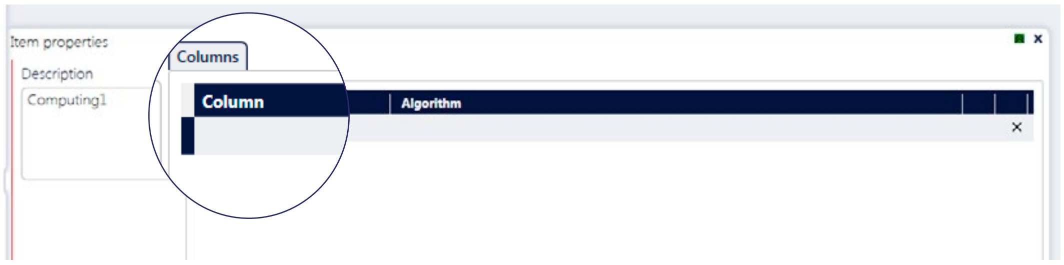
8.3.2 COMPUTING

COMPUTING operator processes the data stream located on operator output. The processing operation is performed for each data stream record. The result of the processing is an operator output data stream.

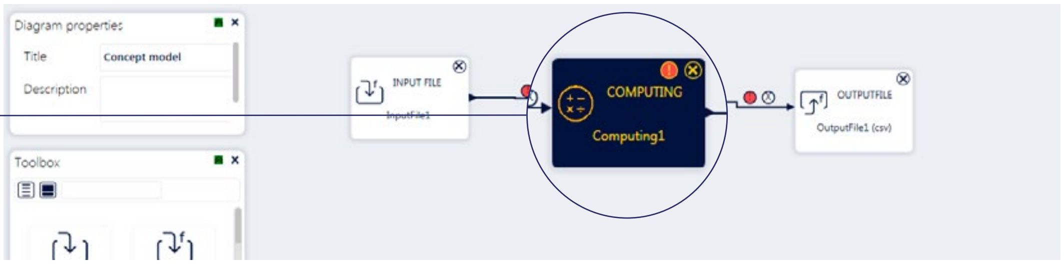
Example of GRAVITY project when using COMPUTING operator.



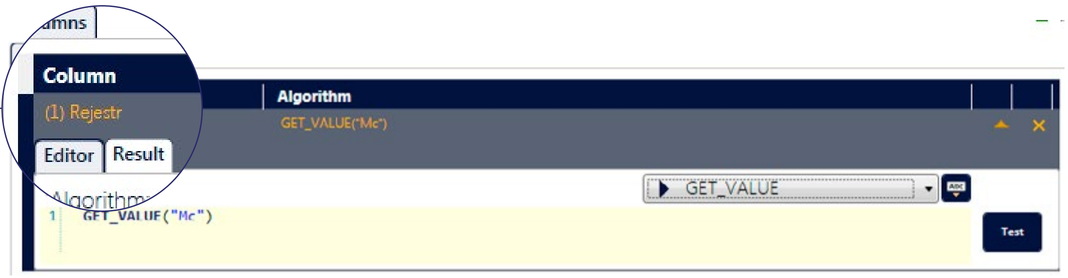
DATA STREAM ALGORITHMS PROCESSING CONFIGURATION



You can process any number of data stream columns, located on the bus input. It is enough, if you point out the column and edit processing algorithm.



The editor helps you: each of functions can be seletec from the dictionary



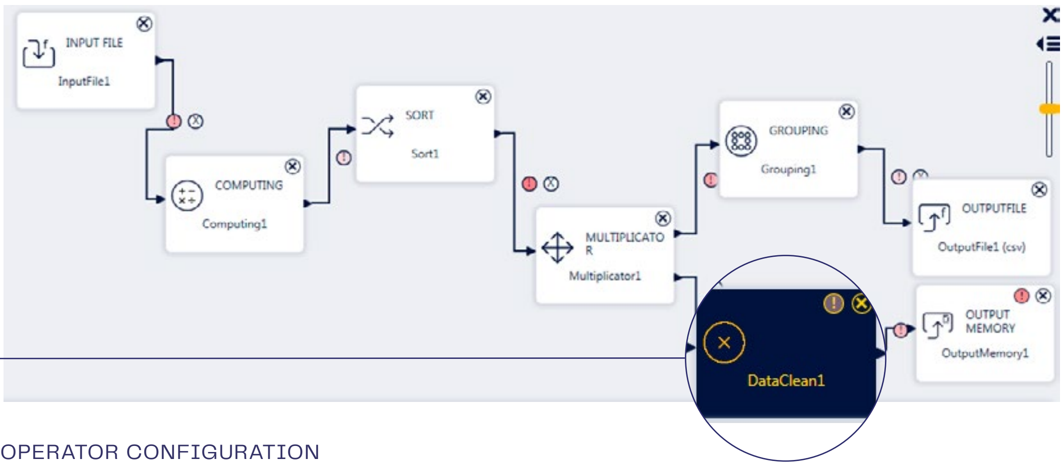
```
GET_VALUE("Mc")
```

In the algorithm you can use the following functions:

- GET_VALUE …. value of the selected bus column.
- GET_AGGR …. Sum of values of the selected bus column, for all data on inputciu
- GET_OUTTEMP_AGGR …. Sum of values of data column gathered in OUTPUT_TEMPORARY set.

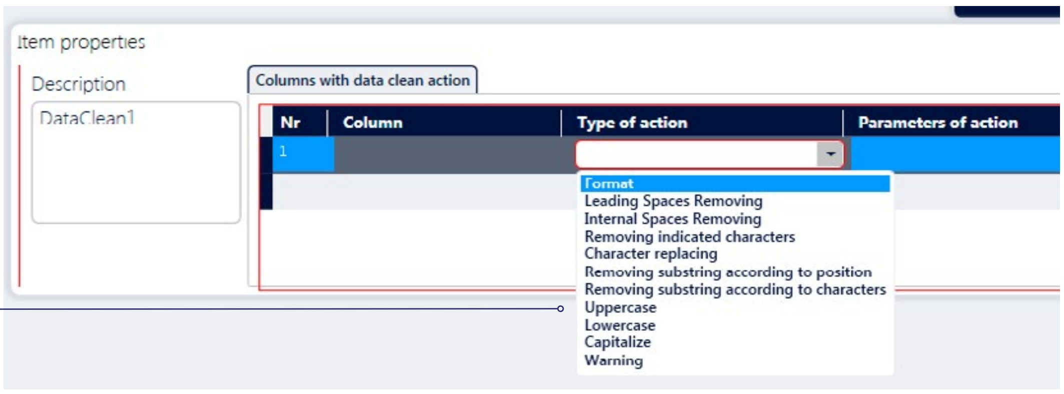
8.3.3 DATA CLEAN

DATA CLEAN operator enables the processing of each stream record located on the input bus, for selected columns according to rules chosen by the developer.



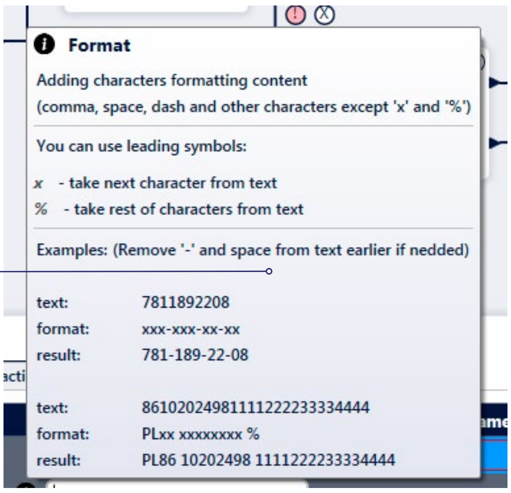
GRAVITY project example with use of DATA CLEAN operator.

OPERATOR CONFIGURATION



Each bus column can be subject to actions GRAVITY application example described below.

FORMAT



Format

Adding characters formatting content
(comma, space, dash and other characters except 'x' and '%')

You can use leading symbols:

x - take next character from text
% - take rest of characters from text

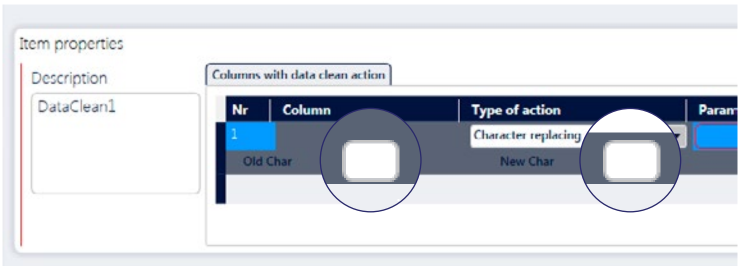
Examples: (Remove '-' and space from text earlier if nedded)

text:	7811892208
format:	xxx-xxx-xx-xx
result:	781-189-22-08
text:	86102024981111222233334444
format:	PLxx xxxxxxxx %
result:	PL86 10202498 1111222233334444

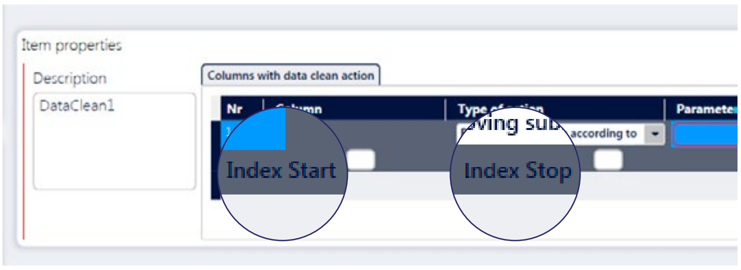
Data formatting for selected column, together with the rules

- LEADING SPACE REMOVING – Removal of leading spaces
- INTERNAL SPACE REMOVING – Removal of spaces occurring in between other characters.
- REMOVING INDICATED CHARACTERS – Removal of a selected sign from the column.

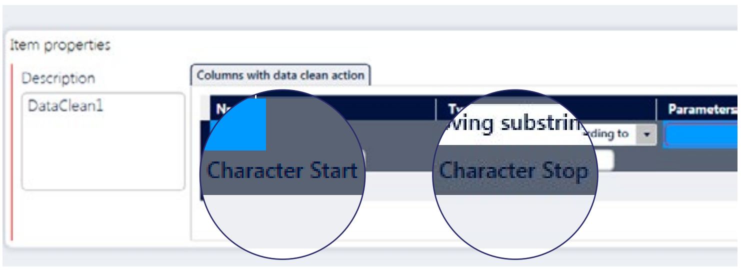
- CHARACTER REPLACING - exchanging the character (see picture below)



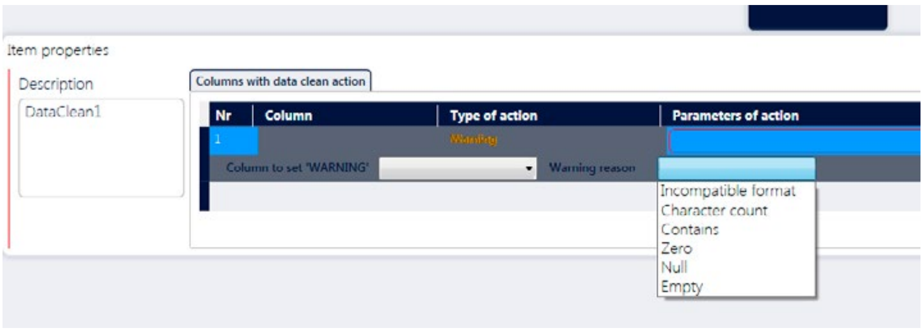
- REMOVING SUBSTRING ACORDING TO POSTITION - removal of the string of characters from a selected location. You configure the number of the beginning and end of deleted string (see picture below)



- REMOVING SUBSTRING ACORDING TO CHARACTERS - removal of string of characters. You configure the initial and the final character of the string to delete (see picture below)

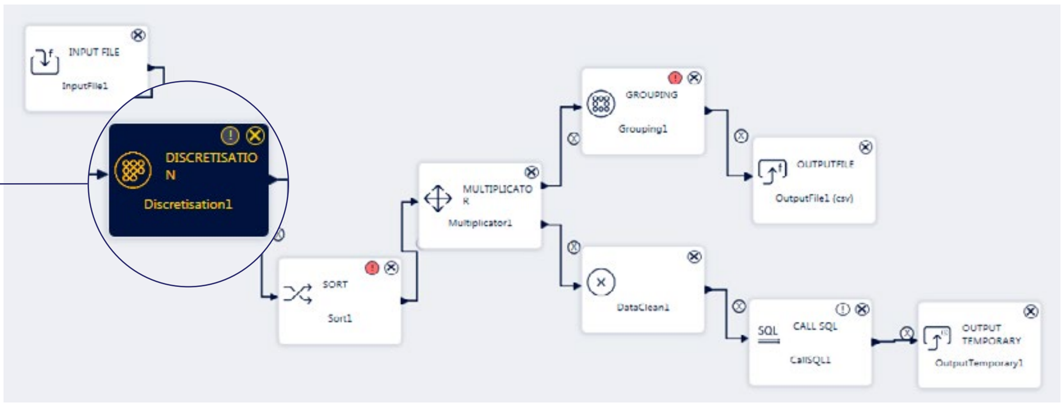


- UPPERCASE - after action, all characters in the selected column are uppercase.
- LOWERCASE - after action, all characters in the selected column are lowercase.
- CAPITALIZE- after action, first character is always uppercase.
- WARNING - warning generating action. During configuration we select the examined column, reason for releasing the warning (see the list on the picture below), and the column of data stream, in which potential warning should be placed.



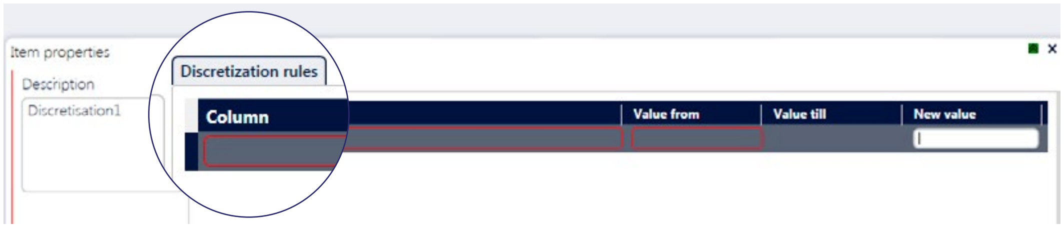
8.3.4 DISCRETISATION

DISCRETISATION operator enables transforming data values in the transformed stream from continuous values to discrete values. Ith means, that the operator changes the values belonging to data range into a single value (discrete), typical for certain range.



GRAVITY project example with DISCRETISATION operator in use.

OPERATOR CONFIGURATION

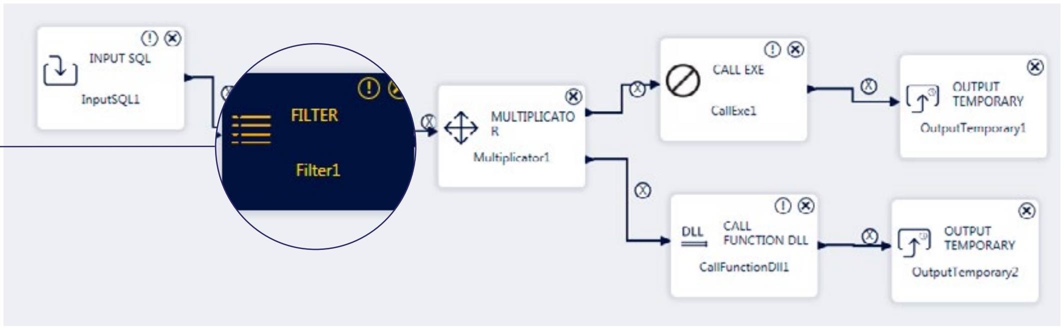


Column	Value from	Value till	New value

The user can discretisate a selected column from the input bus. The result of the operator action (in input bus), is a stream of data, in which discretization was performed on selected columns.

8.3.5 FILTER

FILTER operator enables filtering of the data stream from input bus. The bus has a filtered data stream on the output of the operator.



GRAVITY project example with use of FILTER operator.

FILTERING CONDITIONS SETUP

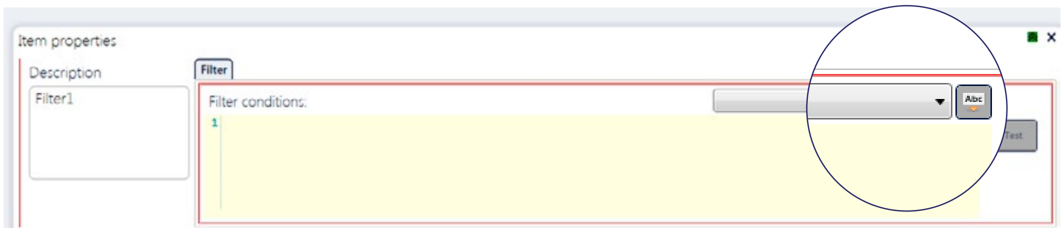
Simple variant

This variant allows you to use the paramters gathered in the dictionary: each column of data stream of input bus. It is enough, that you select the column and declare the condition. If you select more conditions, there will be AND interaction between them.

Basing on the simple variant, system automatically creates a more sophisticated code, available in the next tab.

Complex variant

If you want to create more complex filtering condition, you can use a compiled code in the next tab.

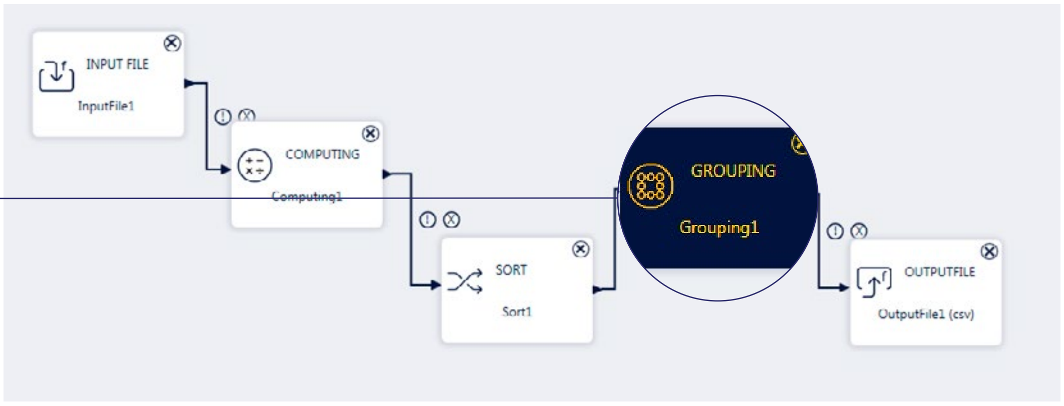


Optionally you can use the parameters gathered in the dictionary: each column of data stream on the input bus and aggregating functions:

- GET_AGGR - Sum of values of the selected bus column, for all data on the input
- GET_OUTTEMP_AGGR - Sum of values of data column gathered in the OUTPUT_TEMPORARY set.

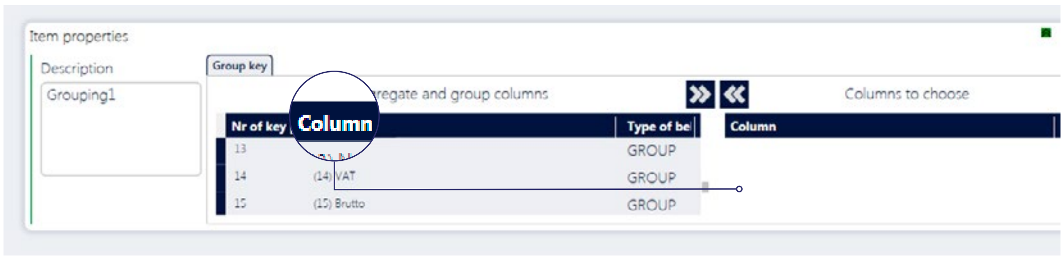
8.3.6 GROUPING

GROUPING operator enables processing of data stream on the input in this way that for selected grouping key (sequence of any number of selected columns) a rule of uniqueness of values is becoming valid (no repeat). That means, that the sequence of grouping columns can only appear once. For columns which are not included in the grouping key, aggregating function needs to be selected (which means how the column processing action should act in case of repeat)



GRAVITY project example with GROUPING operator in use.

GROUPING RULES CONFIGURATION



Nr of key	Column	Type of be
13		GROUP
14	(14) VAT	GROUP
15	(15) Brutto	GROUP

You can setup a grouping key, consisting of any number of bus columns, by using drag and drop action (you catch a input bus column, and drop it onto a grouping key list). Furthermore, for the remaining input bus columns, you set up aggregating function.

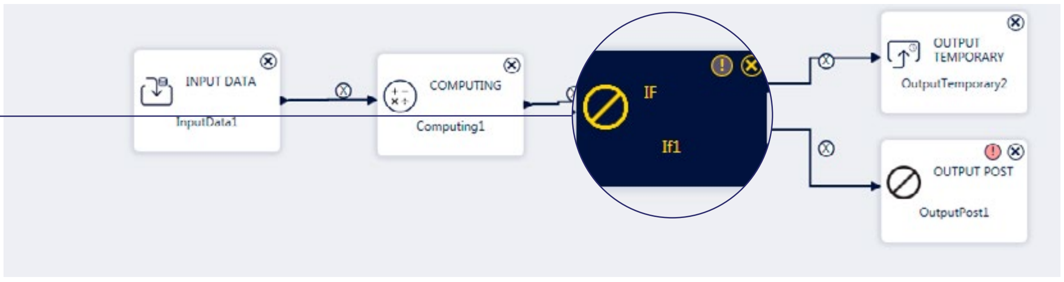
On operator output you will have processed data stream of the same width (which means same number of columns)

8.3.7 IF

IF operator gives you opportunity to direct all data stream into selected direction (you have two to choose from), depending on declared condition. If you want to execute multi-variant choice – you can place as many IF operator in series, as you need to complete your task.

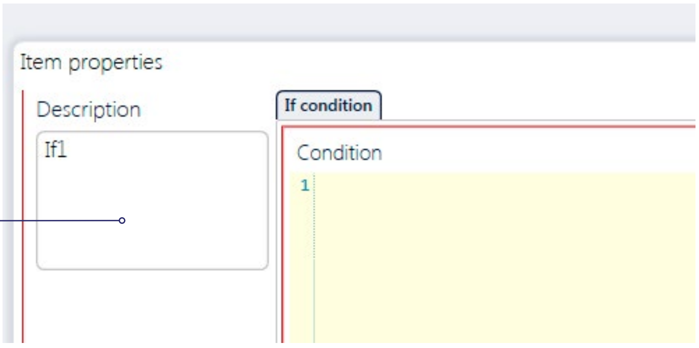
OPERATOR CONFIGURATION

GRAVITY project example with IF operator in use.



In this operator you can declare a condition based in input data stream – see picture below. During processing, if the condition is fulfilled, data stream will be on the upper output bus. In case of not fulfilling the condition, data stream will go through lower bus.

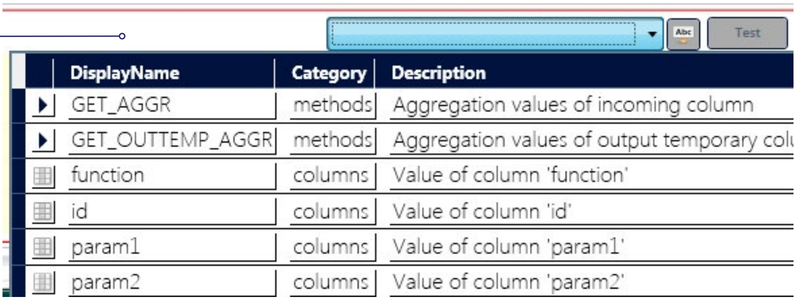
In this operator you can declare a condition based in input data



Available functions

- GET_AGGR - Sum of all values of selected bus column, for all input data.

During creation of the condition, you have possibility to refer to aggregating functions – all are available in the dictionary



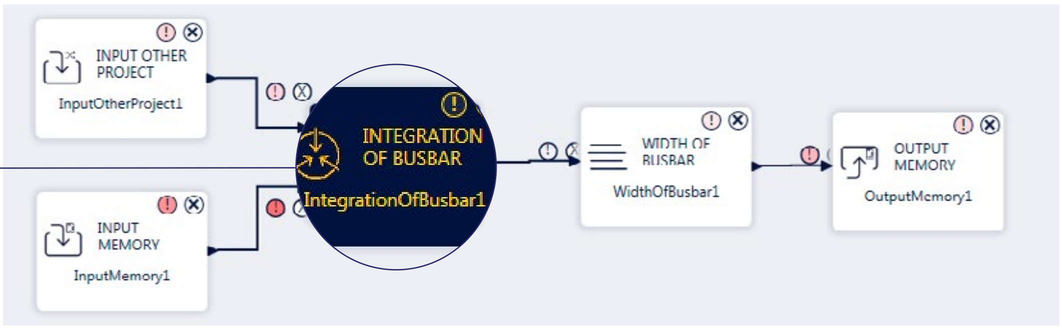
DisplayName	Category	Description
GET_AGGR	methods	Aggregation values of incoming column
GET_OUTTEMP_AGGR	methods	Aggregation values of output temporary colu
function	columns	Value of column 'function'
id	columns	Value of column 'id'
param1	columns	Value of column 'param1'
param2	columns	Value of column 'param2'

- GET_OUTTEMP_AGGR - Sum of values of data column, gathered in OUTPUT_TEMPORARY set.

8.3.8 INTEGRATION OF BUSBAR

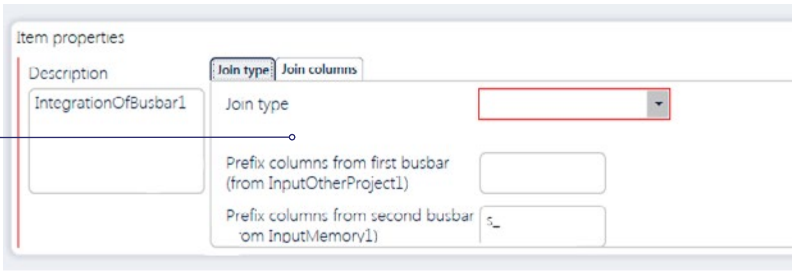
INTEGRATION OF BUSBAR operator enables connection of two data streams, according to rules selected by GRAVITY application designer. The bus on the operator output has width (column number) equal to sum of widths of input buses.

GRAVITY project example with us of INTEGRATION OF BUSBAR operator..



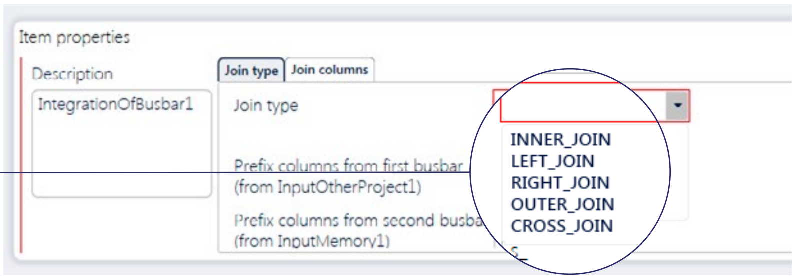
OPERATOR CONFIGURATION

Certainly you would like to have a precise description of each bit (column) of data stream, therefore you can select prefixes typical for input source, that will mark the selected columns of the output data stream


Item properties
Description: IntegrationOfBusbar1
Join type | Join columns
Join type
Prefix columns from first busbar (from InputOtherProject1)
Prefix columns from second busbar (from InputMemory1) s_

On the output bus we will receive a data stream, which width (column number) depends on the action that we will outsource to the operator.

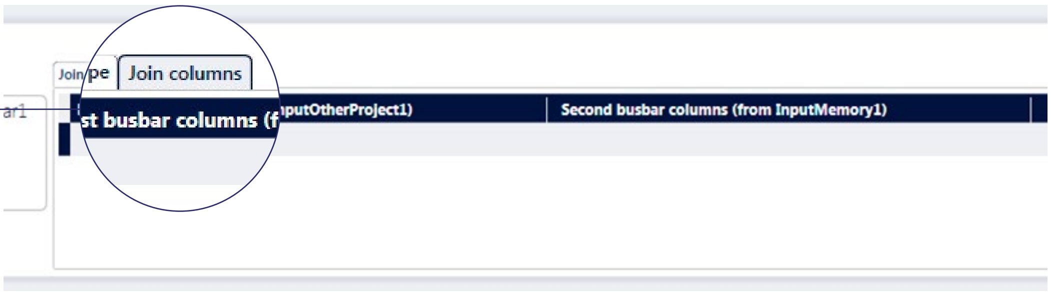
You can chose between the actions typical for SQL language


Item properties
Description: IntegrationOfBusbar1
Join type | Join columns
Join type
INNER_JOIN
LEFT_JOIN
RIGHT_JOIN
OUTER_JOIN
CROSS_JOIN
Prefix columns from first busbar (from InputOtherProject1)
Prefix columns from second busbar (from InputMemory1)

While configuring two buses integration, you will have to make a decision, on which rules you want to connect the records from two data streams

- INNER JOIN – Thanks to this function, you will connect those records from two buses, that have equal values of connecting columns (definition of connecting columns – see picture below). Number of records on the output bus is a product of all matches of connecting columns (common part). Records from both streams, that cannot be connected, are filtered out.
- LEFT JOIN – For this action, let's assume that the upper bus is the parent (goes into the operator on top).
  The output bus includes:
  - » All matched records (see: INNER JOIN action)
    Matching is according to connecting columns (see picture below)
  - » All the records of parent bus, not matched, expanded by child bus column with NULL values.

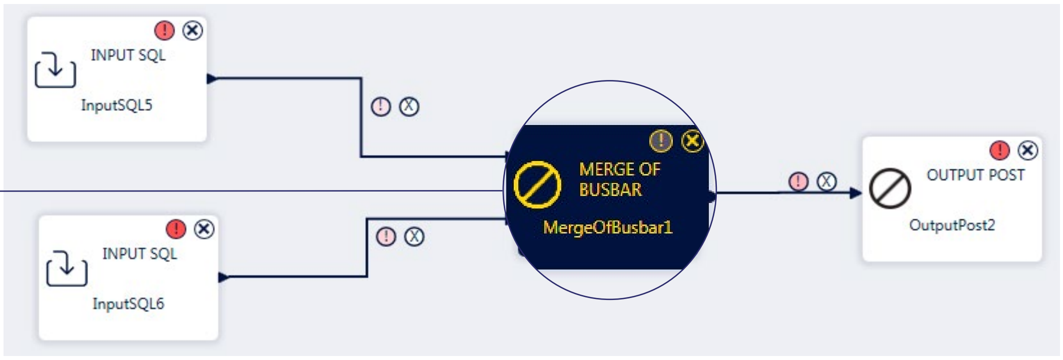
- RIGHT JOIN – for this action, let's assume that the lower bus is the parent (goes into the operator on bottom). The output bus includes:
  - » All matched records (see: INNER JOIN action)
    Matching is according to connecting columns (see picture below)
  - » All the records of parent bus, not matched, expanded by child bus column with NULL values.
- OUTER JOIN – during this action, the operator passes through all the records, that have not been connected during LEFT JOIN action (have NULL on extended bus) and also will pass through all the records, that have not been connected during RIGHT JOIN action (have NULL on extended bus), and also all the records, that during LEFT JOIN (or RIGHT JOIN) actions have been connected with records from child bus (common part).
- CROSS JOIN – Output bus will run a data stream, that is a Cartesian product of both streams (matrix).



On JOIN COLUMN tab you can select connecting columns GRAVITY application example

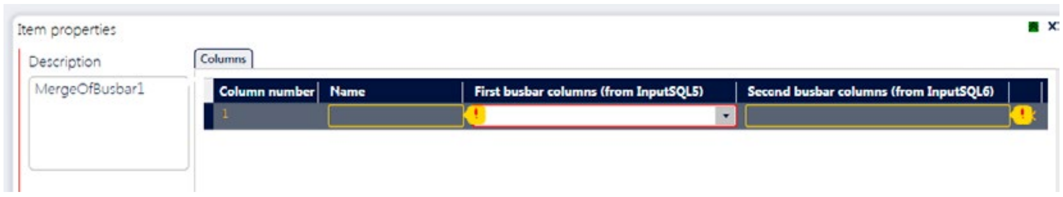
8.3.9 MERGE OF BUSBAR

MERGE OF BUSBAR operator allows connection od two data streams of the same nature. Data stream on output operator is equal to sum of lenghts of two input data streams, while its' widths is equal to the width of the wider of two connected strams.



GRAVITY project example with use of MERGE OF BUSBAR operator.

OPERATOR CONFIGURATION



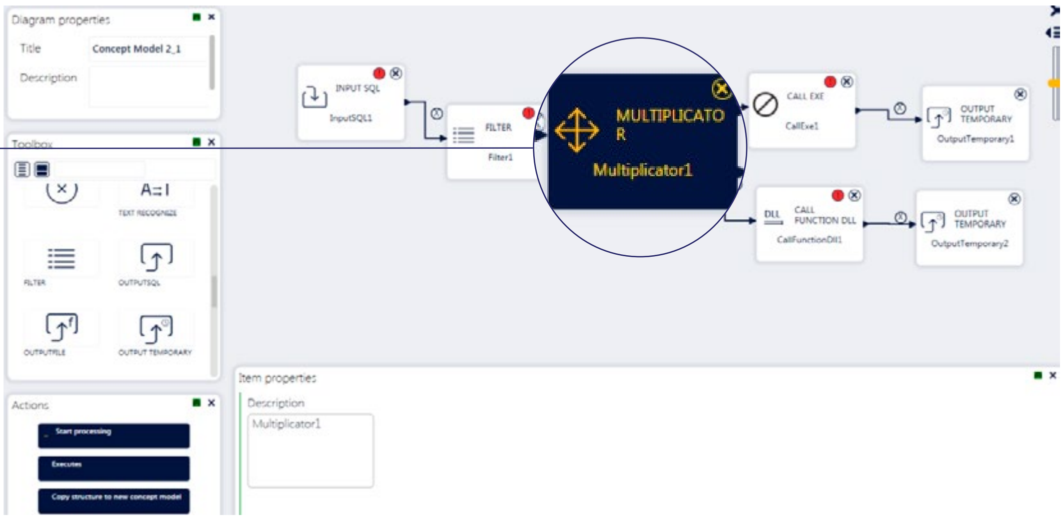
Column number	Name	First busbar columns (from InputSQL5)	Second busbar columns (from InputSQL6)
1			

Each bit of input data bus, should be declared, by selecting the bits from input buses (connected streams buses)

The length of the output stream (not to be confused with width), will be equal to the sum of lengths of streams entering the operator.

8.3.10 MULTIPLICATOR

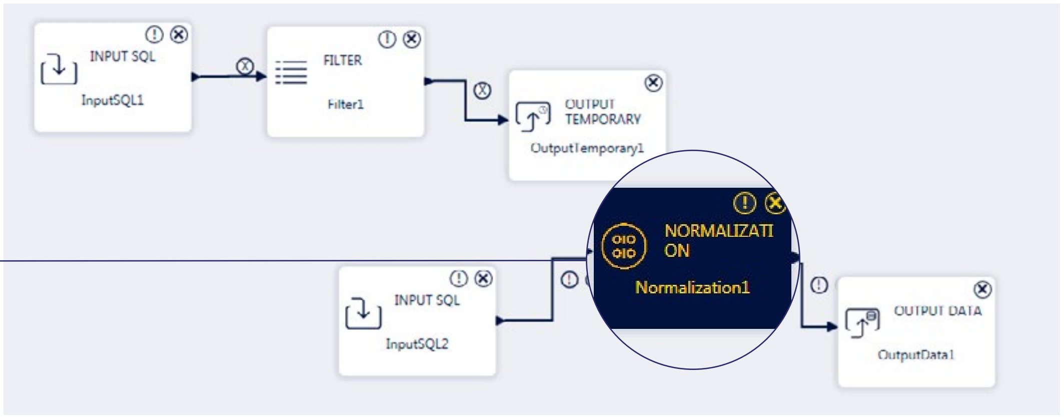
MULTIPLICATOR operator allows transformation of input bus, into two buses containing an identical data stream.



GRAVITY project example with use of MULTIPLICATOR operator.

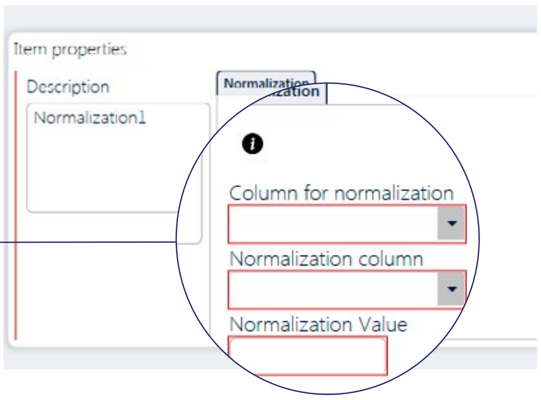
8.3.11 NORMALIZATION

NORMALIZATION operator allows bringing the data to a common unit. During the action, data is calculated according to the converter included in outer set, containing the converters.



GRAVITY project example with use of NORMALIZATION operator.

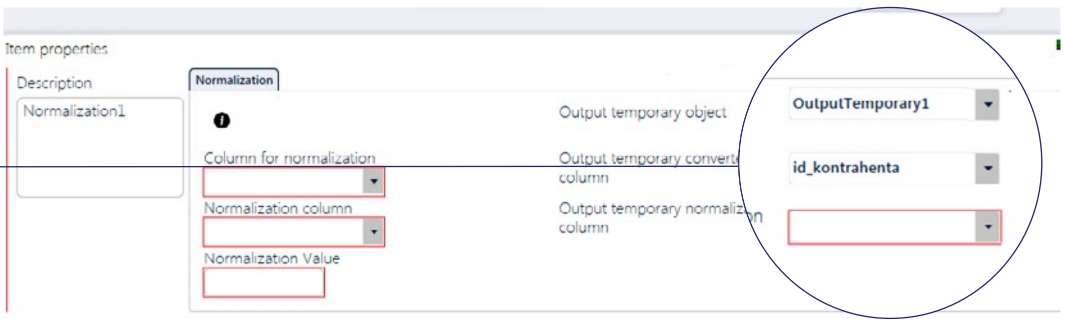
OPERATOR CONFIGURATION



To configure the operator, you need to select

- Input bus column, that will be noralized (COLUMN FOR NORMALIZATION) (this column will be transformed)
- Normalization column (column that will contain the measure)
- The name of normalization unit (NORMALIZATION VALUE) (description of a universal unit)

To configure the operator, you need to select the conversion method


Item properties

Description: Normalization1

Normalization

Column for normalization:

Normalization column:

Normalization Value:

Output temporary object: OutputTemporary1

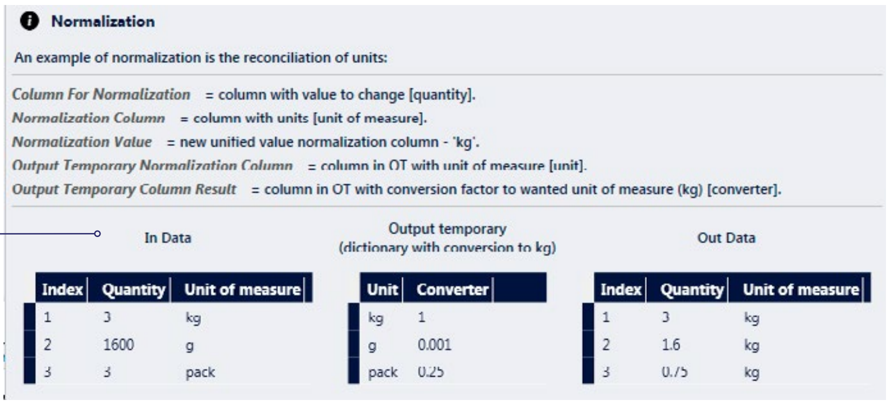
Output temporary converte column: id_kontrahenta

Output temporary normaliz column:

To configure the operator, you need to select the conversion method:

- External data, that will be the basis for the conversion (data is included in OUTPUT TEMPORARY operator, as a result of other transformation data)
- OUTPUT TEMPORARY column (OUTPUL TEMPORARY NORMALIZATION COLUMN) (conversion multiplier column; output data is a product of input data and the multiplier)

Example of normalization action


Normalization

An example of normalization is the reconciliation of units:

Column For Normalization = column with value to change [quantity].
Normalization Column = column with units [unit of measure].
Normalization Value = new unified value normalization column - 'kg'.
Output Temporary Normalization Column = column in OT with unit of measure [unit].
Output Temporary Column Result = column in OT with conversion factor to wanted unit of measure (kg) [converter].

In Data

Index	Quantity	Unit of measure
1	3	kg
2	1600	g
3	3	pack

Output temporary
(dictionary with conversion to kg)

Unit	Converter
kg	1
g	0.001
pack	0.25

Out Data

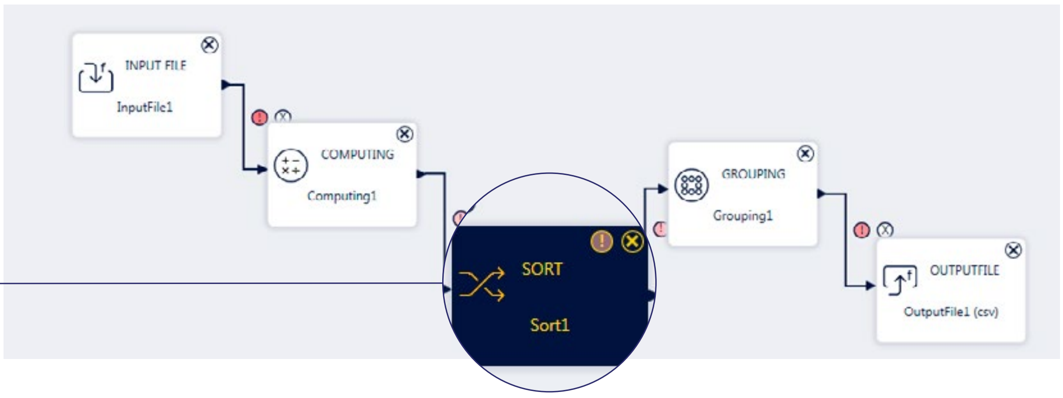
Index	Quantity	Unit of measure
1	3	kg
2	1.6	kg
3	0.75	kg

8.3.12 OCR

OCR operator changes the graphic object into text, by recognizing text in the graphic object.

8.3.13 SORT

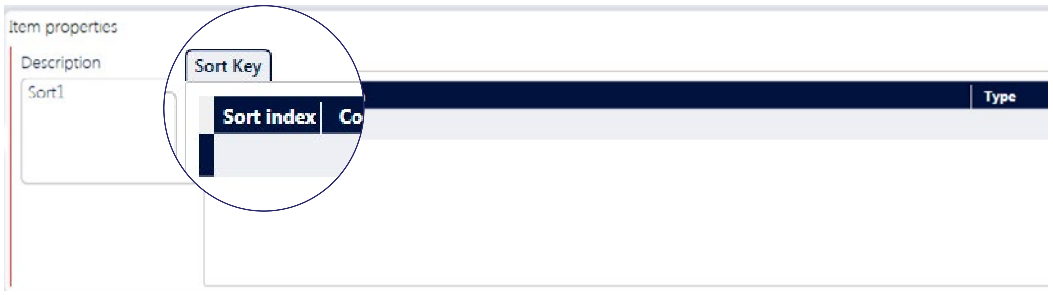
SORT operator enables change of order of the records in the data stream located on the input bus. On the output data bus, a sequenced stream is located, according to the conditions declared on SORT operator.



GRAVITY project example with use of SORT operator.

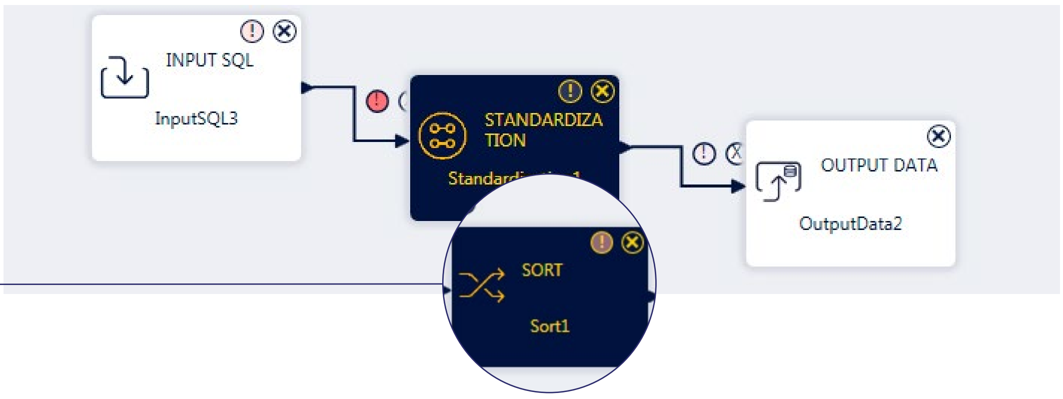
SORTING KEY CONFIGURATION

You can setup a sorting key, consisting of any number of bus columns. For each column you determine type, or direction of sorting (from biggest to lowest or opposite).



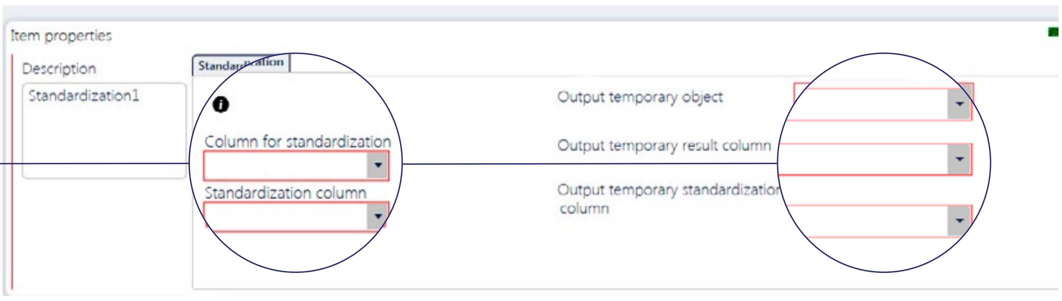
8.3.14 STANDARIZATION

STANDARDIZATION operator allows the change of data value, based on universal identifier, contained in one of input stream columns. New value is searched based on universal ID, taken from the independent list.



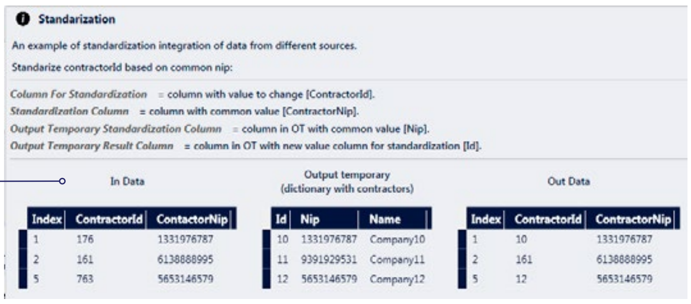
GRAVITY project example with use of STANDARDIZATION operator.

OPERATOR CONFIGURATION



When configuring the operator, you must select

- Input bus column, subjected to standardization (COLUMN FOR STANDARDIZATION) (this column will be transformed)
- STANDARDIZATION COLUMN (column that will contain universal identifier)
- Output data, based on which we will make the conversion (data included in OUTPUT TEMPORARY operator, as a result of transformation stream)
- OUTPUT TEMPORARY STANDARDIZATION COLUMN (column containing universal ID; including operator bus input stream data)
- OUTPUT TEMPORARY RESULT COLUMN (result column)



Standarization

An example of standardization integration of data from different sources.

Standarize contractorId based on common nip:

Column For Standardization = column with value to change [ContractorId].
Standardization Column = column with common value [ContractorNip].
Output Temporary Standardization Column = column in OT with common value [Nip].
Output Temporary Result Column = column in OT with new value column for standardization [Id].

In Data

Index	ContractorId	ContactorNip
1	176	1331976787
2	161	6138888995
5	763	5653146579

Output temporary (dictionary with contractors)

Id	Nip	Name
10	1331976787	Company10
11	9391929531	Company11
12	5653146579	Company12

Out Data

Index	ContractorId	ContractorNip
1	10	1331976787
2	161	6138888995
5	12	5653146579

Expample of standardization action

8.3.15 STANDARIZATION OUTPUT

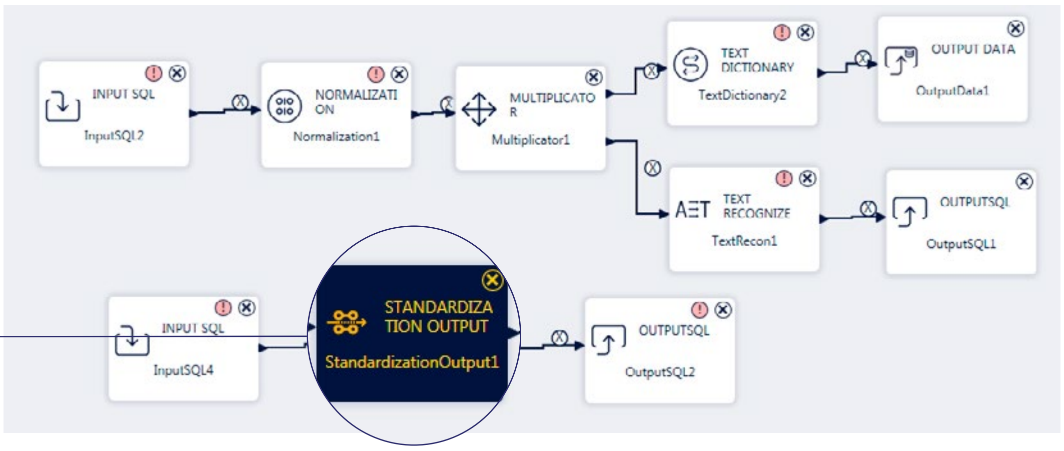
STANDARDIZATION OUTPUT operator is running an action, which is a specific type of standardization.

Context:

Regular STANDARDIZATION operator enables change of data value, based on universal ID included in one of the input stream columns. New value is searched based on universal ID taken from the independent list.

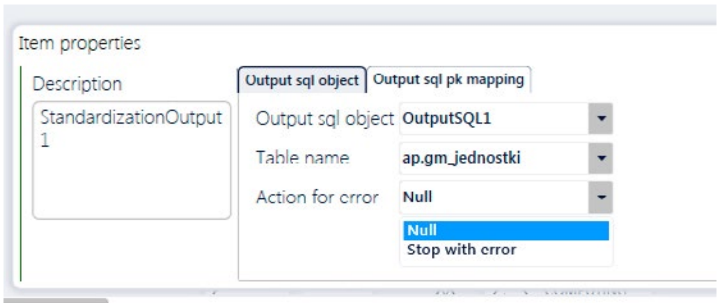
STANDARIZATION OUTPUT definition

STANDARDIZATION OUTPUT operator enables change of data value (column), based on the list created automatically ad hoc by OUTPUT SQL operator, during data saving in the database. There is a possibility of such configuration of OUTPUT SQL, that some of data (columns) changed the value (as it happens with unique ID's of universal objects, in case that come from multiple sources.) Those changes are saved in the recoding tables, created in the target base created by GRAVITY, and available through STANDARDIZATION OUTPUT operator. In other words – you can imagine, that this is a standardization based not on external OUTPUT TEMPORARY list, but on recoding list created by OUTPUT SQL operator.



GRAVITY project example with use of STANDARDIZATION operator.

OPERATOR CONFIGURATION



Item properties

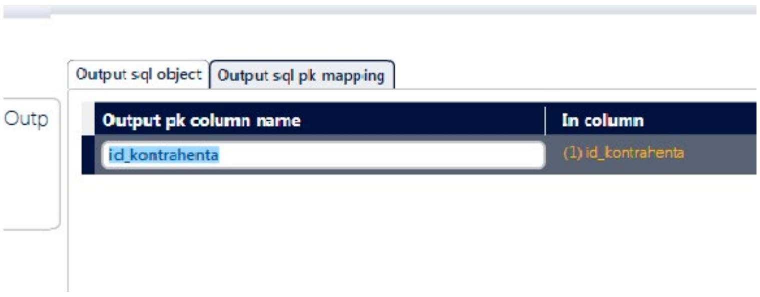
Description: StandardizationOutput1

Output sql object | Output sql pk mapping

Output sql object	OutputSQL1
Table name	ap.gm_jednostki
Action for error	Null

Null
Stop with error

In the first step GRAVITY application example, you must select an OUTPUT SQL operator (GRAVITY application can have multiple such operators), responsible for creating recoding list, target table that is standardized during OUTPUT SQL action (GRAVITY will recommend only those tables, which take part in the selected OUTPUT SQL), and declare GRAVITY mode of action in case of error detection (for example missing position on recoding list) during standardization. You can decide on “stop GRAVITY action” or “proceed”, giving NULL as a transformed value.



Outp

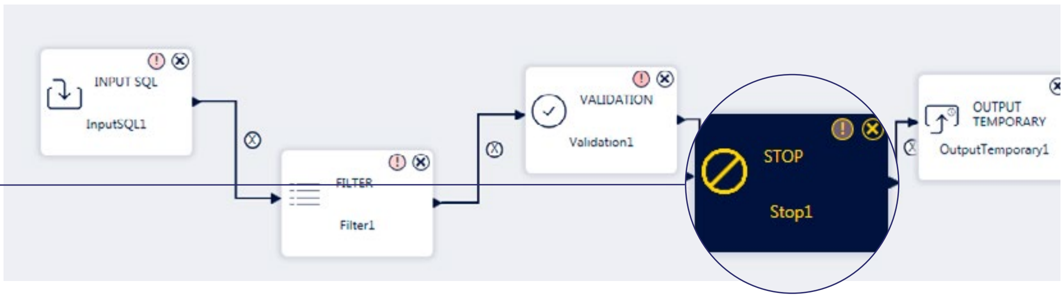
Output sql object | Output sql pk mapping

Output pk column name	In column
id_kontrahenta	(1) id_kontrahenta

In OUTPUT SQL PK tab, you need to bind a column from input bus, with identical column of list created by OUTPUT SQL operator

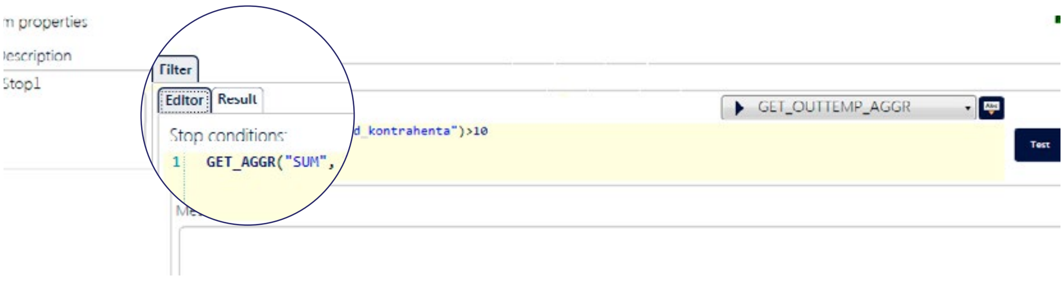
8.3.16 STOP

STOP operator allows you to stop the whole data transformation process in case of fulfilling a declared condition.



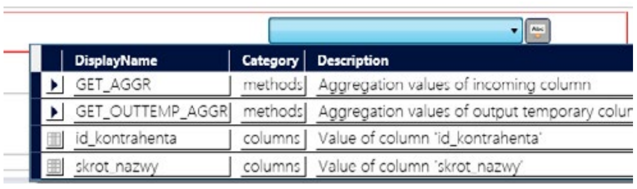
GRAVITY project example with use of STOP operator.

OPERATOR CONFIGURATION



```
GET_AGGR("SUM",
d kontrahenta")>10
```

When declaring a stop condition GRAVITY application example you have a list of functions available (see picture below)



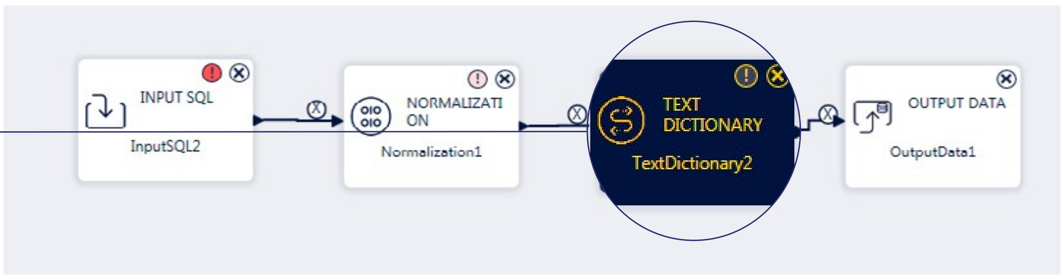
DisplayName	Category	Description
GET_AGGR	methods	Aggregation values of incoming column
GET_OUTTEMP_AGGR	methods	Aggregation values of output temporary colur
id_kontrahenta	columns	Value of column 'id_kontrahenta'
skrot_nazwy	columns	Value of column 'skrot_nazwy'

Available aggregating function:

- GET_AGGR …a function that sums the value of selected column in the data stream
- GET_OUTTEMP_AGGR ...a function that sums the value of selected column in the OUTPUT TEMPORARY set

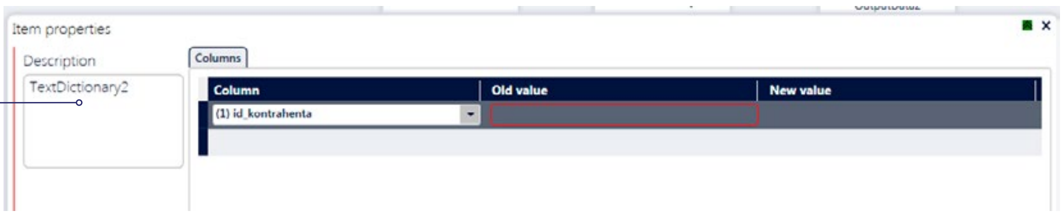
8.3.17 TEXT DICTIONARY

TEXT DICTIONARY operator changes the value of a selected column, to a new value, according to the created list.



GRAVITY project example, with use of TEXT DICTIONARY operator.

OPERATOR CONFIGURATION



Item properties

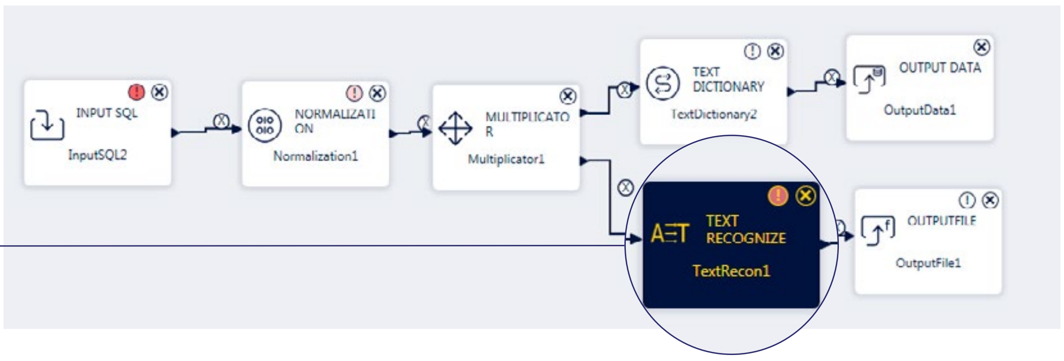
Description: TextDictionary2

Column	Old value	New value
(1) id_kontrahenta		

You create the recoding list by selecting a column, the current value and a new value. A list can have any number of recodings.

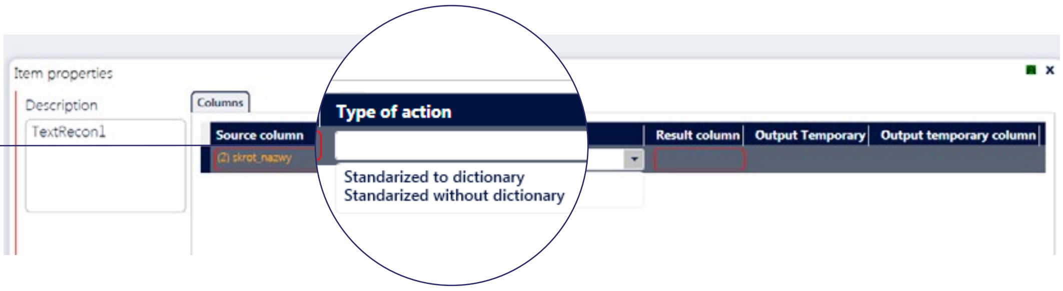
8.3.18 TEXT RECOGNIZE

TEXT RECOGNIZE operator transforms the data, by comparing the value of data to a universal dictionary. In case it does not detect the data on the list, it finds a value included on the list, that is most likely to the listed value, and changes the current value to the listed value. The universal list can arise in one of two ways: it can be selected by the user (unanimously declared) or be created automatically based on repetition scale test.



GRAVITY project example, with TEXT RECOGNIZE operator in use.

OPERATOR CONFIGURATION



Item properties

Description: TextRecon1

Source column	Type of action	Result column	Output Temporary	Output temporary column
(2) skrot_nazwy				

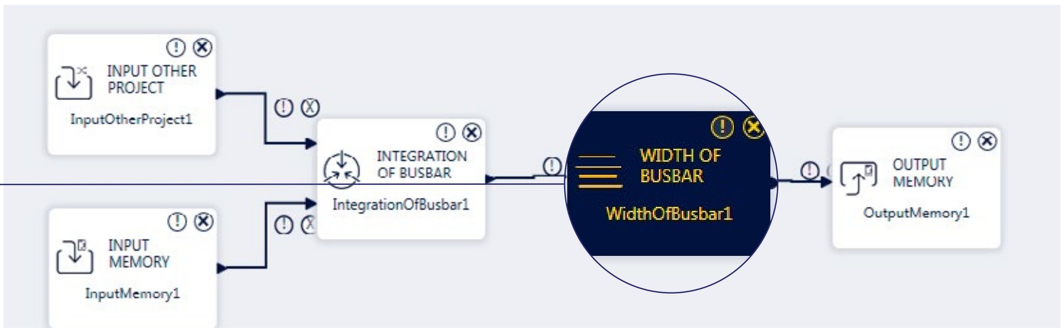
Type of action options: Standarized to dictionary; Standarized without dictionary

During configuration, you select a column subjected to diagnosis and diagnosis action variant.

If you select diagnosis variant based on permanent dictionary, you must additionally select OUTPUT TEMPORARY set, which includes a universal list and a column that stays related to the column examined during the action

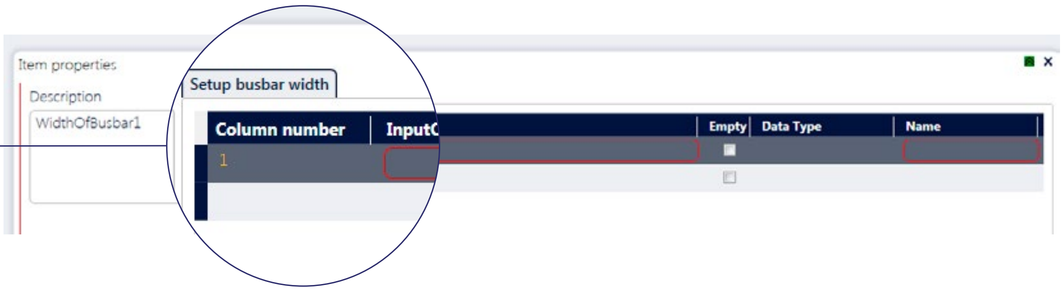
8.3.19 WIDTH OF BUSBAR

WIDTH OF BUSBAR operator allows to change the width of data bus. Output bus – its' width and name of columns – is declared during configuration, based on the columns of input bus.



GRAVITY project example, with WIDTH OF BUSBAR operator in use

OPERATOR CONFIGURATION

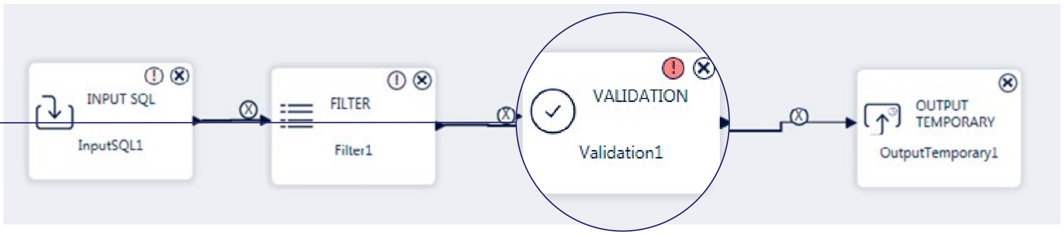


Column number	InputC	Empty	Data Type	Name
1				

By configuring, you will be able to declare each element of width of output bus (which means: every column) based on the columds of input bus GRAVITY application *example*

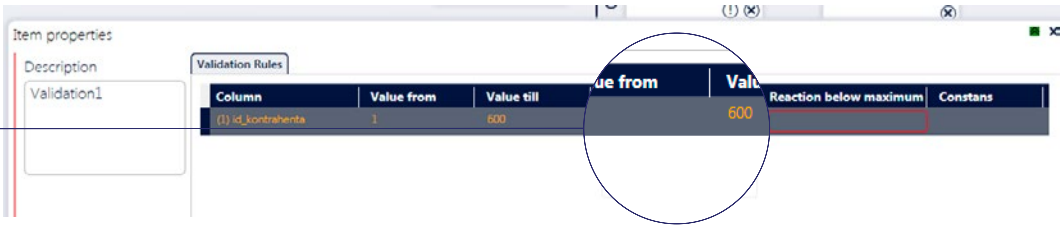
8.3.20 VALIDATION

VALIDATION operator examines the values of declared data (columns) of the stream on the input bus. In case of detection of values not belonging to declared (validated) ranges, system performs one of actions: gives new value based on the limit (minimum or maximum – depending on where it exceeded), constant value (declared) or deletes the record from the data stream.



GRAVITY project example, with VALIDATION operator in use.

OPERATOR CONFIGURATION



Column	Value from	Value till	Reaction below maximum	Constans
(1) id_kontrahenta	1	600		

You must declare a column of input bus

You must declare a column of input bus (see picture abive), which is subject to validation, range of values which is valid and type of action in case of exceeding the ranges. You can declare independent actions in case of exceeding minimal and maximal. You can decide on the following actions:

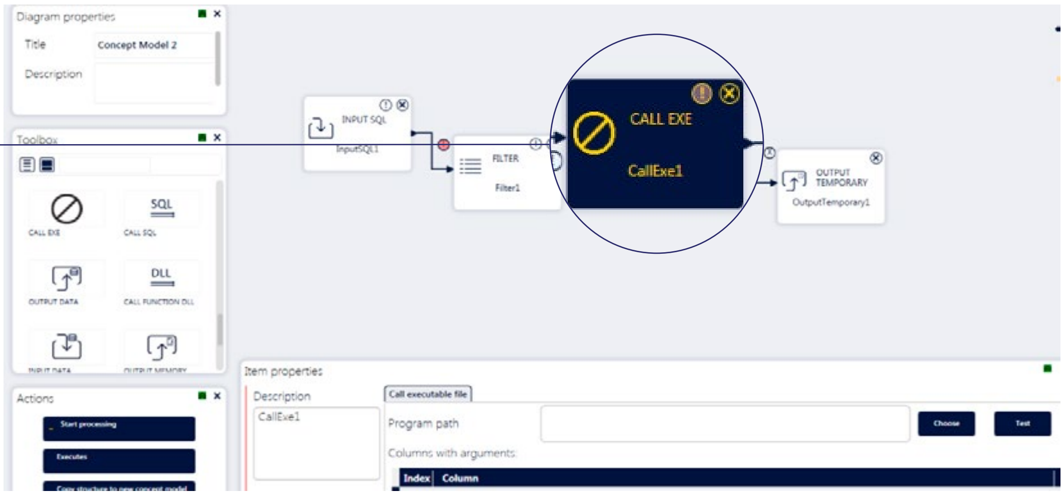
- Give a new value equal to minimal (if the value is below minimal)
- Give a new value equal to maximal (if the value is above maximal)
- Give constans value, if the value is out of validation range
- Filter out the record if it is out if validation range.

8.4 INTERNAL TRANSFORMATION OPERATORS

8.4.1 CALL EXE

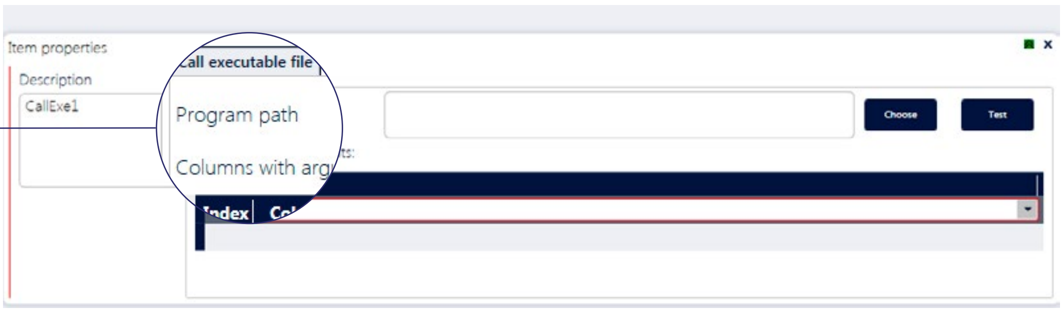
CALL EXE operator enables the call of an external EXE program, for each record located on the input operator bus. Possible external program call with parameters located on data bus.

GRAVITY project example, with CALL EXE operator in use.



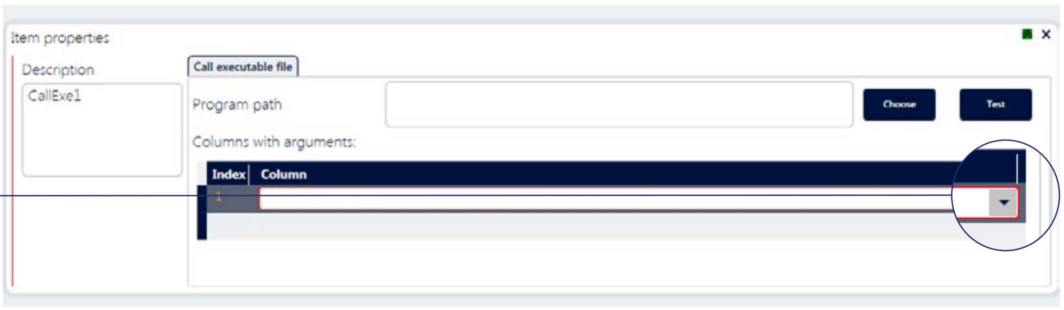
PROGRAM PATH: PARAMETER SETUP

This option allows you to set acces path to the external program



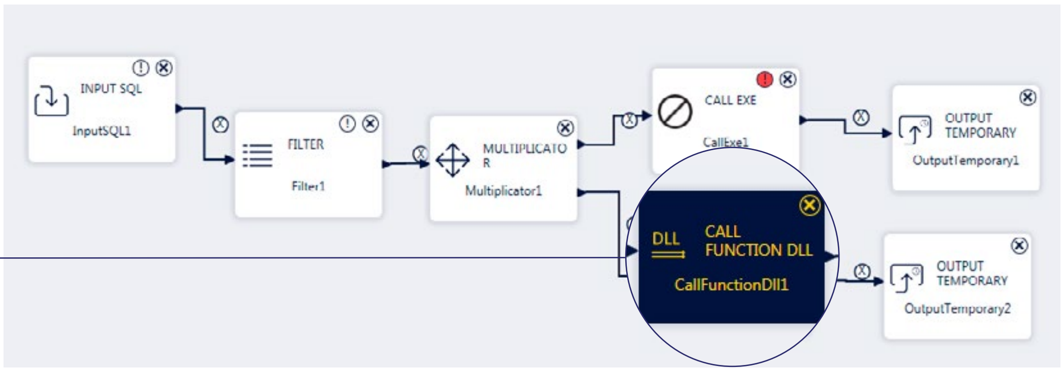
SETTING UP THE ARGUMENT OF AN EXTERNAL PROGRAM CALL

As an option, you set up potential call arguments of external program. As a parameter, we select bus column on operator's input.



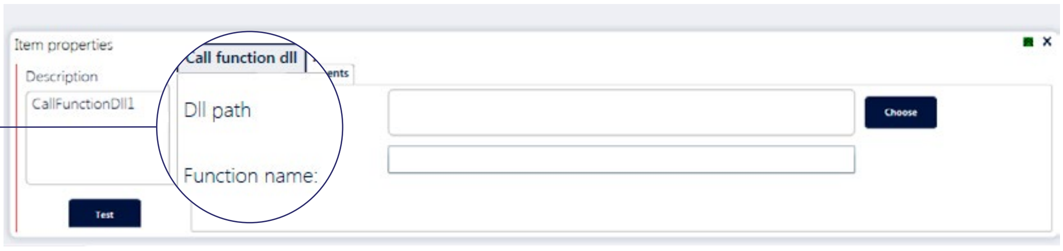
8.4.2 CALL FUNCTION DLL

CALL FUNCTION DLL allows to call an external function, located in a DLL library, for each record located on operator's input bus. Possible external function call with parameters located on the data bus.

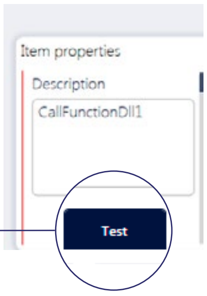


GRAVITY project example, with CALL DLL operator in use

PARAMETER SETUP: PROGRAM PATH



As an option, you setup access path for external function and declare name of the function.



The option above allows you to test the call of declared external function.

SELECTING LOCATION INSIDE DATA BUS AS A RESULT CARRIER

Select which column of the data bus will be the result carrier of an external function.

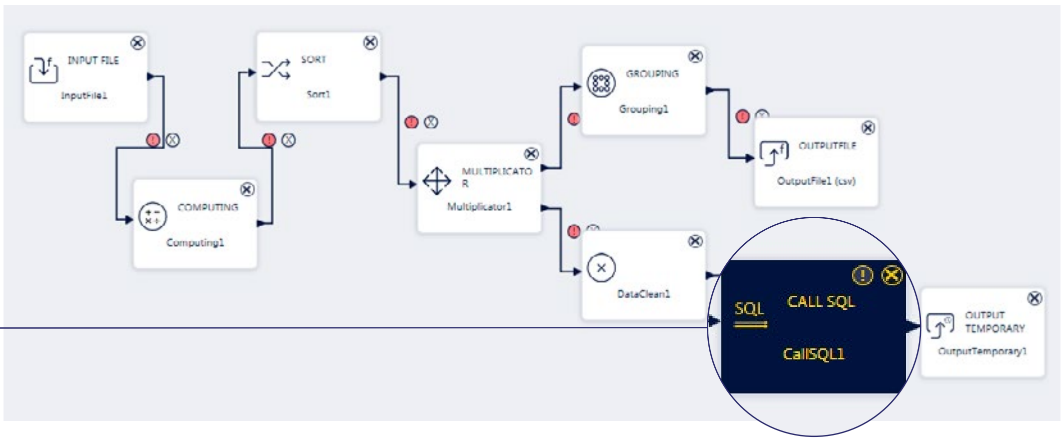
For proper use of external function, you need to know type of data returned by external function. By editing the field, a list with variants of data types will be recommended automatically.

SETUP OF ARGUMENT OF EXTERNAL FUNCTION CALL

In this option, you will setup potential arguments of calling of external function. As a parameter we select databus column of operator's input.

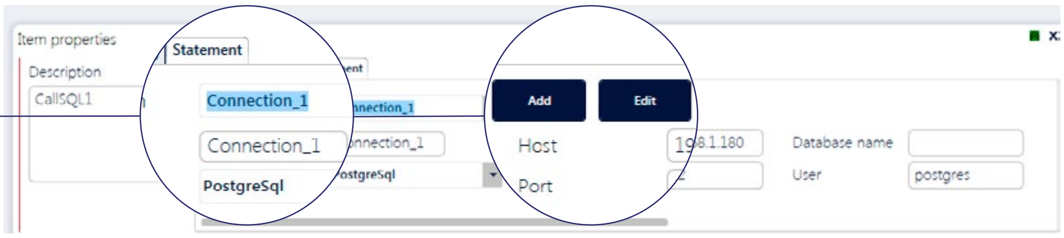
8.4.3 CALL SQL

CALL SQL operator call SQL action on declared SQL database, using the data located on the input bus.



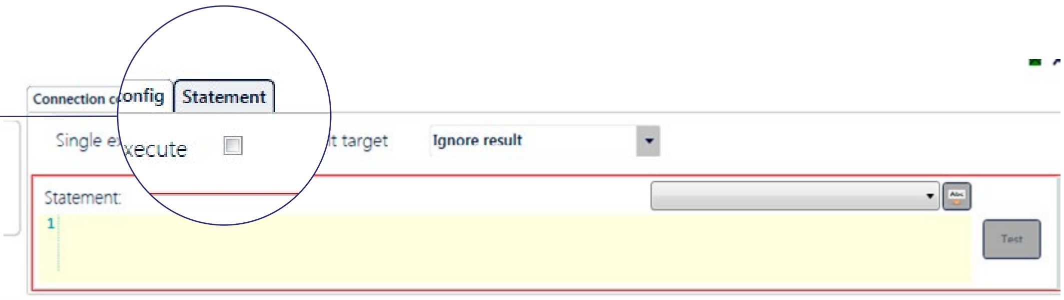
GRAVITY project example, with CALL SQL operator in use.

OPERATOR CONFIGURATION



The connection can be selected through a dictionary, if you defined such connection earlier, or define a new one (options ADD or EDIT)

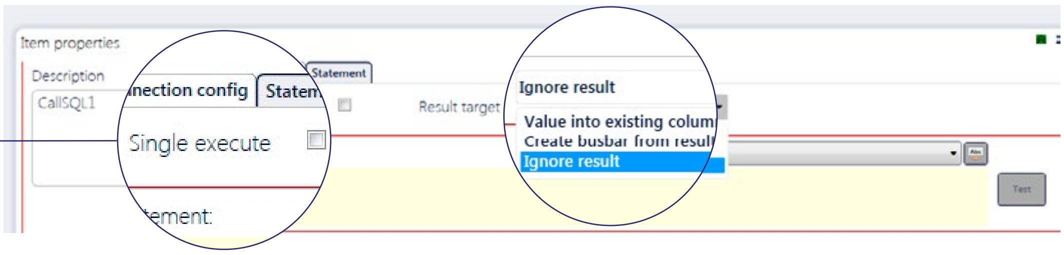
The operator is used to execute SQL code in the database, so first of all you need to select a connection to a selected database.



Możesz dokonać wyboru bez kłopotliwej edycji

In the STATEMENT tab, yoy can edit SQL expression. Access to the parameters (including all the columns of input data) is ensured by a list, from which you can make a selection without problematic edit GRAVITY application example

Whole expression can be tested by choosing TEST option.



You can however decide on single call variant: you need to select SINGLE EXECUTE option

You have two options of using the operator. First choice variant is a variant of calling SQL code for each record, located in the data stream of input data bus.

SQL code can return a result, that is transformed in a different way, depending in RESULT TARGET setup GRAVITY application example:

- IGNORE RESULT – SQL code result has no influence at all: the data stream on bus output, will be identical as on bus input
- VALUE INTO EXISTING COLUMN – SQL code result will be located in the selected column of output bus (remaining input bus data will remain their values)
- CREATE BUSBAR FROM RESULT – output bus is compatible with the result generated by SQL code (applies for SINGLE EXECUTE)

8.4.4 CALL WEB SERVICE

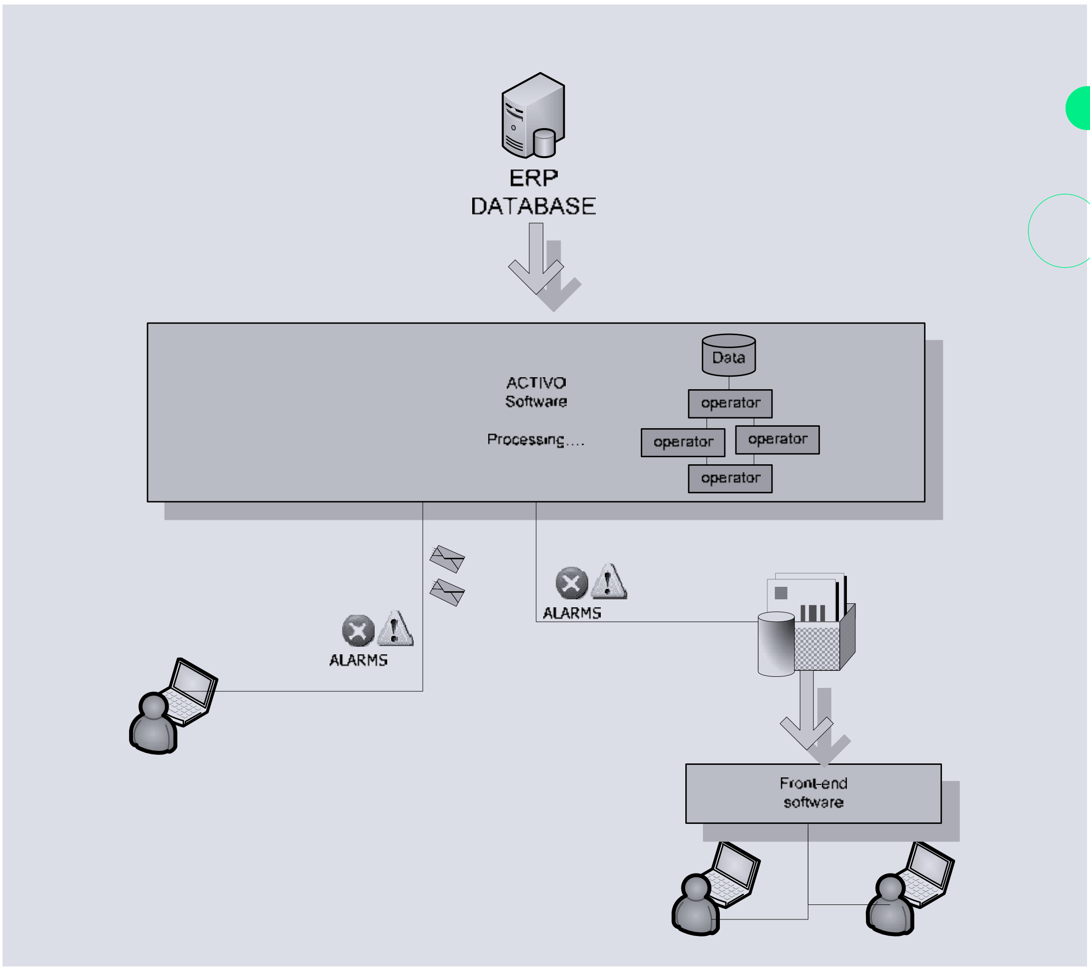
CALL WEB SERVICE operator calls an external WEB SERVICE function. In that case a called external WEB SERVICE function plays the role of external GRAVITY operator. On CALL WEB SERVICE operator input a data bus is located, and the data stream running through this bus can be used as input data stream for CALL WEB SERVICE operator. By configuring the operator you declare a structure of JSON protocol, that runs the external WEB SERVICE function (the declaration is obviously fully automatic: you declare basing on a sample JSON file and the signal on input bus). Furthermore, you configure the structure of transformation result data stream, provided that this is a JSON protocol. The declaration is automiatic, basing on the sample file and the signal of output bus.

GRAVITY operates WEB SERVICE servers with REST protocol.

9. APPLICATION EXAMPLES

Below you can find examples of GRAVITY applications. This is only a handful of inventions. The ideas can be infinite.

9.1 INTERNAL ALARMS



CONTEXT

An enterprise using ERP class software wants to create an automatic employees notification system. Notification system has to be automatic and non user-dependent. The system has to automatically detect alert situations and notify proper recipients.

TASK

Create a functionality, which will help the user immediately react on sales decrease.

GRAVITY APPLICATION EXAMPLE

For our example, let's say we need to inform on sales decrease in the last 7 days, under condition, that the 7-days sale was lower, than the median for last 12 months

I case of identifying a condition that we have set up, GRAVITY sends an e-mail with information to the configured recipient.

Impulse to trigger the processing: scheduled calls scanning.

TASK

Create a functionality, which will help the user immediately react on exceeding trade credit by the contractor.

GRAVITY APPLICATION EXAMPLE

In case of condition identification – exceeding the trade credit – GRAVITY send an e-mail with information to the confiruged recipient (recipients)

Impulse to trigger the processing: scheduled calls scanning.

GRAVITY APPLICATION EXAMPLE

As above, with one crucial difference. Triggering impulse is database notification. The moment of releasing the notification is adding contractor invoice record.

GRAVITY APPLICATION EXAMPLE

Identical as above, with the difference, that information on detected event is sent as SMS message to a selected person (use of CALL WEB SERVICE as an external function, servicing the SMS gate.)

TASK

Create a functionality, that will help the user to immediately receive information on on expired customer's order execution.

GRAVITY APPLICATION EXAMPLE

In case of identifying the condition (existing expiries), GRAVITY sends e-mail with information to configured recipient (recipients)

Impulse to trigger the processing: scheduled calls scanning.

GRAVITY APPLICATION EXAMPLE

In case of identifying the condition (existing expiries), GRAVITY leaves the information on each expiry on the leading data bus, to the temporary data base (session memory cache)

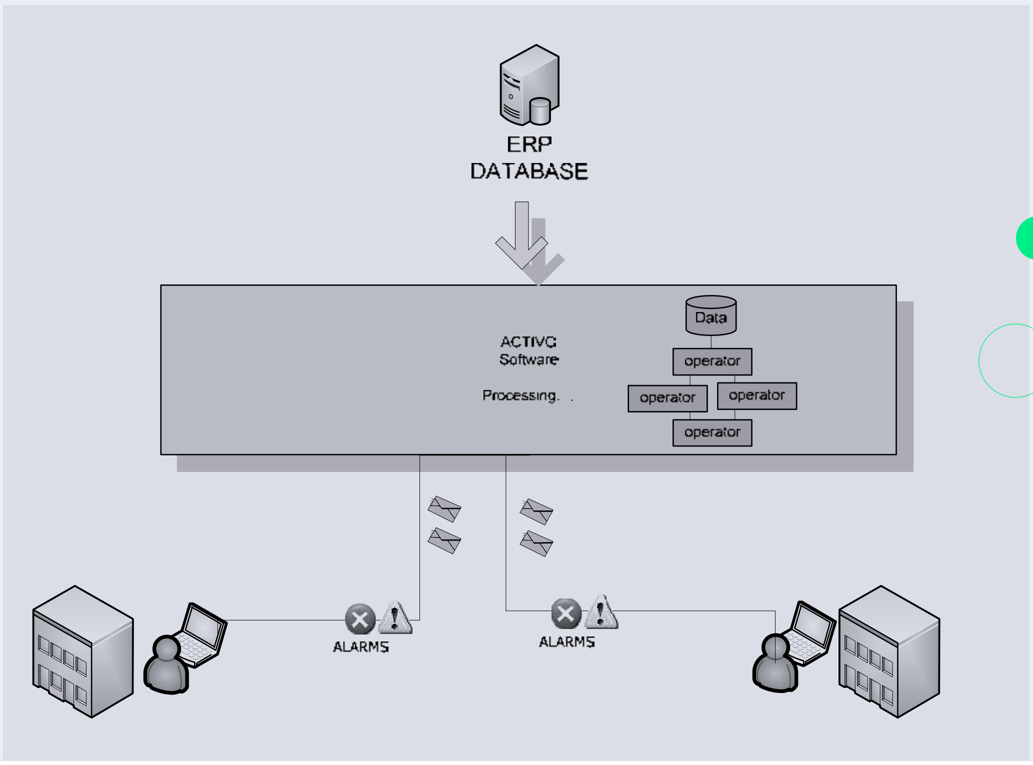
Furthermore, WEB SERVICE function was configured, which gives access to memory data file.

Thanks to this, each external software has access to the information that resides in the memory.

Impulse to trigger the processing: scheduled calls scanning.

Warning: if the user wants to have the memory data file (accesible through WEB SERVICE function) always up-to-date, he can set a database refresh interval (so, calling the hereby project) for a very short period of time

9.2 EXTERNAL ALERTS



CONTEXT

An enterprise using ERP class software wants to create an Contractor automatic notification system. The notification system is to be automatic and user independent. The notification system is to detect the alerts automatically and notify the proper recipients.

TASK

Running Contractor's automatic notification system in case of detecting unpaid and overdue invoices.

GRAVITY APPLICATION EXAMPLE

In case of identifying a condition (overdue contractor's invoices), GRAVITY sends e-mail notifications to recipients.

Impulse to trigger the processing: scheduled calls scanning.

The Diagram shows the project modeled in GRAVITY:

TASK

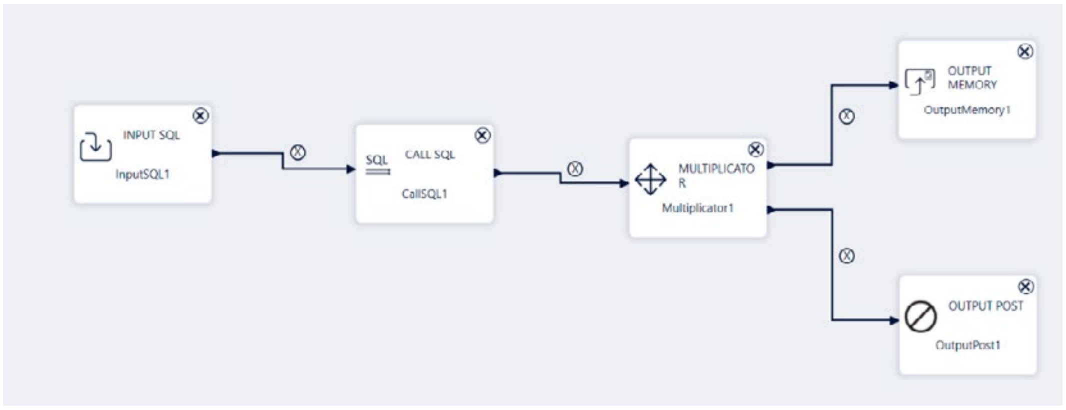
Running a Contractor's automatic notification on current balances every quarter

GRAVITY APPLICATION EXAMPLE

Balance calculation process for each contractor, report sending and setting a marker in source database (CALL SQL operator usage)

Impulse to trigger the processing: scheduled calls scanning.

Example diagram.



TASK

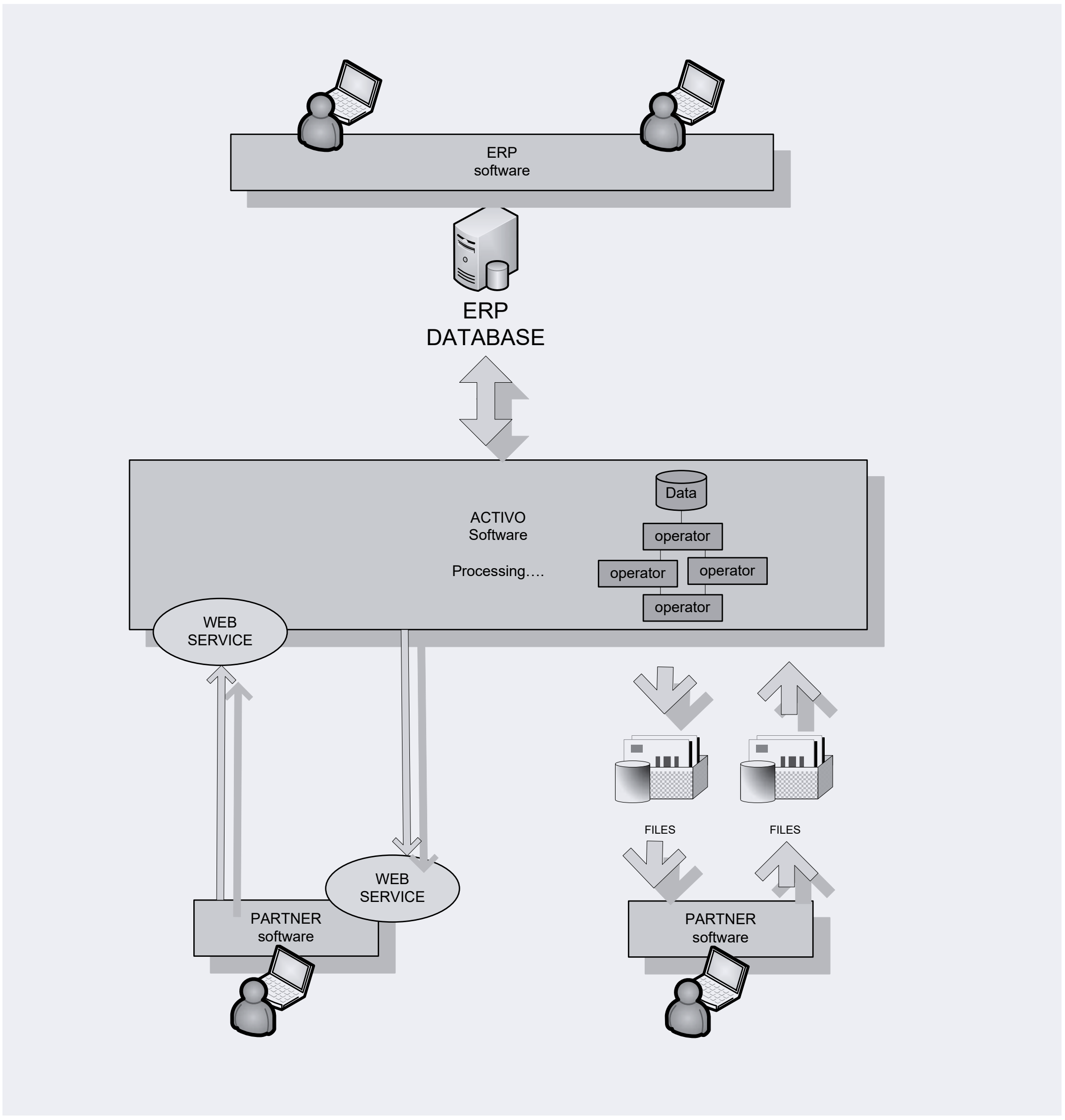
Running Contractor’s automatic notification in case of expired delivery detection.

GRAVITY APPLICATION EXAMPLE

In case of indentifying the condition (existing open order expiry), GRAVITY sends e-mail with the information to the supplier.

Impulse to trigger the processing: scheduled calls scanning.

9.3 INTEGRATION WITH CUSTOMERS' IT SYSTEMS



CONTEXT

An enterprise cooperates with customers, that have open IT systems.

TASK

Create a functionality, that will help the user to integrate product ordering process with the IT systems of selected customers.

GRAVITY APPLICATION EXAMPLE

Automatic customer's order download process. Let's assume, that the IT system of the customer places a file with order in the specified location, specific for each contractor. The locations are constantly scanned by GRAVITY. In case of order file detection, it is being added by GRAVITY to the ERP system, using WEB SERVICE function of the ERP system (CALL WEB SERVICE operator). The example above does only make sense if ERP has a WEB SERVICE function.

Downloaded order file is saved in different location (downloaded orders)

GRAVITY APPLICATION EXAMPLE

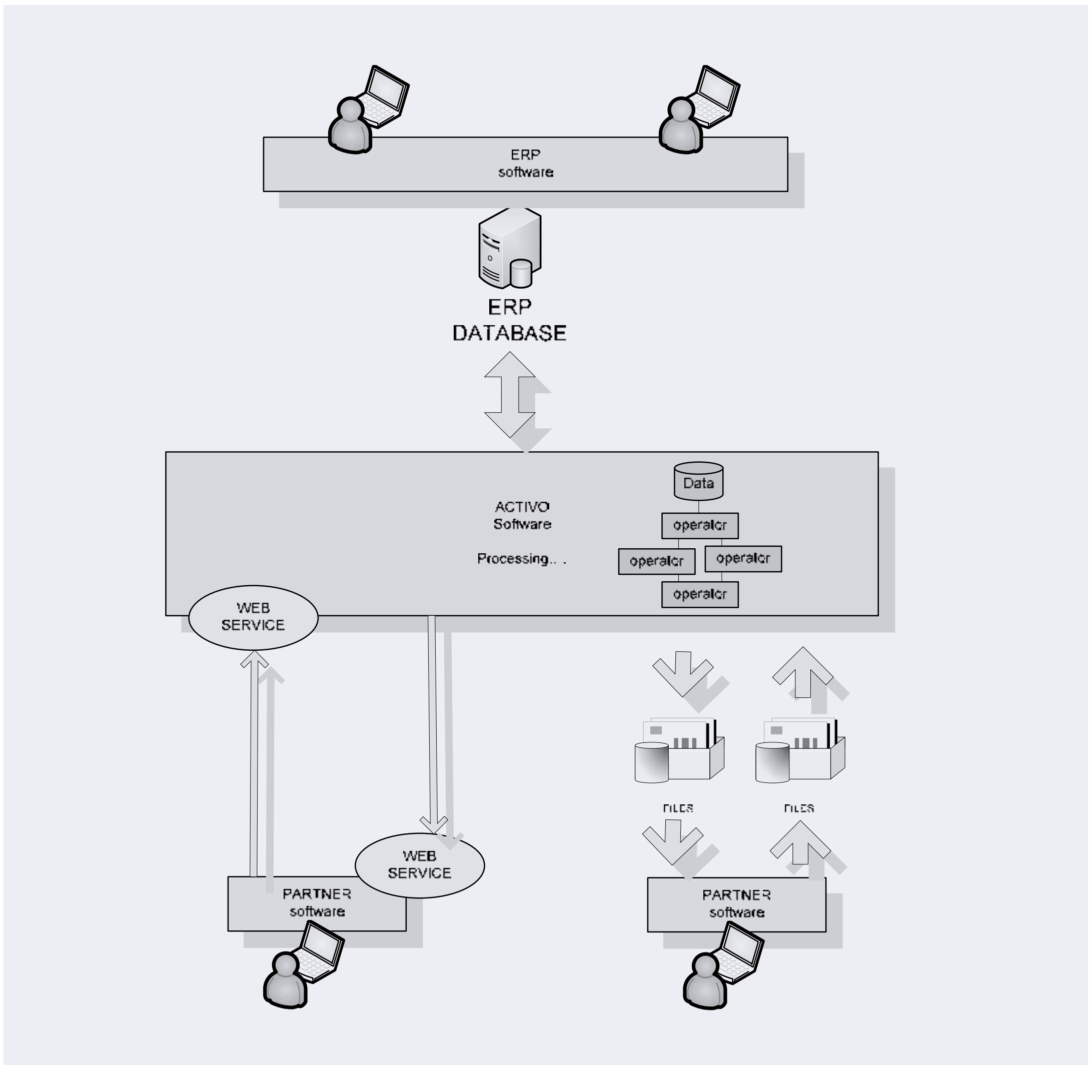
Automatic information to the recipient on automatic or manual order registration. Action based on database notification call. The contractor receives an e-mail with information on order processing start (OUTPUT POST operator).

GRAVITY APPLICATION EXAMPLE

Automatic information to the recipient on change of order status (for example: sent for execution). Action based on database notification call. The contractor receives an e-mail with information on order processing start (OUTPUT POST operator). Additionally, a file with customer's order positions is sent.

Note: identical actions can be implemented for any number of customer's order statuses.

9.4 INTEGRATION WITH SUPPLIERS' IT SYSTEMS



CONTEXT

An Enterprise cooperates with suppliers, that have open IT systems.

TASK

Create a functionality, that will help the user to integrate product ordering process with the IT systems of selected suppliers.

GRAVITY APPLICATION EXAMPLE

Automatic order sending process. Action based on database notification call (detection of order confirmation). Supplier receives an e-mail with order (OUTPUT POST operator).

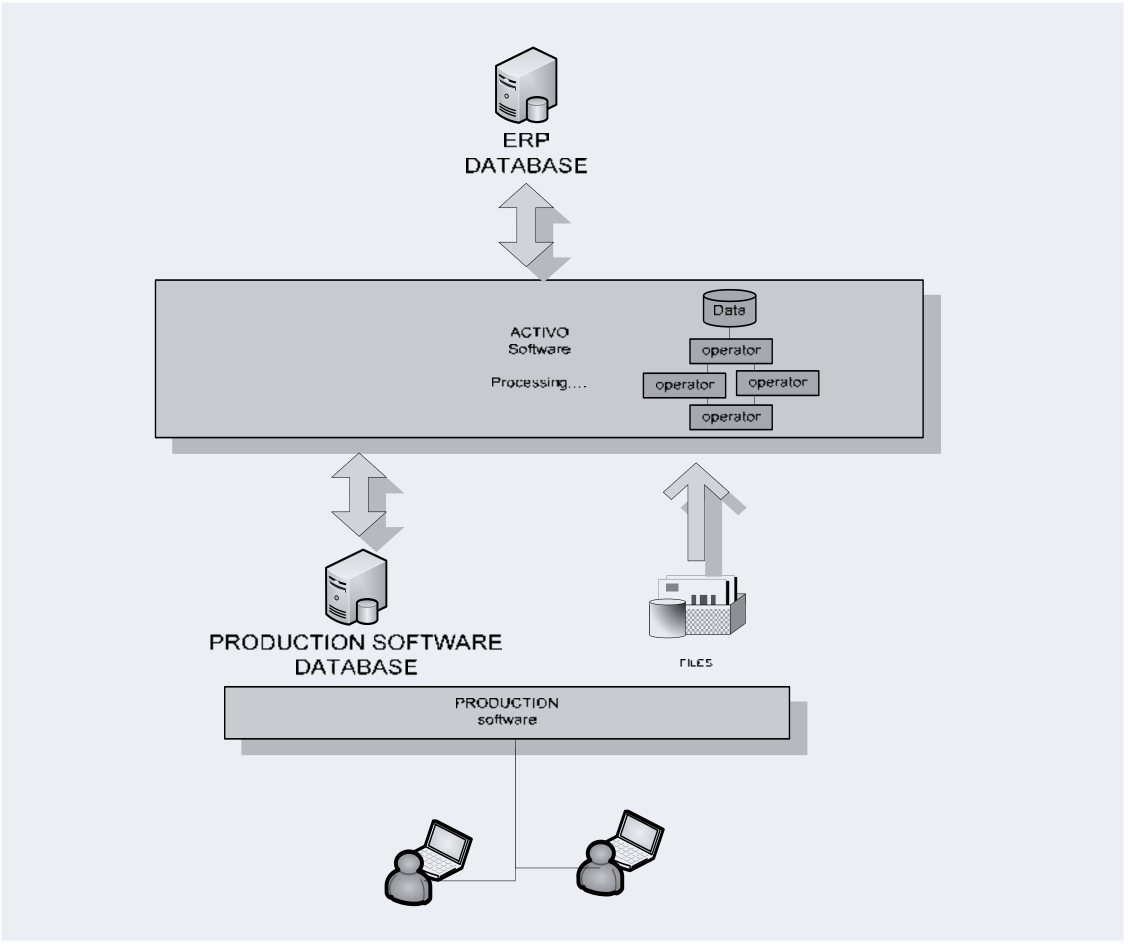
GRAVITY APPLICATION EXAMPLE

Automatic order confirmation download.

Let's assume, that supplier's IT system saves order file in a specific location. Locations are constantly scanned by GRAVITY. In a moment an order file is detected, it is added by GRAVITY into ERP system, as a warehouse order document, using ERP system's WEB SERVICE function (which is – using CALL WEB SERVICE operator – assuming that ERP has such function)

Dowloaded file is saved in other location (downloaded delivery orders)

9.5 INTEGRATION WITH MANUFACTURING SUPPORT SYSTEMS



CONTEXT

An enterprise uses ERP system and an independent, dedicated manufacturing support systems.

TASK

The enterprise wants to create a direct layer, integrating manufacturing support systems with ERP software. Thanks to an intermediate later, the enterprise becomes independent from integrated systems developers and has much bigger capability of integrated systems exchange.

GRAVITY APPLICATION EXAMPLE

One-Time data migration: downloading of technologies gathered in Excel files. Thanks to migration project creation, the action can be executed multiple times, for example for use of manufacturing support IT system beta tests.

GRAVITY APPLICATION EXAMPLE

Integration of ERP system, with a dedicated domain software (for example CAD type)

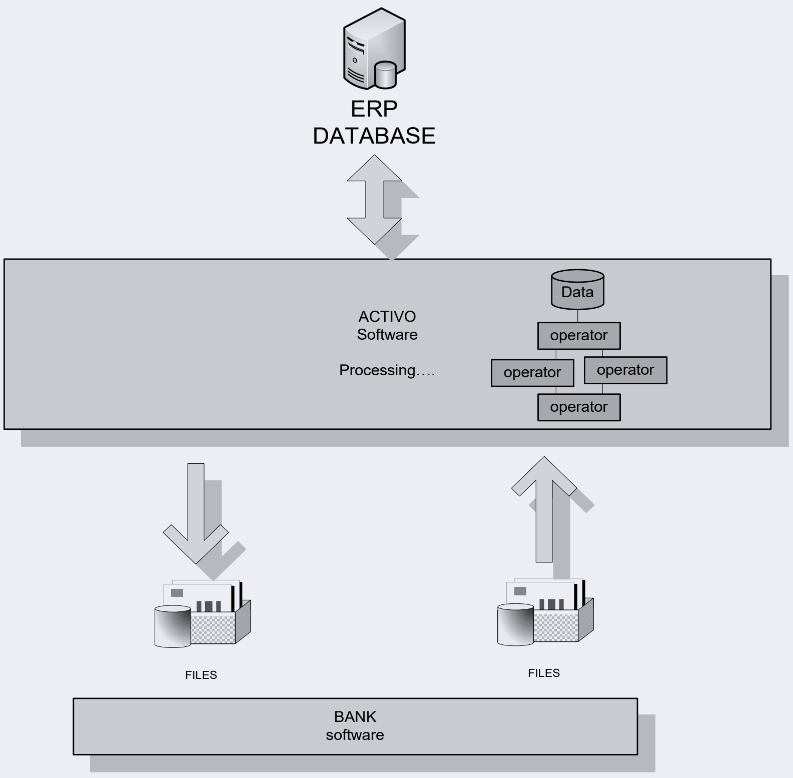
Technologies and material recipes transfers on user's demand.

GRAVITY APPLICATION EXAMPLE

Production progress follow-up. Integration of ERP system with dedicated service software (operation, or detail status card etc.)

9.6 INTEGRATION WITH BANK SYSTEMS

CONTEXT



An enterprise uses ERP software and targets bank records service automatization.

TASK

Create a functionality, that will help integrate ERP system with the bank.

GRAVITY APPLICATION EXAMPLE

Automatic bank statements download. Action based on scheduled calls. GRAVITY scans declared locations, specific for each bank.

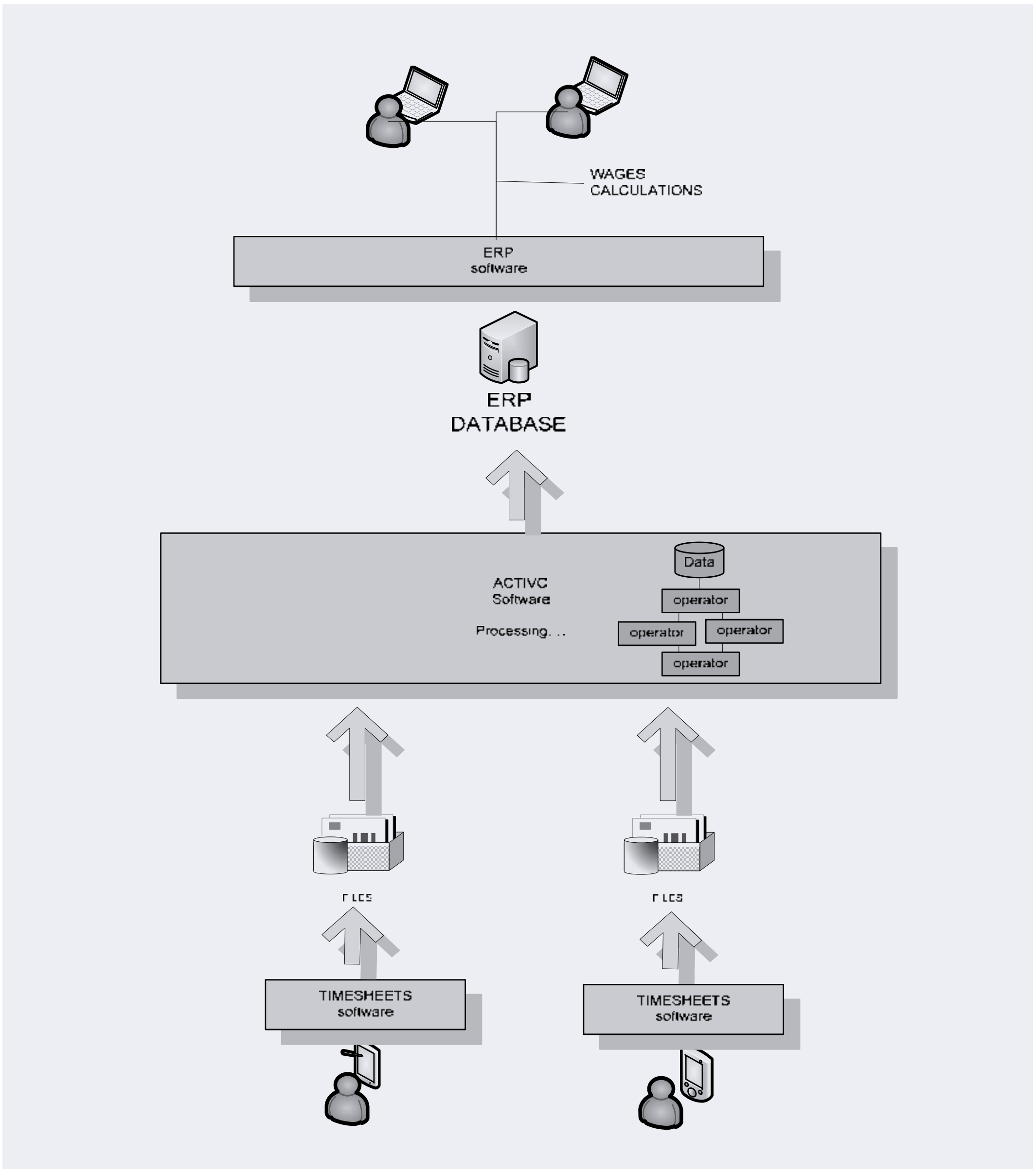
In case of detecting a bank statement file, items on the statement are added by GRAVITY to ERP system, using ERP system's WEB SERVICE function. The downloaded file is saved in other location (downloaded bank statements).

9.7 INTEGRATION OF TIME-SHEETS WITH HR

AND PAYROLL SOFTWARE

CONTEXT

An enterprise uses ERP software and time-sheets system from different developers.



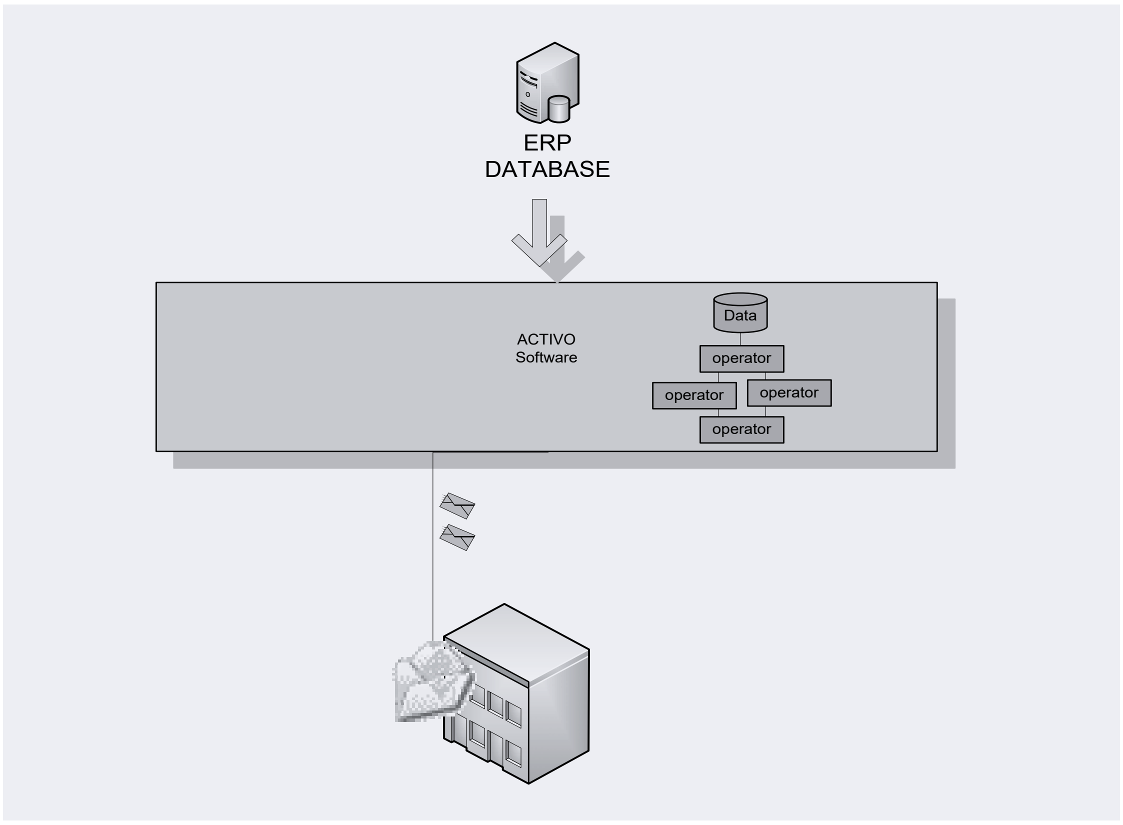
TASK

The entrprise wants to create an independent, intermediate layer, integrating time-sheets system with ERP software.

GRAVITY APPLICATION EXAMPLE

Integration of ERP system with dedicated domain software. Automatic scheduler was configured to scan selected location, in search of employees time-sheet files. CALL SQL operator call to place an information inside HR&payroll system (an alternative could be WEB SERVICE function call, provided ERP system has such option).

9.8 GENERATING FILES FOR PUBLIC SERVICE OFFICES



CONTEXT

An enterprise uses ERP software and targets automatic communication with Public Service Offices.

TASK

Computerization of communication between the enterprise and public service offices, creates need of fast reaction, often without ERP developer's support.

Example: Tax Office introduces obligatory XML format file supply in a one month time interval.

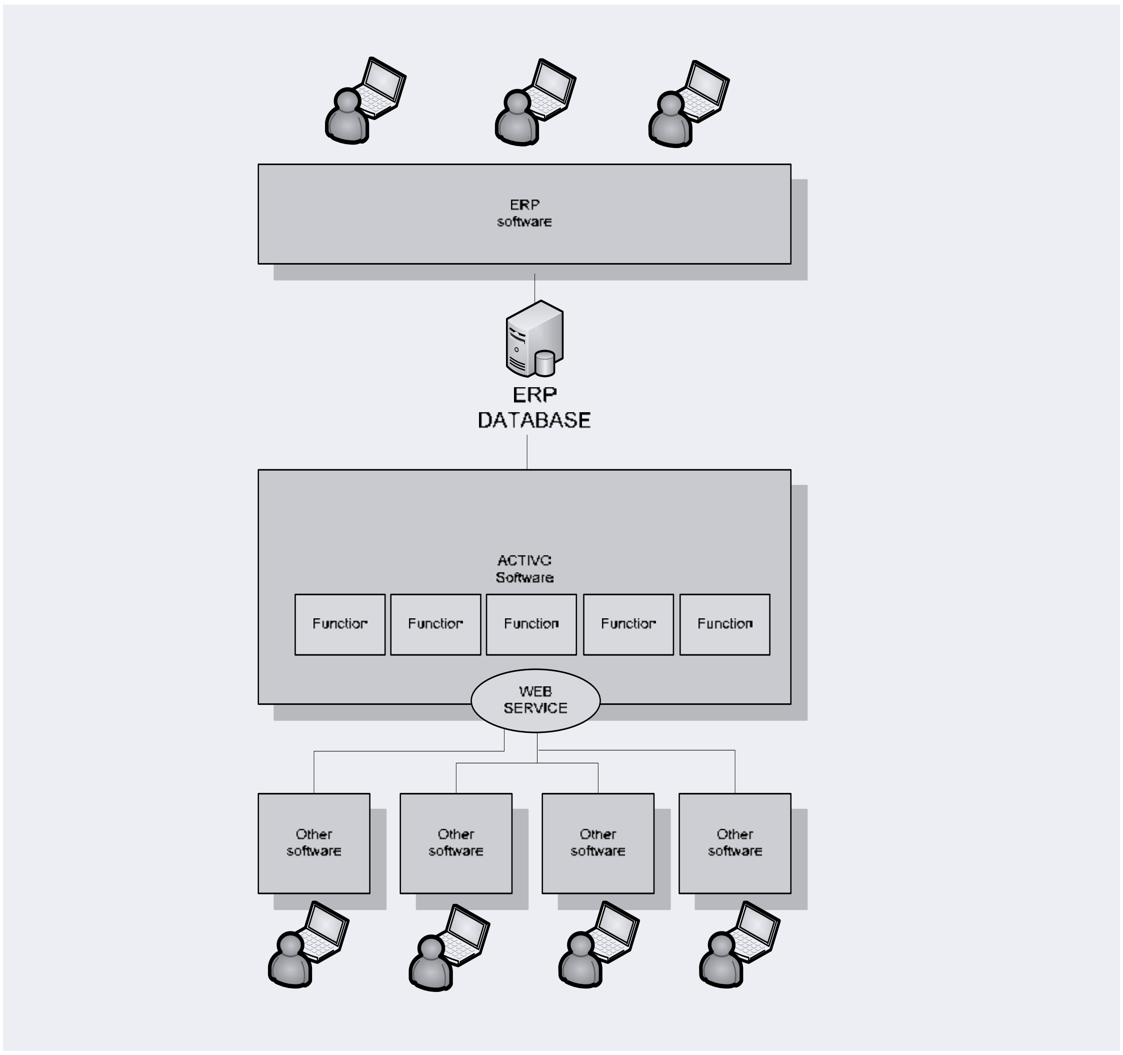
GRAVITY APPLICATION EXAMPLE

The process starts with downloading the data from database, ERP system data (INPUT SQL) and processing the data. Output was modeled as OUTPUT POST (XML output file format declared), which means that XML file will be sent to a selected recipient.

The example process is executed as WEB SERVICE function, which can be called for example from ERP system.

.

9.9 ERP SYSTEM DEVELOPMENT



CONTEXT

An enterprise uses an ERP software, without WEB SERVICE function. The software is "deaf", and integration can only take place on SQL database level.

TASK

We challenge ourselves to "open" the software, by creating a WEB SERVICE function, available for any external IT systems.

GRAVITY APPLICATION EXAMPLE

Step one"

Tworzymy funkcje, do których mogą odwoływać się systemy zewnętrzne (np. zapisz zamówienie, pobierz kartotekę indeksów, pobierz kredyty kupieckie dla kontrahentów etc.)

We create functions, available to be called by external IT systems (for example: save order, download index, download trade credits etc.)

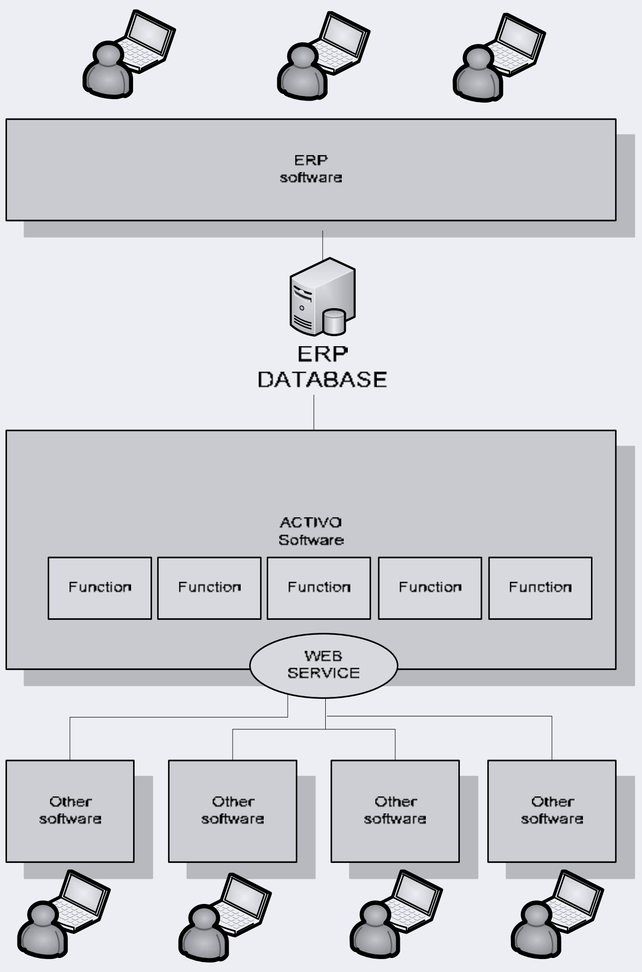
Step two:

We run WEB SERVICE server in GRAVITY

Action

Through this server, when calling WEB SERVICE function with a proper parameter, ERP software can call processing defined in GRAVITY (in terms of range defined in step one).

9.10 REPORTING



CONTEXT

An enterprise uses software, that does not have many useful reporting tools.

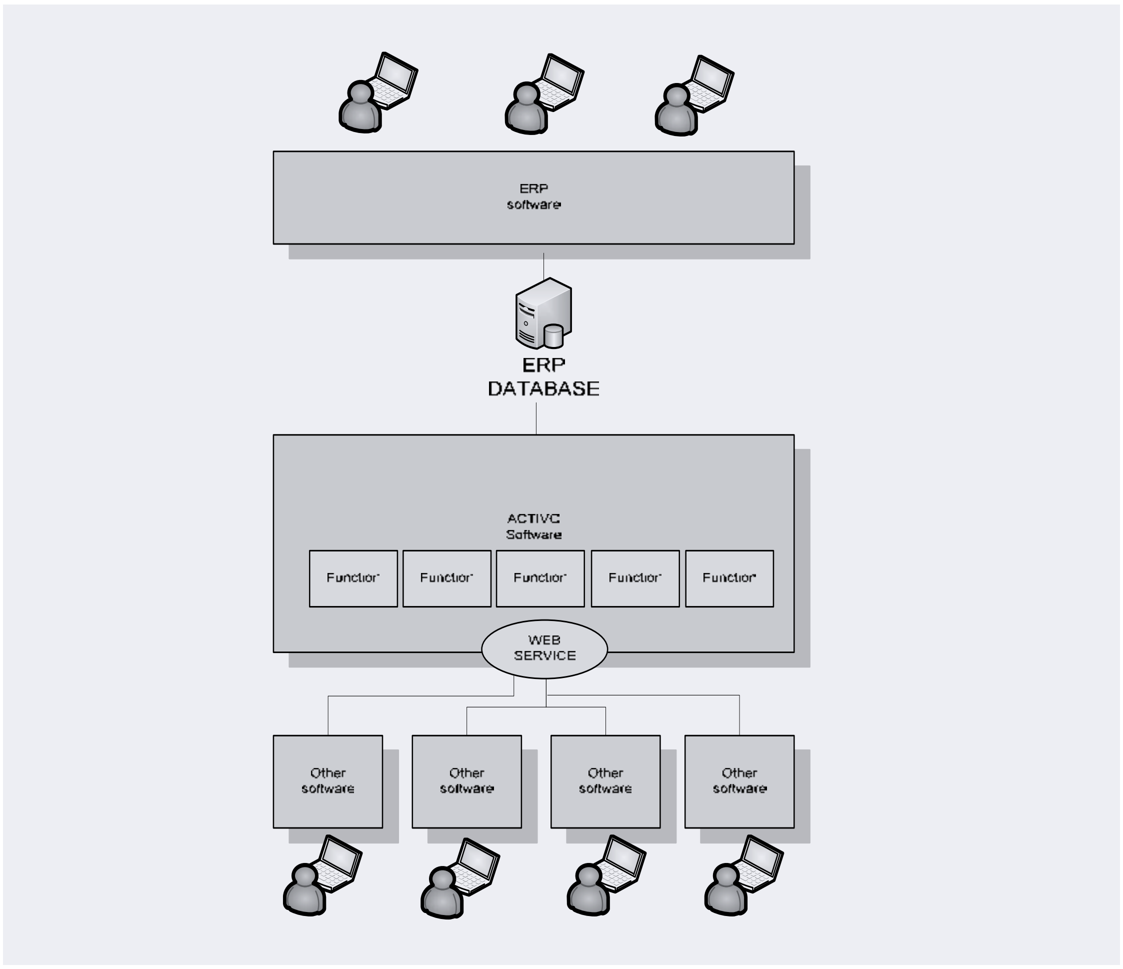
TASK

Generating reports with graphic layout and range according to needs.

GRAVITY APPLICATION EXAMPLE

In GRAVITY we can create reports, using data streams downloaded by the process. We can create any reports in .pdf formats, generated automatically, or in a time period, or as a reaction to an event, or call through external software as a WEB SERVICE function.

9.11 INTEGRATION OF ONLINE APPS WITH ERP SYSTEM



CONTEXT

An enterprise uses ERP software without WEB SERVICE function. The enterprise wants to use online solutions from other IT developer.

TASK

Let's make a task, to "open" the software by creating WEB SERVICE function, available for any external IT system.

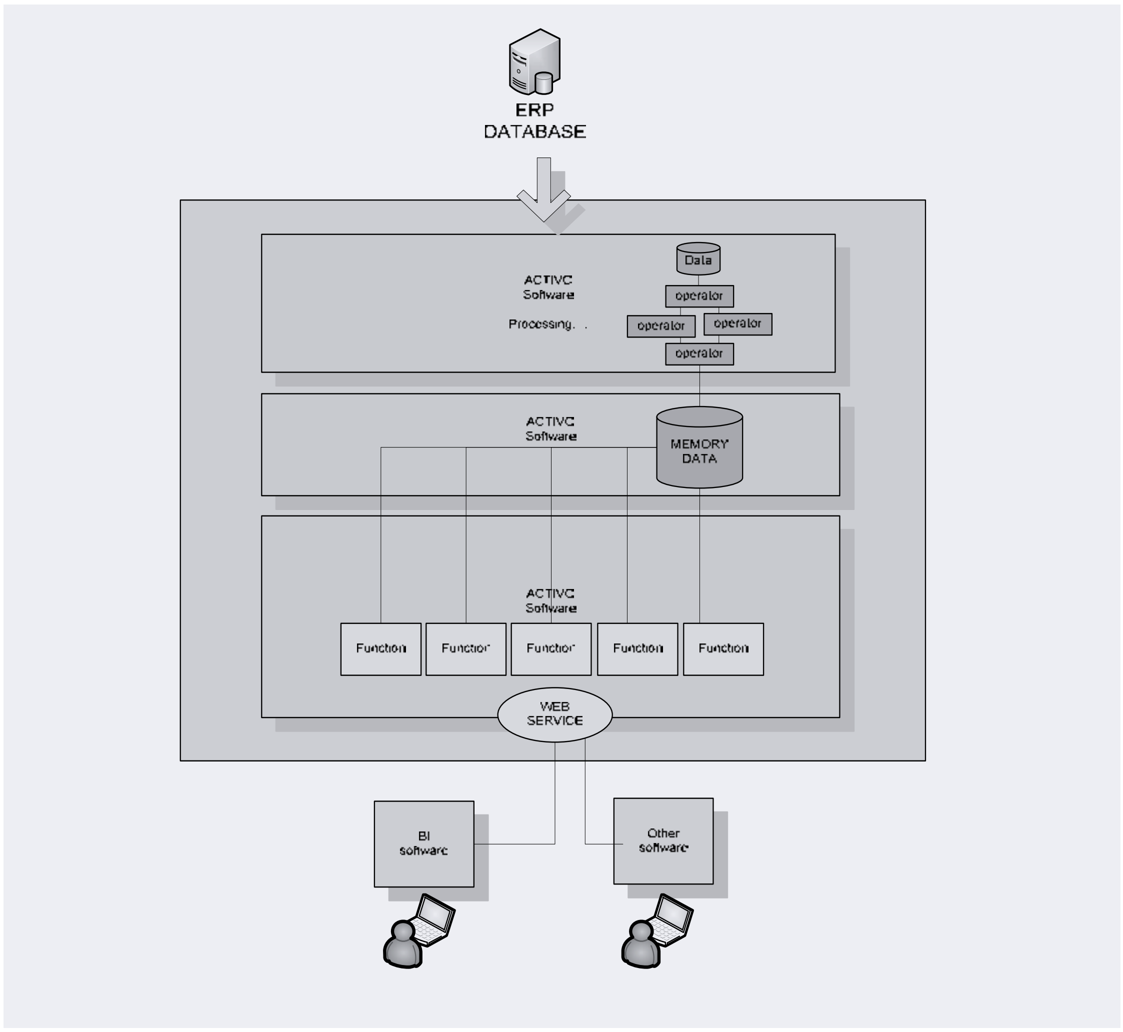
By creating an independent communication layer in GRAVITY, we create an IT system, in which the front applications can be exchanged without time-consuming integration.

The task is in fact an example generally described in the point above (ERP development).

GRAVITY APPLICATION EXAMPLE

The task is in fact a specific example of the "ERP development" point. We wanted to mark this use as an independent example, due to the fact it is very common need for such model of communication.

9.12 PROCESSING ACCELERATION AND FAST REPORTS CREATION



CONTEXT

An enterprise uses ERP software which has reporting system. Unfortunatelly, due to large number of data and processing complexity – the reporting takes too much time, which disables the user from effective usage of the tools..

TASK

Acceleration of data processing. Aggregated data for selected reports and analysis are available in cache memory.

GRAVITY APPLICATION EXAMPLE

Step 1

Create a project, whose task is to keep the in server memory fresh processed calculations and aggregations. (profit: fast access to processed data)

The project can use INPUT SQL operator to download the data and COMPUTING operator for data processing. Processed, aggregated data are located in temporary database, available for other projects (OUTPUT MEMORY operator)

The project is called by Scheduler, with short interval setup.

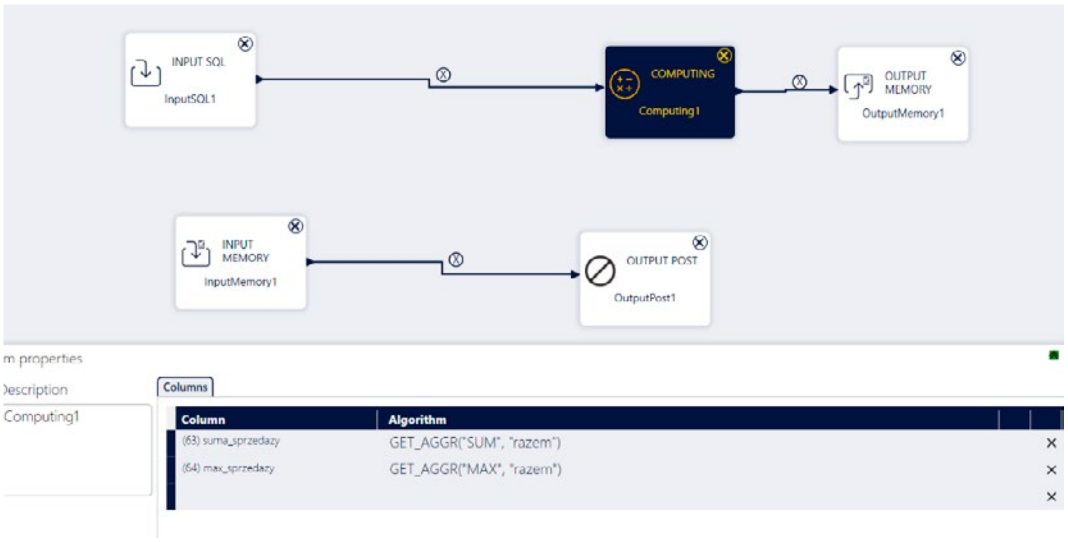
Step 2

Create a project, whose job is to send data for presentation.

INPUT MEMORY operator is used for fast access to data and OUTPUT DATA operator (output data for WEB SERVICE function) were used in the project.

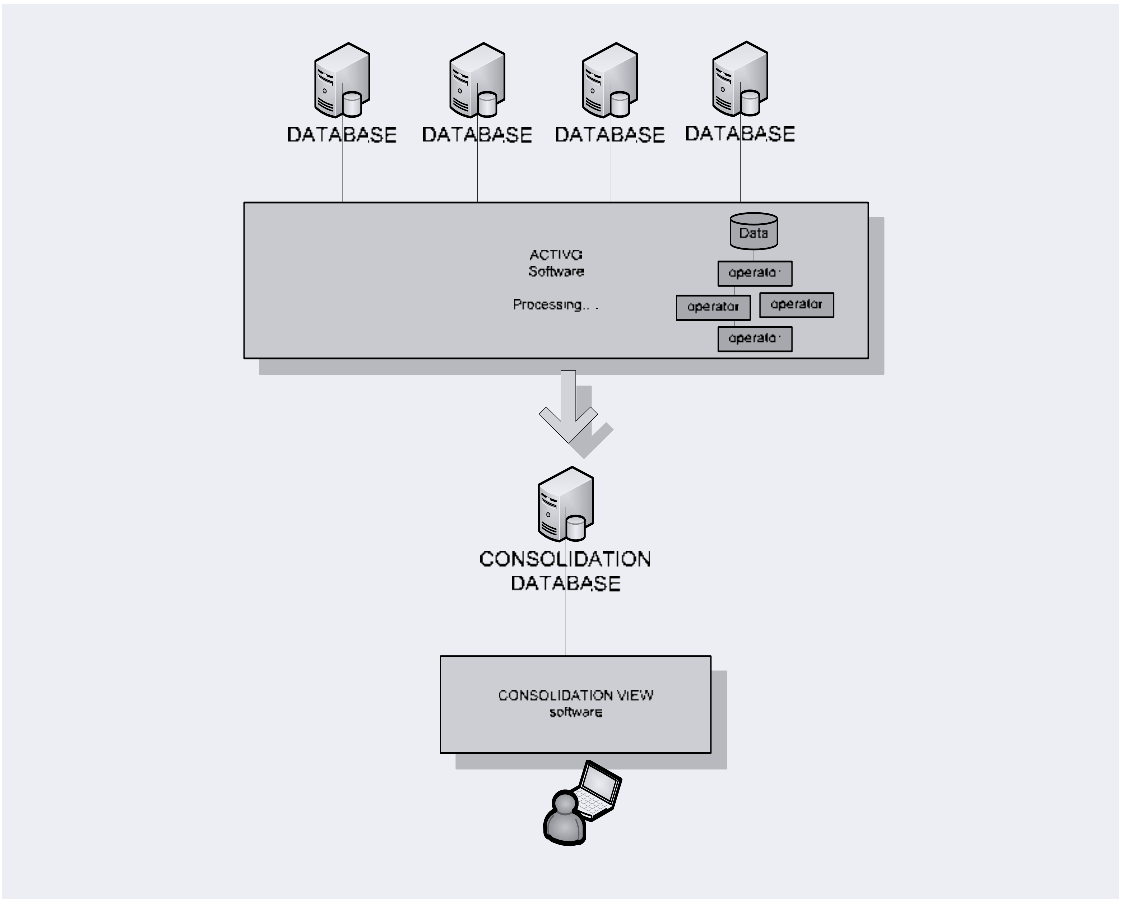
Project is called ast WEB SERVICE function fro ERP software.

Example diagram



Column	Algorithm
(63) suma_sprzedazy	GET_AGGR("SUM", "razem")
(64) max_sprzedazy	GET_AGGR("MAX", "razem")

9.13 DATA CONSOLIDATION



CONTEXT

An enterprise has a lot of subsidiaries, using different (or the same) software.

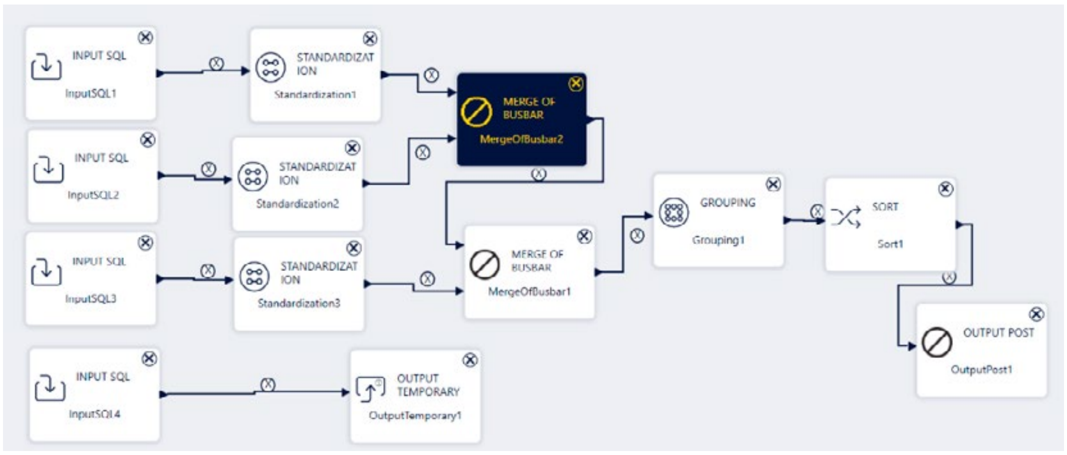
TASK

The enterprise wants to consolidate data on contractors' turnovers.

GRAVITY APPLICATION EXAMPLE

In GRAVITY we create INPUT SQL operators for each consolidated database. Thanks to STADARDIZATION operator, we identify the same contractors in multiple databases, and we sum the turnover. The result of calculation is trasferred to output database, which is connected to analytic software, or the result is send via e-mail to selected recipients.

Example Diagram



TASK

The enterprise wants to gather all sales invoices, of all subsidiaries, in one consolidated database. (this task is an expansion of the previous task).

GRAVITY APPLICATION EXAMPLE

In GRAVITY we develop INPUT SQL operators to all consolidated databases. Thanks to STANDARDIZATION operator, we identify the same contractors in multiple databases, and we add aup the turnovers. The result is transferred to an output database, which is connected to analytic software.

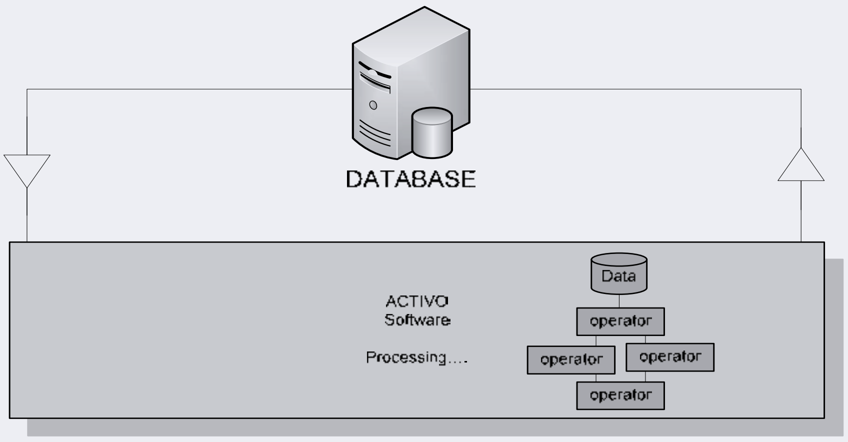
TASK

The enterprise wants to have consolidated economy sheets for all enterprises within the group

GRAVITY APPLICATION EXAMPLE

In GRAVITY we develop INPUT SQL operators to all consolidated databases. Thanks to STANDARDIZATION operator, we identify bookkeeping accounts structures in multiple databases. The result is transferred to an output database, which is connected to data presentation software (as an alternative: the xls file with results is sent to an e-mail address).

9.14 ERP SYSTEM DATA PROCESSING



CONTEXT

An enterprise is using ERP software with SQL database. During standard system operation, multiple alert situation appear, that are borh related and non-related to ERP software usage.The influence on the quality and integrity of stored data.

TASK

The job of IT services is to create the procedure of data examination and validation, to minimize the risk of corrupted data.

GRAVITY APPLICATION EXAMPLE

In GRAVITY we design the project of record testing within the selected tables, controlling, whether they contain forbidden values. If they are detected, depending on the table and type, the corrupted record is deleted or forwarded with information about corrupted data.

CONTEXT

An enterprise uses ERP software with SQL database. During ERP system usage, unintentional duplicates were reported (for example: on contractors list).

TASK

Database testing and elimination of potential duplicates.

GRAVITY APPLICATION EXAMPLE

In GRAVITY we develop a process of record testing, within the table of contractors data, examining if there are any duplicates, that is records containing similar cells (TEXT RECON operator). In case such cases are detected, we place a parent record’s reference inside the table (provided ERP system allows it.). IF ERP software does not allow it, information on results can be placed in a file and sent or placed in a declared database (access through external software).

CONTEXT

The enterprise uses ERP software. User would like to expand integrity testing by adding two rules, which he could define in any way.

TASK

Accounting rules control, in terms of booking of turnover on particular accounts.

GRAVITY APPLICATION EXAMPLE

In GRAVITY we model a process of testing of the rule, claiming that sum of turnovers of team 5, must be covered with control sum, booked in Account 490 turnover.

CONTEXT

An enterprise uses ERP software and analytic software (for example: BI class).

TASK

Preparation of ERP system data, to be rapidly used with analytic BI class software.

GRAVITY APPLICATION EXAMPLE

In GRAVITY we design a process of data download and aggregating sums calculations. Aggregating sums and potential duplicats are kept in cache memory, providing BI system the access to prepared data.

The process is called by a scheduled mechanism with short interval. Thanks to this the data in cache memory are always up to date.

GRAVITY APPLICATION EXAMPLE

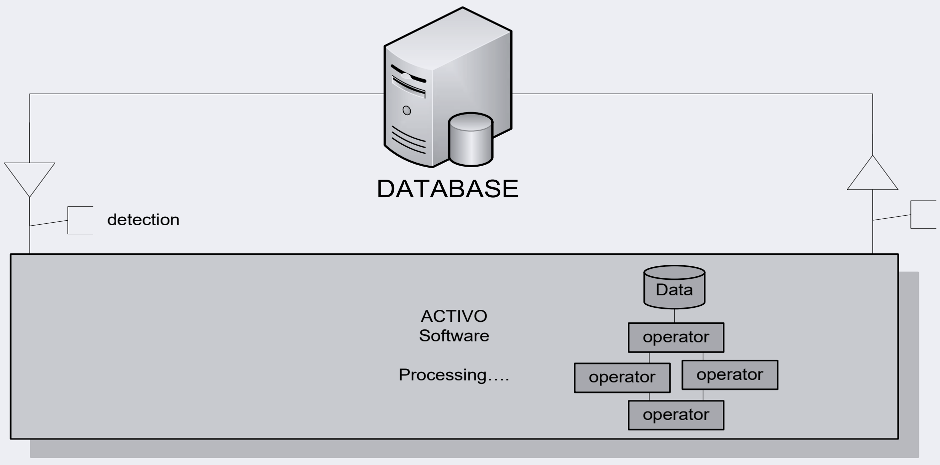
In GRAVITY we design a process of downloading data and normalization of units, to be used during summing and consolidating data.

GRAVITY APPLICATION EXAMPLE

We design a process of data download and bonding of multiple databases.

9.15 CRITICAL DATA CHANGES TRACKING

CONTEXT



An enterprise is using ERP class software, which does not have an option to follow the changes happening in a critical type of event.

TASK

The enterprise wants to introduce a system of tracking of all the modifications for a selected event. (for example edit of internal expenditure document).

GRAVITY APPLICATION EXAMPLE

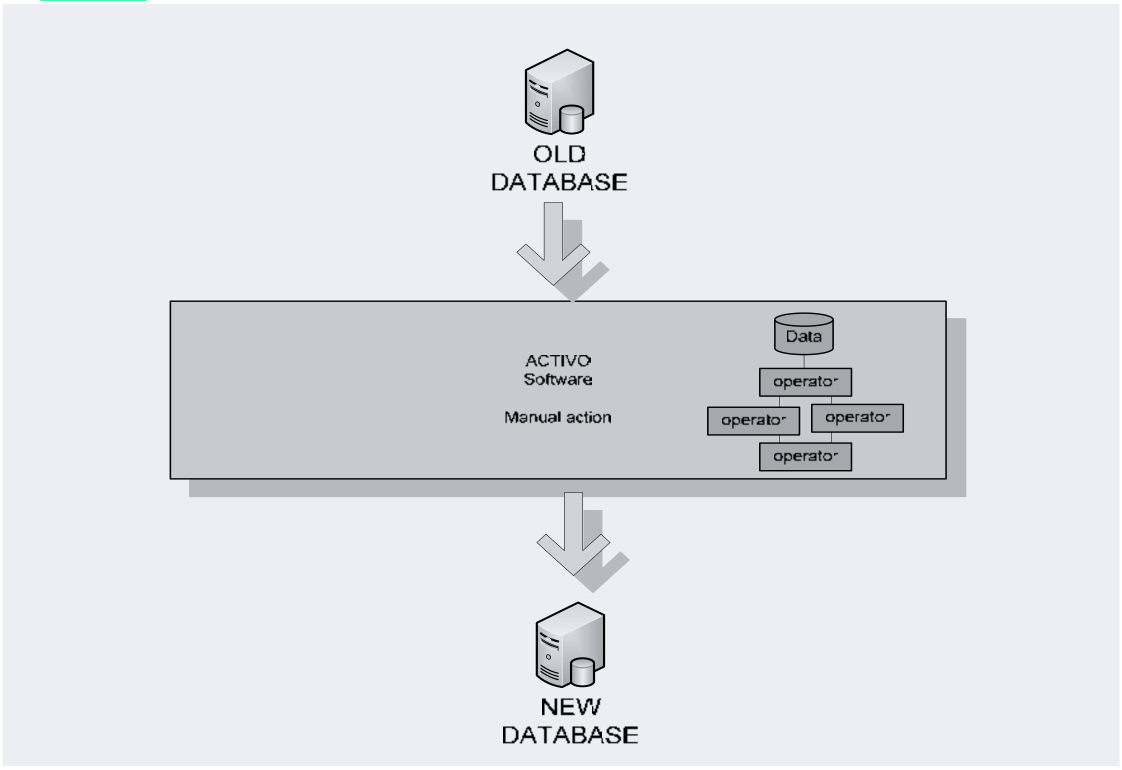
The project is called by database notification, in connection with the editing of the selected table (containing data of the considered event)

After being called – the project stores the data before and after editing (including timestamp) in other table.

9.16 CONVERSION OF EXTERNAL DATA –

MIGRATION OF DATA

CONTEXT



An enterprise is in the middle of new software implementation. Lots of data remains in the old database. The data needs to be transferred to a new system.

TASK

The enterprise has to create a migration project for production technology data. The project should be reusable for testing purposes and for productive start.

GRAVITY APPLICATION EXAMPLE

Project model includes material index, product index, material recipe and technological route downloading action.

We assume that the input data is gathered in an xls file, while basic output database of hypothetical ERP software is an SQL database.

Project is called manually, by the user.

TASK

The enterprise needs to create migration of HR & payroll data, which can be used multiple time for testing purposes and for productive start.

GRAVITY APPLICATION EXAMPLE

The model of the project includes downloading of employees index, contracts and payroll list.

Let’s assume, that input data is gathered in an SQL database of the current system, and the output data of hypothetical ERP software is an SQL database.

Project is called manually by the user.

9.17 OUTPUT SQL OPERATOR USAGE

OUTPUT SQL operator is the most complex component, and because of that, requiring special attention. Below we prepared few examples for you, which will allow better understanding of the usage

CONTEXT

The enterprise owns a lot of subsidiaries, using different software, or the same software, but keeping the data in different databases.

TASK

The enterprise wants to gather all data on sales in one database, to consolidate and make BI type analysis

GRAVITY tasks, formukated by us – fit into the examples described in this chapter, in the part marked as CONSOLIDATIONS

In the examples below, we will focus only on OUTPUT SQL operator usage methods.

OUTPUT SQL OPERATOR EXAMPLE USAGE

Source data interaction

In this example we will present a configuration with assumption of interactions with source data.

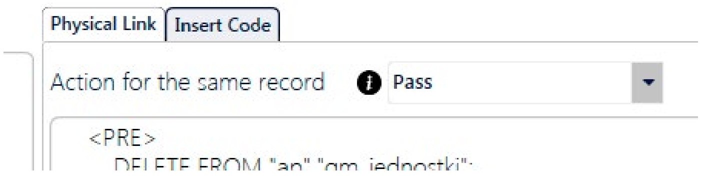
Standadization

Since we are consolidating data from different sources, unique identifiers have to be standardized during the processing. (changed so, to avoid duplicates)

Strumień danych wejściowych

Input data stream

During saving of each record, we can setup a flag of data download, in each source database. Thanks to this the INPUT SQL operators (creation of input data stream) can be build with use of the condition that limits the download of data stream only to those, which were not earlier downloaded.



Action for the same record: Pass

```
<PRE>
  DELETE FROM "an" "gm_jednostki";
```

The upper tab of the table, marked as “target”, must point STANDARDIZATION action (see picture below). After choosing STANDARDIZATION action, system asks for source ID (note, that the standardization dictionary can have multiple sources of data: you need to name your source ok select the bit of input data bus, which contains a unique name)

To set a source data flag – you must choose SETUP INPUT (see picture below) action for every each one of them on SQL ACTIONS area.

To finish INPUT SQL configuration, you need to select: (see picture below)

- INPUT - select INPUT type operator, where you download the bus data from
- INPUT TABLE - select source table from INPUT OPERATOR, which you want to edit.

- COLUMN FOR UPDATE - Select column from source table, which will be edited (flag setup location)
- VALUE FOR UPDATE - Select editable value (flag value)
- ID SOURCE RECORD - select record ID method: you select a unique column and a column from OUTPUT SQL operator input data bus.

Warning!

You can configure data stream flow for each source independently, which means, that you will have as many INPUT SQL operators and OUTPUT SQL opeators – as sources.

You can also complete this task in a different way: in this case process will be powered from as many INPUT SQL, as there re data sources, but the data are connected to one data bus (using for example MERGE OF BUSBAR operator). Next – only one OUTPUT SQL operator is configured. However in that case you must select as many SETUP INPUT type actions (ACTIONS SQL) in the OUTPUT SQL operator, as there are sources (for each source independent one action), together with adding action execution condition (there is a possibility of selecting a condition of implementation of SETUP INPUT action, depending on bit value of data stream)

Warning!

The example above is effective for objects (in our task those are invoices), which are due to change, but are not subject to be deleted in source (invoices can at most be corrected).

OUTPUT SQL OPERATOR EXAMPLE USAGE

Interaction with data source

The example below assumes no possibility of interaction with source databases.

Standardization

Since we are consolidating data from different sources – unique IDs must be standardized during saving (altered so, to avoid duplactes).

Since use of OUTUT SQL operator is connected with INPUT SQL operators, let us try to consider different methods of cooperation of those operators. Below we studied the effect, depending from configuration setup of source INPUT SQL operator, in range of downloading and configuration setup of OUTPUT SQL operator in terms od SQL action, concerning deleting of old data in target table

INPUT SQL OUTPUT SQL	Download whole range of data	Incremental Download	'tough position' incremental download'
Deleting all the records in target table SQL action * DELETE ALL RECORDS * DELETE ALL RECORDS FOR SOURCE	PROs: - Always full and clear information in target table CONs: - time consuming action	DENIED You cannot download incrementally and delete all information in target table, as you will not habe full information.	DENIED See remark to your left
Deleting the records in target table in constant declared range SQL actions * DELETE DECLARED SPACE	Identical as above The action is different from above as we assume the ID areas for each independent source, which causes that the standardizartion is not required. Case with no standardization!	DENIED See remark above	DENIED See remark above
Deleting records in target table in range resulting from input datastream SQL action * DELETE SPACE BY MIN MAX	Possible, but risky: Reason: there can be records from outside of datastream range, which were deleted in primary source (in this case – will not be deleted in target table)	Condition: input stream records once downloaded, can no longer be edited or deleted PROs: - processing speed	Condition: - input stream records with ID s below the firm position cannot be deleted or edited. PROs: - processing speed
Deleting of change of the redownloaded record SQL action * DELETE OLD RECORD * UPDATE OLD RECORD	Identical as above	Action effect identical as action above	DO NOT USE Action effect identical as above, but compromised by a situation, when we have deleted records in input stream in range of above the firm position, but downloaded earlier.

CREATED BY

CAFFEINE MINDS

FUTURE TECH FORMULA

We create ERP an BI
software since 20 years.

gravity.integration

gravity-integration.com