mapp Database: Connecting to a database

mapp Database establishes a connection to a database and can be used to retrieve, update and delete information and much more.

Tutorials about this topic are also available on our B&R online tutorial portal.

The contents of databases can be visually displayed in a mapp View HMI application using widget "Database".

1 Concept

1.1 Basic principle

mapp Database can be used to connect to a database. Large amounts of data can be stored efficiently in a database. For more information on databases, see here.

The following databases have already been successfully connected to mapp Database:

• MySQL (tested with V8.0)
• Microsoft SQL Server (tested with V2017)
  When using Microsoft SQL Server, the Microsoft SQL Server Native Client must also be installed. This can here be downloaded.
• MariaDB (tested with V10.0.34)
• PostgreSQL (tested with version 10.0)

It is also possible to use other databases. A suitable connector must be downloaded and the script modified accordingly. For more information, see section Installing the server.
With mapp Database, queries can be used in a database to create a table and add, deleted or modify entries. The user can create the queries using a simplified configuration interface or work directly with SQL queries. For more information on the subject, see section Query types in the MpDatabaseCore configuration.

Communication between the controller and the database takes place via an intermediate HTTP server. The controller sends requests to an HTTP server. The HTTP server is managed using a Python script. This script processes the data received from the controller and transfers it to the database as an SQL request. The database returns the corresponding data or executes the desired action. The advantage of this solution is that there are a wide range of different database connectors available for common operating systems for the PC where the script is running. To use a specific database, it is only necessary to install the appropriate connector and possibly modify the script.

For a step-by-step explanation of how to customize the script and establish communication, see section Connecting to the database.

1.2 Connecting to the database

mapp Database can be used to connect to a database. But before mapp Database can be implemented, a few steps must be completed. This section provides a step-by-step explanation of how to establish a connection to a database.

1.2.1 Downloading and configuring a database

To be able to work with mapp Database, an executable database system must be installed on the operating system. mapp Database has been tested in combination with the following databases:

- MySQL (tested with V8.0)
- Microsoft SQL Server (tested with V2017)
  When using Microsoft SQL Server, the Microsoft SQL Server Native Client must also be installed. This can be downloaded. [here](#)
- MariaDB (tested with V10.0.34)
- PostgreSQL (tested with version 10.0)

It is also possible to use other databases. A suitable connector must be downloaded and the script modified accordingly. For more information, see section Installing the server.

For information about how to configure the database, see the documentation for the respective database. In the next steps, the following information will be required by the database:

- IP address
- Port number
1.2.2 Installing the server

As explained in section Basic principle, communication between mapp Database and the database runs over an HTTP server. This is started by a script. The script is executed with Python. Python can be downloaded here.

For the communication between the HTTP server and database to work, a Python connector must be installed. For which one is used, see the documentation of the respective database.

Regardless of which database system is used, the MySQL connector must also be installed. The reason for this is that the data types in mapp Database are based on MySQL. Using the MySQL connector, data types from other database systems are correctly interpreted by mapp Database automatically.

For the MySQL connector, see here, for example. "Connector/Python" under "Connectors & APIs" must be installed.

Depending on the database version used, the Python connector version may change. This should always be checked when using a new database version!

Configuring and enabling the Python script

If Python has been installed, the Python script must be started. This script runs with Python 2.7. The script uses the installed MySQL connector to connect to a database.

The script can be downloaded here:

```
<table>
<thead>
<tr>
<th>Python script</th>
<th>mapp Services version</th>
</tr>
</thead>
<tbody>
<tr>
<td>mappDatabaseConnector</td>
<td>5.4</td>
</tr>
<tr>
<td>mappDatabaseConnector</td>
<td>5.5</td>
</tr>
<tr>
<td>mappDatabaseConnector</td>
<td>5.6 or later</td>
</tr>
</tbody>
</table>
```

The script is started using a command-line interface ("cmd" in Windows and "Terminal" in Linux Debian). The following must then be entered in the command-line interface:

- Path where the script is located
- Port number on which the script runs. Any free port can be specified. The port number must match the port number in the MpDatabaseCore configuration.
- IP address of the PC on which the database is running. In most cases, this will be the same PC on which the script is executed, so 127.0.0.1 can be used.
- Port number on which the database is running. The port number is specified when configuring the database.
- The last parameter determines the database type. The following abbreviations can be used: mysql: Used for MySQL database systems. mssql: Used for MS SQL database systems. postgres: Used for PostgresSQL database systems.

This could look like this in the command-line interface, for example:

```
C:\Users\admin\C:\Projects\Tutorial\mappServices\mappDatabase\mappDatabaseConnector.exe 86 127.0.0.1 85 mysql
```

In the example, the HTTP server uses port 86; the database uses port 85.

For example, if database system Microsoft SQL Server ("mssql") is used, this can look like this:

```
Path\\Script\./mappDatabaseConnector.exe 86 127.0.0.1 85 mssql
```

If you want to connect to another database system, the script can be modified as needed.

Since Python is an open-source program, it cannot be supported by B&R Support!

This command window can also be used, for example, to find out whether problems occurred while establishing the connection. For more information, see Diagnostics.
1.2.3 Implementing mapp Database

In the last step, mapp Database must be implemented. The IP address, port number, etc. of the database is defined in the `MpDatabaseCore` configuration. The connection is established using function block `MpDatabaseCore`. Information from the database can be stored in process variables using queries, which are also defined in the `MpDatabaseCore` configuration. The queries are performed using `MpDatabaseQuery`.

For an explanation of how to edit the configuration and how to implement the function block `MpDatabaseCore`, see Connecting to the database in the Getting started documentation.

How queries can be defined in the `MpDatabaseCore` configuration and executed with `MpDatabaseQuery` is explained in use case Query values from the database.

The contents of a database table can be visually displayed in a mapp View HMI application using widget "Database". For an explanation of how to implement the widget, see here.

1.3 Best practice

This section explains what to look out for when implementing mapp Database.

Limit the results of a query

A keyword can be used to define the maximum number of results for a query. This stops the search function of the database system after a certain number of values. To determine which keyword must be used, see the documentation for the respective database.

When using a MySQL database, "LIMIT" is used. In a MSSQL, "TOP" is specified instead of "LIMIT".

Example:

There are 30,000 entries in a MySQL database. A query applies to 10,000 entries, i.e. 10,000 entries would be returned to mapp Database. To limit the result of the query, keyword "LIMIT" can be used. Thus, the result can be limited to 100 entries, for example. This means that only the first 100 entries to which the query applies are sent to mapp Database.

In order to use keyword "LIMIT", the `MpDatabaseCore` configuration of query type "Custom" must be used:
The query could look something like this:

\[
\text{SELECT * from testtable WHERE id = 1 LIMIT 100}
\]

In MSSQL, "TOP" must be defined instead of "LIMIT":

\[
\text{SELECT TOP 100 * FROM testtable WHERE id = 1}
\]

2 Guides

2.1 Getting started

2.1.1 Connecting to the database

This section provides a step-by-step explanation of how to establish a connection to a database using MpData-baseCore.

In this example, a MySQL database system is installed with database "myData". All necessary steps to establish communication between mapp Database and the database have already been performed. They are also explained step-by-step in section Connecting to the database.
2.1.1 Creating a project

You must first create a new project in Automation Studio. For more information, see here.

2.1.1.2 Adding the mapp component

Adding the MpDatabaseCore configuration

TheMpDatabaseCore configuration is added in order to establish a connection to a database.

It contains the mapp Link, which uniquely identifies the mapp component within the mapp environment. The mapp Link establishes the connection between programming and configuration.

The database name and the required connection parameters have already been configured. This data can be taken from the MySQL database being used. It is important to ensure that the specified port number is not occupied by another program. For more information on this topic, see section Diagnostics.

2.1.1.3 Adding a program

The next step is to add a Ladder Diagram program. For more information, see here.

2.1.1.4 Adding MpDatabaseCore

Function block MpDatabaseCore is added in order to establish a connection to a database.

2.1.1.5 Configuring MpDatabaseCore

Connecting MpLink

Using input "MpLink", we establish a connection to the configuration we created earlier named "gDatabaseCore".
2.1.1.6 Activating the script

The script must be activated before a connection between MpDatabaseCore and the database can be established. It is responsible for ensuring that data can be exchanged between the database and mapp Database. The script is started using a command-line interface ("cmd" in Windows and "Terminal" in Linux Debian). For more information, see here.

If problems occur and additional information is required, see section Diagnostics.

2.1.1.7 Generating the file structure

Now generate a file structure for the memory card for ARsim. For more information, see here.

2.1.1.8 Testing the program

After the changes are downloaded, the program can be tested. Monitor mode is enabled. Function block MpDatabaseCore was already added to the Watch window.

Command "Connect" is used to establish a connection to the database.
Output parameter “Connected” indicates that the connection to the database has been established successfully. This can also be checked in the command window for the Python script (an entry appears with the current time). Afterwards, information can be requested from or added to the database using MpDatabaseQuery.

Use case Query values from the database is also available for this purpose.

2.2 Diagnostics

The following section contains answers to possible questions and issues that can occur when establishing a connection between mapp Database and the database.

A server must be active for communication between mapp Database and the database. A script is started for this. For more information about this topic, see here. The command window can also be used for diagnostics. It appears as soon as the script is started. From this command window, you can always get additional information such as shown here:

```
Error "SQL server error (-1064167424)" or "HTTP server error (-1064167423)" is shown on the function block. What should I do?

If error "SQL server error (-1064167424)" or "HTTP server error (-1064167423)" is shown, this can have different causes. In this case, the command window should always be checked for additional information. The errors may have the following causes:
- **Port already in use**: The port number of the database must be specified in the `MpDatabaseCore` configuration. This port number is also specified in the script. If the port is already being used by another program, the program must be terminated or a different port number must be used. The following message is displayed in the command window:

  ```
  Command Prompt
  File "C:\Python27\lib\SocketServer.py", line 417, in __init__
  self.server_bind()
  File "C:\Python27\lib\socket.py", line 431, in server_bind
  self.socket.bind(self.server_address)
  File "C:\Python27\lib\socket.py", line 228, in meth
  return getattr(self, sock_name)(args)
  socket.error: [Errno 10604] Only one usage of each socket address (protocol/network address/port) is normally permitted
  ```

- **Required connectors have not been installed**: Section Installing the server explains that a connector must be installed in order to start the server. Check if they have been installed. A message appears in the command window stating that the required connectors are missing.

- **Incorrect port or IP address in the script**: The script must be checked to determine if the correct port number and IP address for the database have been specified. The following message appears in the command window:

  ```
  C:\Python27\python.exe
  SQL server host 127.0.0.1:89
  Exception: code 2003, message 2003: Can't connect to MySQL server on '127.0.0.1:89' (10061)
  No connection could be made because the target machine actively refused it
  ```

- **Login failed**: It is necessary to check whether the correct user name and password for the database have been entered in the `MpDatabaseCore` configuration. If an incorrect specification is made, the following is displayed in the command window:

  ```
  C:\Python27\python.exe
  Starting httpd at port 86
  SQL server host 127.0.0.1:85
  Exception: code 1045, message 1045 (28000): Access denied for user 'root45'@'localhost' (using password: YES)
  ```

- **Update of the connectors**: If, for example, a new database version is used, the connector version may also need to be updated. This should be checked. Depending on the database being used, the command window will indicate that connector authentication has failed.

- **Microsoft SQL Server**: It is necessary to ensure that "SQL server authentication" is selected as the authentication type when starting Microsoft SQL Server:
Otherwise, it is not possible to log in as the users that are defined in the MpDatabaseCore configuration!

- **Firewall:** It is necessary to ensure that the firewall being used does not block the connection! Depending on how the firewall reacts, error "HTTP server error (-1064167423)" can be triggered in MpDatabaseCore.

- **Missing software:** If Microsoft SQL Server is used as the database system and displays the following message in the command console:

  "Exception: code IM002, message [IM002] [Microsoft][ODBC Driver Manager] Data source name not found and no default driver specified (0) (SQLDriverConnect)"

the Microsoft SQL Server Native Client must be installed! This can be downloaded here.

  Older Microsoft SQL Server Native Client versions, such as 2012, can be used with newer Microsoft SQL Server versions, such as 2017!

### My query cannot be executed

Queries can be executed using MpDatabaseQuery. If a function block error is shown when the query is started, the following should be checked:

- **Query failed:** The query defined in MpDatabaseCore configuration should be checked. Make sure that the desired database table has been specified correctly. The validity of user-defined queries (query type = "Custom") cannot be checked by mapp Database. A check must be carried out to ensure that the query has been specified correctly. For more information about specifying SQL queries, see here. If a query fails, the logger always shows additional information!

- **Microsoft SQL Server query:** If a query is executed on a Microsoft SQL Server, it is necessary to ensure that executing the query is permitted. By default, this is not permitted. This is done in the server settings (under "User mapping").

### Output parameter "CommandBusy" on MpDatabaseCore always remains TRUE.

If output parameter status remains "CommandBusy = TRUE" after the "Connect = TRUE" command on MpDatabaseCore, this may have the following causes:

- **Incorrect port specified:** If an incorrect port is specified in the script, it is possible that output parameter "CommandBusy" always remains TRUE depending on the database used.

- **Firewall:** It is necessary to ensure that the firewall being used does not block the connection! Depending on how the firewall reacts, error "HTTP server error (-1064167423)" can be triggered in MpDatabaseCore. It is possible that the firewall blocks the connection but does not alert the user. It is therefore possible that output parameter "CommandBusy" on MpDatabaseCore always remains TRUE.

### Messages in the command-line interface

Various messages appear in the command-line interface. The cause and solution of entries in the command-line interface are explained here:
Default value

If the following/similar message appears in the command-line interface:

```
Exception: code 1364, message 1364 (HY000): Field 'mapp' doesn't have a default value
```

Then no default value is defined for the specified column. The error occurs if a process variable was used in a query that does not cover all column names. For more information, see "Specifying variables" in section Query types.

A default value must be defined for the column. When creating a new table, it can look like this:

```
CREATE TABLE t1 (
    i     INT DEFAULT -1,
    c     VARCHAR(10) DEFAULT '',
    price DOUBLE(16,2) DEFAULT 0.00
);
```

For an explanation about how an existing table can be edited, see the documentation of the database system used.

Another reason could be that "NOT NULL" was specified for a column when the table was created. This means that no empty value can be specified for the column. This must be taken into consideration for the process variable used for the query!

Missing column

If the following/similar message appears in the command-line interface:

```
Exception: code 1054, message 1054 (42S22): Unknown column 'AP' in 'field list'
```

Then a process variable containing variables that do not match any column in the table was used in a query. For more information, see "Specifying variables" in section Query types. The PV being used must be adjusted so that the error message no longer occurs.

3 Configuration

3.1 MpDatabaseCore configuration

General

The following parameters can be configured in the general settings.

- **Database name**: The name of the desired database is specified under "Database name".
- **Username**: Specified username to connect to the database.
- **Password**: Specified password to connect to the database.
- **IP address**: Defines the IP address to connect to the database.
- **Port number**: Defines the port number to connect to the database. The port number of the HTTP server used for communication is specified. For more information, see here.
- **DBMS**: The database system is selected under "DBMS". "Open" can be used. This means that a connection can be established to any database system. For more information, see section Basic principle.

Database settings

Some databases support 8 byte integers. They are not supported by Automation Studio however. For this reason, any 8 byte integers used can be converted to UDINT, DINT or LREAL.

"No conversion" can be selected if a conversion should not take place. In this case, the user must ensure that the values are converted correctly. A "Cast" must be performed here. For more information, see here.

Automatic mapping

Under "Automatic mapping", you can define how structure variables are mapped to the columns in the database table in Automation Studio. There are two options to choose from:
mapp Database: Connecting to a database

- **Strict**: With "Strict", the order and name of the variables within the structure must match the name and order of the columns within the table. The structure is not permitted to contain any additional variables. If the structure does not correspond exactly to the table, an error is displayed on MpDatabaseQuery. Depending on the database used, additional information can be found in the command-line interface. For more information, see section Diagnostics.

- **Flexible**: With "Flexible", the order of the variables within the structure can vary. Only the name of the variable must match the column name in the table. If the number of the columns and variables do not match, or if not all columns are covered, the behavior depends on the database system used. For more information, see here in section "Specifying variables".

For an explanation about which query types the automatic mapping function can be used for, see section Query types.

**Query**

Queries can be created under "Queries". A unique name for the query must be specified under "Name". If the request is to be executed via MpDatabaseQuery, the name defined here must be specified on input parameter "Name".

The query type is determined using "Type". The following types are possible:

- SELECT
- INSERT
- UPDATE
- CREATE TABLE
- DELETE FROM
- Custom

For more information about the individual query types, see here.

### 3.1.1 Query types

For examples of different query types, see here.

**Specifying variables**

The variables used in the different query types can be specified in two different ways.

**Automatic**

Option "Automatic" depends on the setting made under "Automatic mapping":

```plaintext
For an explanation about which query types the automatic mapping function can be used for, see section Query types.

---

For examples of different query types, see here.

**Specifying variables**

The variables used in the different query types can be specified in two different ways.

**Automatic**

Option "Automatic" depends on the setting made under "Automatic mapping":

```
• **Strict:** With "Automatic" and "Strict", the variable connects to the columns of the table automatically. A variable must be specified that has the exact structure and variable names as in the database table. The table "product" in the picture below has the columns "id", "name" and "quality". Automation Studio creates a variable, such as "SetData", which has type "Product_Type":

If the variable names do not match the column names or if too few variables were specified, error "Error in the query" is triggered on the function block MpDatabaseQuery. Error "Incorrect PV entry" is also displayed in the Logger.

• **Flexible:** With "Automatic" and "Flexible", the order of the variables within the structure can vary. Only the name of the variable must match the column name in the table. If the number of columns ("SA") and variables ("VA") does not match or if not all columns are covered, the behavior depends on the database system used and the query type.

<table>
<thead>
<tr>
<th>Mapping case</th>
<th>SELECT</th>
<th>INSERT INTO</th>
</tr>
</thead>
<tbody>
<tr>
<td>VA and SA are identical. Column names are covered. Variable names and column names in the same order</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>VA and SA are different. Column names are covered. Variable names and column names in the same order</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>VA and SA are different. Column names are not covered. Variable names and column names in the same order</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>VA and SA are identical. Column names are covered. Variable names and column names not in the same order</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>VA and SA are different. Column names are covered. Variable names and column names not in the same order</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>VA and SA are different. Column names are not covered. Variable names and column names not in the same order</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
mapp Database: Connecting to a database

- ✔: Query can be executed.
- ❌: Query cannot be executed An error is triggered. For more information, see "Messages in the command-line interface" in section Diagnostics.
- !: Behavior depends on the setting of the database used. This can be found in the documentation of the respective database. Since not all column names are covered, either a default value defined in the database is inserted or an error is triggered. For more information, see "Messages in the command-line interface" in section Diagnostics.

Single columns

Under "Single columns", the desired columns are specified individually. The desired column name is defined under "Column". A PV must be specified in order for the information to be stored, deleted or updated, etc. The name of the PV and the column name in the database do not have to match!

Specifying PVs

If a variable with a simple data type is specified, this is done as follows: "::Taskname:PVName"

The following SQL, MSSQL or PostgreSQL data type should be used for the following IEC data types:

<table>
<thead>
<tr>
<th>IEC data type</th>
<th>MySQL/MariaDB data type</th>
<th>MSSQL data type</th>
<th>PostgreSQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>TINYINT(1)</td>
<td>BIT</td>
<td>BOOL</td>
</tr>
<tr>
<td>SINT</td>
<td>TINYINT(4)</td>
<td>TINYINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>INT</td>
<td>SMALLINT(6)</td>
<td>SMALLINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>DINT</td>
<td>INT(11)</td>
<td>INT, BIGINT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>USINT</td>
<td>TINYINT(3) (unsigned)</td>
<td>TINYINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>UINT</td>
<td>SMALLINT(5) (unsigned)</td>
<td>SMALLINT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>UDINT</td>
<td>INT(10) (unsigned)</td>
<td>INT, BIGINT</td>
<td>BIGINT</td>
</tr>
<tr>
<td>STRING</td>
<td>VARCHAR(81)</td>
<td>VARCHAR</td>
<td>CHARACTER VARYING</td>
</tr>
<tr>
<td>DATE_AND_TIME</td>
<td>DATETIME</td>
<td>DATETIME2</td>
<td>TIMESTAMP WITHOUT TIMEZONE</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
<td>TIME</td>
<td>Currently not usable</td>
</tr>
<tr>
<td>LREAL</td>
<td>DOUBLE</td>
<td>FLOAT, BIGINT</td>
<td>DOUBLE PRECISION</td>
</tr>
</tbody>
</table>

MSSQL data type BIGINT is an 8-byte data type. This cannot be processed automatically by Automation Studio. A conversion to LREAL, DINT or UDINT must be carried out. This can be done in the MpDatabaseCore configuration under the general settings ("Database setting").

Specifying array PVs

If an array PV is specified, this is done as follows: "::Taskname:MyArrayPV[]"

- The brackets [] must be added by the user.

Specifying PV snippets

In queries, it is also possible to specify PVs in the form of snippets. This is especially relevant in combination with "Custom" queries. They are specified in the configuration using "{&pv[Taskname:PVName]}". The snippet must always be specified in curly brackets ("{}"). The "&pv" stands for the value of a process variable. The actual PV is indicated in the square brackets ("[]").

For examples of how variables can be specified in queries, see section Query examples.

SELECT

"SELECT" can be used to copy the information from a table/column to a PV. The table from which information should be queried is defined under "FROM". The column from which the information should come is specified under "Columns". There are two query options here, "Automatic" and "Single columns". They are described above under "Specifying variables".

In addition, filter criteria can also be specified under "WHERE". The name of the desired column is specified using "Column". A value to be compared can be specified under "Value". "Operator" defines how the value should be compared with the column name.

If "LIKE" is used as an operator, similar entries can be searched for. Wildcard symbol "*" must be used. If "Test" is used as "Value", for example, entries "Test01", "MyTest" and "MyTest01" can be searched for.
In some databases, wildcard symbol "%" is also used.

Various filter options can be specified. These are connected to each other via "Connect".

**INSERT**

Information can be added to a database using "INSERT". The desired database table must be specified under "INTO". "Columns" determines how the information is added. For an explanation, see section "Specifying variables" above.

**UPDATE**

"UPDATE" can be used to update information in the table. The desired table name is specified under "Table name". You can then use "Columns" to determine how the information should be updated. There are two options here, "Automatic" and "Single columns". They are described above under "Specifying variables".

In addition, filter criteria can also be specified under "WHERE". The name of the desired column is specified using "Column". A value to be compared can be specified under "Value". "Operator" defines how the value should be compared with the column name.

If "LIKE" is used as an operator, similar entries can be searched for. Wildcard symbol "*" must be used. If "*Test*" is used as "Value", for example, entries "Test01", "MyTest" and "MyTest01" can be searched for.

In some databases, wildcard symbol "%" is also used.

Various filter options can be specified. These are connected to each other via "Connect".

**CREATE TABLE**

"CREATE TABLE" can be used to create a new table in the database. The name of the table is defined under "Table name".

"Columns" defines how the columns in the table should be created. There are two options here, "Automatic" and "Single columns". They are described above under "Specifying variables".

The data types of the specified structure/variable determine the data types in the database table!

**DELETE FROM**

"DELETE FROM" can be used to delete one or more columns in a table, which must be specified using "Table name".

The entries that should be deleted are defined in section "WHERE". Different filter options can be defined here.

The name of the desired column is specified using "Column". A value to be compared can be specified under "Value". "Operator" defines how the value should be compared with the column name.

If "LIKE" is used as an operator, similar entries can be searched for. Wildcard symbol "*" must be used. If "*Test*" is used as "Value", for example, entries "Test01", "MyTest" and "MyTest01" can be searched for.

In some databases, wildcard symbol "%" is also used.

Various filter options can be specified. These are connected to each other via "Connect".

**Custom**

A self-defined SQL query can be created under "Custom". The SQL code is added under "Query". For more information about creating SQL queries, see [here](#) for example.

The validity of the query cannot be checked in mapp Database. If errors occur during the execution of the query and additional information is required, see the command window for the script. For more information, see [here](#).

**3.1.1.1 Query examples**

This section lists examples of various query types.

The MySQL database system was used for the examples. The existing database "machinedata" contains a table named "product". Columns "id", "name" and "quality" are in the table. This database is used in the examples below.
Automation Studio has user-defined data type "Product_Type", which has the same structure as the table:

```
<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>INT</td>
</tr>
<tr>
<td>name</td>
<td>STRING[80]</td>
</tr>
<tr>
<td>quality</td>
<td>INT</td>
</tr>
</tbody>
</table>
```

Adding new data to the table ("INSERT")

The "INSERT" query type is used to add data to the table. The data can be added to the table automatically for all columns or this can be done individually. Variable "SetProductData" of data type "Product_Type" exists here.

Inserting automatically

```
SELECT * INTO product FROM ::DatabaseMg:SetProductData
```

Inserting individually

```
SELECT * INTO product WHERE quality <= 20 FROM ::DatabaseMg:SetProductData
```

Querying data from the table ("SELECT")

The "SELECT" query type is used to query and analyze data from the table.

The quality of the products should be checked. All bad products (quality ≤ 20) should be shown here.

This can be queried as follows:
It is also possible, for example, to combine several queries. This example queries the quality (#1) and the product name "RED" (#2) or "BLUE" (#3):

Deleting data from the table ("DELETE")
The "DELETE" query type is used to delete data from the table.
Since the product named "ProductZ" is no longer produced, it should be removed from the database. This can be done as follows.

In order to use different product names, a PV snippet can be used. This makes it possible to change the product name at runtime:

**Updating data in the table ("UPDATE")**

The "UPDATE" query type is used to update data in the table.

Adjustments were made to "ProductA" in production, so the quality of the product has increased. Therefore, the information in the database should be adapted. This can be done as follows:
Creating a table ("CREATE")

Query type "CREATE" is used to create a new table in the database.

A new table "shifts" is being created in the database, which should contain information about the shift.

In the example, a PV named "TableData" of data type "Shift_Type" has been specified. A table will be created in the database that uses precisely the variables found in data type "Shift_Type" as columns. In addition, column "mappId" will be created. When working with databases, we recommend marking the first column as "Primary key". For more information, see here. Since the "Primary key" cannot be defined in the mapp Database configuration, it is automatically created in the database.

The other columns have the same data types as the variables in Automation Studio.

Creating a user-defined query ("Custom")

A user-defined query can be created using the "Custom" query type.

The validity of the query cannot be checked in mapp Database. If errors occur during the execution of the query and additional information is required, see the command window for the script. For more information, see here.

Any type of query can be created using "Custom". PV snippets can also be used here:

It is also possible, for example, to change the query itself at runtime. The entire query in the configuration must be a PV snippet to do this. In the following example, variable "queryFlex" exists in a task named "process". This variable can be used to determine the content of the query:
4 Use cases

4.1 Use case 1: Adding production data to the database and querying it

Requirement
The objective is to collect statistical information about produced products in a MySQL database. The data in the database should then be reusable for various analyses and calculations.

Solution

Component list

- **MpDatabaseCore** (Custom MpLink): Establishes a connection to database
- **MpDatabaseQuery** (Custom MpLink): Starts specified queries

Connection diagram

Database
Database system MySQL is used. Database "machine" is created.
The database includes a table named "jobs" that has columns "id", "job", "goodParts" and "badParts". It should be possible to save and request product information in the table.
Configuration

The MpDatabaseCore configuration is added in order to establish a connection to a database. The connection parameters must be specified.

A connection to database "machine" should be established. When configuring the database, the user name, password, IP address and port number were specified. They must be specified in the configuration.

The queries must then be configured. A query is used to store information in the "jobs" table and one is used to read all information from the table.

Query for adding data to the table

Query "SetData" of type "INSERT" is created. "INSERT" can be used to add data to the specified database table.

The specified variable "JobData" is of data type "Job_Type". This data type contains the same variables as those in the database table. It is important that the data types are also identical.

Query for requesting data from the table

Query "GetData" of type "SELECT" is created. "SELECT" can be used to request data from the database table.
The specified variable "GetJobData" is an array of data type "Job_Type". This variable shows the requested production information.

For more information about the individual query types, see here.

**Script**

Before mapp Database can connect to the database, a script must be enabled. It is responsible for ensuring that data can be exchanged between the database and mapp Database. The script can be downloaded in section Installing the server. Additional information about the script can also be found there.

**Using the mapp components**

Function blocks **MpDatabaseCore** and **MpDatabaseQuery** are added. The function blocks are connected to one another as depicted in "Connection diagram".

The function blocks are then configured.

The configured queries can be started using **MpDatabaseQuery**. The query name defined in the configuration must be specified in input parameter "Name" of **MpDatabaseQuery**.

"Connect = TRUE" on **MpDatabaseCore** establishes a connection to the database. The query is started using "Execute = TRUE" on **MpDatabaseQuery**.

First, the production information can be saved in the database table using query "SetData". Then you can request the information from the table using query "GetData". The information ends up in process variable "GetJobData". Further analyses can then be carried out using the data obtained.

**5 Libraries**

**5.1 Data types**

**5.1.1 MpDatabaseCoreInfoType**

This data type provides additional information for the function block.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diag</td>
<td>MpDatabaseDiagType</td>
<td>Diagnostic structure for the status ID.</td>
</tr>
</tbody>
</table>

**5.1.2 MpDatabaseDiagType**

This data type is used as a substructure within the structure to hold additional information for diagnostic purposes as well as to supply additional data about the status ID.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusID</td>
<td>MpDatabaseStatusIDType</td>
<td>StatusID diagnostic structure.</td>
</tr>
</tbody>
</table>

**5.1.3 MpDatabaseQueryInfoType**

This data type provides additional information for the function block.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArraySize</td>
<td>UDINT</td>
<td>Specifies how many entries can be written to the array.</td>
</tr>
<tr>
<td>Rows</td>
<td>MpDatabaseQueryRowsInfoType</td>
<td>This parameter provides information about the started query.</td>
</tr>
<tr>
<td>Diag</td>
<td>MpDatabaseDiagType</td>
<td>Diagnostic structure for the status ID.</td>
</tr>
</tbody>
</table>

**5.1.4 MpDatabaseQueryRowsInfoType**

This data type provides information about the started query.
### 5.1.5 MpDatabaseStatusIDType

This data type is used as a substructure within the structure to hold additional information for diagnostic purposes as well as to supply additional data about the status ID.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>MpDatabaseErrorEnum</td>
<td>Error code for the function block.</td>
</tr>
<tr>
<td>Severity</td>
<td>MpComSeveritiesEnum</td>
<td>Describes the type of information supplied by the status ID (success, information, warning, error).</td>
</tr>
<tr>
<td>Code</td>
<td>UINT</td>
<td>Code for the status ID. This error number can be used to search for additional information in the help documentation.</td>
</tr>
</tbody>
</table>

### 5.2 Function blocks

**Name**

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpDatabaseCore can be used to connect to a database defined in MpDatabaseCore configuration.</td>
<td>MpDatabaseCore</td>
</tr>
<tr>
<td>MpDatabaseQuery can be used to start a query defined in the MpDatabaseCore configuration.</td>
<td>MpDatabaseQuery</td>
</tr>
</tbody>
</table>

#### 5.2.1 MpDatabaseCore

MpDatabaseCore can be used to connect to a database defined in MpDatabaseCore configuration.

**Function block**

---

## Optional parameters

```
| MpDatabaseCore &MpComIdentType MpLink |
|-----------------------------|--------------------------------------|
| Enable                      | BOOL                                 |
| Error                       | BOOL                                 |
| ErrorReset                  | BOOL                                 |
| StatusID                    | DINT                                 |
| Connect                     | BOOL                                 |
| CommandDone                 | BOOL                                 |
| Disconnect                  | BOOL                                 |
| Connected                   | BOOL                                 |
| Info                        | MpDatabaseCoreInfoType               |
```

```
| MpDatabaseCore &MpComIdentType MpLink |
|-----------------------------|--------------------------------------|
| Enable                      | BOOL                                 |
| Error                       | BOOL                                 |
| ErrorReset                  | BOOL                                 |
| StatusID                    | DINT                                 |
| Connect                     | BOOL                                 |
| CommandBusy                 | BOOL                                 |
| CommandDone                 | BOOL                                 |
| Connected                   | BOOL                                 |
| Info                        | MpDatabaseCoreInfoType               |
```
mapp Database: Connecting to a database

Interface

<table>
<thead>
<tr>
<th>I/O</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>Pointer to Mp-</td>
<td>Connection to mapp (MpLink of an MpDatabaseCore configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>IN</td>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>IN</td>
<td>Connect</td>
<td>BOOL</td>
<td>Command to connect to a database.</td>
</tr>
<tr>
<td>IN</td>
<td>Disconnect</td>
<td>BOOL</td>
<td>Command to disconnect from a database.</td>
</tr>
<tr>
<td>OUT</td>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</td>
</tr>
<tr>
<td>OUT</td>
<td>Error</td>
<td>BOOL</td>
<td>Error occurred during execution.</td>
</tr>
<tr>
<td>OUT</td>
<td>StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>OUT</td>
<td>Connected</td>
<td>BOOL</td>
<td>Indicates whether a connection to a database has been established.</td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>MpDatabaseCoreInfoType</td>
<td>Additional information about the component.</td>
</tr>
</tbody>
</table>

mapp concept

Section mapp components explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).

5.2.1.1 Description

The MpLink of an MpDatabaseCore configuration is used for this function block.

Use command "Connect = TRUE" to connect to a database defined in MpDatabaseCore configuration. Output parameter "Connected = TRUE" indicates that the function block has successfully connected to the database.

Command "Disconnect" is used to disconnect from the database.

The current status of the function block can be read from structure "Info" for the function block.

5.2.2 MpDatabaseQuery

MpDatabaseQuery can be used to start a query defined in the MpDatabaseCore configuration.

The contents of databases can be visually displayed in a mapp View HMI application using widget "Database".

Function block

## Optional parameters

<table>
<thead>
<tr>
<th>&amp;MpComIdentType</th>
<th>MpLink</th>
<th>Active</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>Enable</td>
<td>Error</td>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
<td>ErrorReset</td>
<td>StatusID</td>
<td>DINT</td>
</tr>
<tr>
<td>&amp;STRING</td>
<td>Name</td>
<td>CommandDone</td>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
<td>Execute</td>
<td>Info</td>
<td>MpDatabaseQueryInfoType</td>
</tr>
<tr>
<td>BOOL</td>
<td>Next</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
mapp Database: Connecting to a database

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;MpComIdentType</td>
<td>MpLink</td>
<td>Connection to mapp (MpLink of an</td>
</tr>
<tr>
<td>BOOL</td>
<td>Enable</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>BOOL</td>
<td>ErrorReset</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>&amp;STRING</td>
<td>Name</td>
<td>Name of the query. The query is defined in the MpDatabaseCore configuration.</td>
</tr>
<tr>
<td>BOOL</td>
<td>Execute</td>
<td>Command to start the query.</td>
</tr>
<tr>
<td>BOOL</td>
<td>Next</td>
<td>Command to display further entries of the query.</td>
</tr>
<tr>
<td>OUT</td>
<td>Active</td>
<td>Function block active.</td>
</tr>
<tr>
<td>OUT</td>
<td>Error</td>
<td>Error occurred during execution.</td>
</tr>
<tr>
<td>OUT</td>
<td>StatusID</td>
<td>Status information.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandBusy</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandDone</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>Additional information about the component.</td>
</tr>
</tbody>
</table>

mapp concept

Section mapp components explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).

5.2.2.1 Description

The same MpLink used by MpDatabaseCore is used for this function block.

Starting the query

Input parameter "Name" defines which query should be started. A name must be specified that was defined in the "Queries" section of the MpDatabaseCore configuration.

The query is started with command "Execute = TRUE". The information is shown on the process variable that was specified in the query defined in the MpDatabaseCore configuration.

"Next" can be used to request further information. The parameters in structure "Info" are required for this.

The following information can be taken from structure "Info":

Data sheet V 25
• **ArraySize**: Specifies how much space the process variable defined in the `MpDatabaseCore` configuration in section "Queries" offers. If the process variable is a STRING array of 100 elements, 100 is displayed.

• **Rows**: For further information about the executed query, see `MpDatabaseQueryRowsInfoType`. This includes the total number of entries found in the database ("Total"), how many entries have already been shown using the PV ("Read") and how many entries can still be read ("Remaining"). If a STRING array of 6 elements was used as the process variable, but the query returns 9 entries from the database, "Remaining=3" is shown. To get the last 3 entries, command "Next = TRUE" of `MpDatabaseQuery` must be used. The last 3 entries are written to the variable. The information is written to the first elements of the process variable.

![Diagram](image)

Remaining = 3

### 5.3 Status numbers

#### 5.3.1 1083194018: Table extended

**Description:**
A column has been added to the table.

#### 5.3.2 1083194019: Table not edited

**Description:**
The table has not been edited.

#### 5.3.3 1083194013: Buffer full

**Description:**
Data has been copied to the PV, but there is more information in the database. PV buffer is full.

**Cause/Solution:**
- Request further information by sending command "Next" to `MpDatabaseQuery`

#### 5.3.4 -1064239091: Invalid configuration

**Description:**
Could not read configuration while creating components. See the Logger for additional information.

**Reaction:**
The function block indicates on outputs "StatusID" and "Error" that an error is active. During this time, no other functions are available.
Cause/Solution:
- The configuration is damaged.
- The function is not enabled in the configuration.

These function blocks / functions can report this error:
- MpDatabaseCore

5.3.5 -1064239098: MpLink already in use

Description:
This MpLink is already in use.

Reaction:
The function block indicates on outputs "StatusID" and "Error" that an error is active. During this time, no other functions are available.

Cause/Solution:
- If "Overload" is the selected download mode, then the mapp component should be disabled in the exit subroutine with "Enable" = FALSE. It is also possible to use "Copy" or "One cycle" mode. In these cases, it is not necessary to disable the mapp component.
- Check which components are already using this MpLink.
- Create a new MpLink.

These function blocks / functions can report this error:
- MpDatabaseCore

5.3.6 -1064239099: Invalid MpLink contents

Description:
The value of variable "MpLink" on the function block input is invalid.

Reaction:
The function block indicates on outputs "StatusID" and "Error" that an error is active. During this time, no other functions are available.

Cause/Solution:
- Writing to the values in structure MpComIdentType is not permitted.
- A corresponding configuration for this mapp component must be available (see first paragraph of the description for the functions / function blocks listed below).

These function blocks / functions can report this error:
- MpDatabaseCore

5.3.7 -1064239100: MpLink modified

Description:
The value on input "MpLink" was modified while the components were running ("Enable" = TRUE).

Reaction:
The function block indicates on outputs "StatusID" and "Error" that an error is active. During this time, no other functions are available.

Cause/Solution:
- The value of input "MpLink" can only be changed while the component is inactive ("Enable" = FALSE).
These function blocks / functions can report this error:
   • MpDatabaseCore

5.3.8 -1064239101: MpLink connection not permitted

Description:
The value on input "MpLink" is not allowed.

Reaction:
The function block indicates on outputs "StatusID" and "Error" that an error is active. During this time, no other functions are available.

Cause/Solution:
   • The component is connected directly to mpCOM_MAIN or mpCOM_STANDALONE. This is not supported.

These function blocks / functions can report this error:
   • MpDatabaseCore

5.3.9 -1064239102: MpLink is null pointer

Description:
Input "MpLink" is not connected, null pointer.

Reaction:
The function block indicates on outputs "StatusID" and "Error" that an error is active. During this time, no other functions are available.

Cause/Solution:
   • Check input "MpLink" on the function block.

These function blocks / functions can report this error:
   • MpDatabaseCore

5.3.10 -1064239103: Could not create component

Description:
The mapp component could not be created and is not enabled. See the Logger for additional information.

Reaction:
The function block indicates on outputs "StatusID" and "Error" that an error is active. During this time, no other functions are available.

Cause/Solution:
   • Service for configuring the function block not available → Problem with MpCom
   • Unable to read registry -> Problem with MpCom
   • Details about the cause of error in the logger

These function blocks / functions can report this error:
   • MpDatabaseCore

5.3.11 -1064167416: Query error

Description:
An error occurred while interpreting the SQL query. Cause of error: {2:ErrorNumber}
Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Additional information:
• {2:ErrorNumber}: Cause of error

Cause/Solution:
• Check the query to determine if correct table and column names were used.

Constant
mpDATABASE_ERR_QUERY_RESULT

These function blocks / functions can report this error:
• MpDatabaseQuery

5.3.12 -1064167419: Invalid data type

Description:
The data type {2:TypeName} is not supported.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Additional information
• {2:TypeName}: Data type

Cause/Solution:
• Use a different data type. For a list of supported data types, see here in section "Specifying variables"

Constant
mpDATABASE_ERR_INVALID_DATATYPE

These function blocks / functions can report this error:
• MpDatabaseQuery

5.3.13 1083316228: No further data

Description:
All of the requested data has already been copied to the specified PV. No further data is available.

Reaction:
The function block indicates this information on output "StatusID".

Cause/Solution:
• No further data requests by sending "Next = TRUE" to MpDatabaseQuery
• Start a new query.

Constant
mpDATABASE_INF_NO_DATA

These function blocks / functions can report this error:
• MpDatabaseQuery
5.3.14 -1064167421: Double query

**Description:**
The query `{2:QueryName}` cannot be added because the query already exists.

**Reaction:**
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

**Additional information**
- `{2:QueryName}`: Name of the query

**Cause/Solution:**
- Use a different query name.

**Constant**
`mpDATABASE_ERR_DUPLICATE_QUERY`

**These function blocks / functions can report this error:**
- MpDatabaseQuery

5.3.15 -1064167422: Invalid query

**Description:**
The query `{2:QueryName}` does not exist.

**Reaction:**
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

**Additional information**
- `{2:QueryName}`: Name of the query

**Cause/Solution:**
- Check the specified query name.
- Use a different query name.

**Constant**
`mpDATABASE_ERR_INVALID_QUERY`

**These function blocks / functions can report this error:**
- MpDatabaseQuery

5.3.16 -1064167423: HTTP server error

**Description:**
Cannot connect to database. The server reports an error. See the Logger for additional information. Cause of error: `{2:ErrorNumber}`

**Reaction:**
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.
Additional information

- `{2:ErrorNumber}`: Error number

Cause/Solution:

- Check the specified IP address and port number in the `MpDatabaseCore configuration`.
- Check if the script has been enabled. For more information, see [here](#).
- See the Logger for additional information.
- Check the proposed solutions in section Diagnostics.

Constant

`mpDATABASE_ERR_HTTP_SERVER`

These function blocks / functions can report this error:

- `MpDatabaseCore`

5.3.17 -1064167424: SQL server error

Description:

Cannot connect to database. The SQL server reports an error. See the Logger for additional information. Cause of error: `{2:ErrorNumber}`

Reaction:

The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Additional information

- `{2:ErrorNumber}`: Error number. This is an SQL error number that is available [here](#), for example.

Cause/Solution:

- Check the specified database name in the `MpDatabaseCore configuration`.
- See the Logger for additional information.
- For additional suggested solutions, see section Diagnostics.

Constant

`mpDATABASE_ERR_SQL_SERVER`

These function blocks / functions can report this error:

- `MpDatabaseCore`

5.3.18 -1064289631: Incorrect PV specification

Description:

The PV specified in the SQL query does not match the columns of the table in the database.

Cause/Solution:

- Review the PV
- If a structure is specified, the PV names must match the column names of the table in the database.

5.3.19 -1064289636: PV already exists

Description:

The process variable `{2:PVName}` already exists in this query.
mapp Database: Connecting to a database

Additional information
  • {2:PVName}: Process variable

Cause/Solution:
  • Check the PV.
  • Specify a different PV

5.3.20 -1064289637: SQL error message

Description:
The SQL server reports error {2:ErrorNumber} with error message {3:ErrorMessage}.

Additional information
  • {2:ErrorNumber}: Error number
  • {3:ErrorMessage}: Error message

Cause/Solution:
  • See the Logger for additional information.
  • For additional suggested solutions, see section Diagnostics.

5.3.21 -1064289638: Invalid HTTP response

Description:
The HTTP response is invalid. The script used to connect to the database is invalid.

Cause/Solution:
  • Checking the Python script. For more information, see here. For additional suggested solutions related to this task, see section Diagnostics.

5.3.22 -1064289639: Invalid HTTP request

Description:
HTTP request invalid. The script used to connect to the database is invalid.

Cause/Solution:
  • Checking the Python script. For more information, see here. For additional suggested solutions related to this task, see section Diagnostics.

5.3.23 -1064289640: SQL server not connected

Description:
There is no connection to the SQL server.

Cause/Solution:
  • Check the general settings in the MpDatabaseCore configuration
  • For additional suggested solutions, see section Diagnostics.

5.3.24 -2137909241: Warning for query

Description:
When interpreting the SQL query, PV {2:PvName} triggered a warning.

Reaction
The function block indicates an active warning on output "StatusID".
Additional information

• {2:PVName}: Process variable

Cause/Solution:

• Specify a different data type for the process variable. The permitted data types are listed in section Specifying variables.
• A column returns the value NULL.
• See the Logger for additional information.

Constant

mpDATABASE_WRN_QUERY_RESULT

5.3.25 -2138031458: PV not found

Description:
Process variable {2:PVName} could not be located.

Additional information

• {2:PVName}: Process variable

Cause/Solution:

• Check the process variable.
• Check if the process variable is used in the program.

5.3.26 -2138031457: PV outside limit

Description:
The process variable is outside the permitted limit.

Cause/Solution:

• Check the process variable.
• Specify a different data type for the PV.

5.3.27 -2138031456: Value of database is NULL

Description:
The value received from the database is NULL.

Cause/Solution:

• Check the value in the database

5.3.28 -2138031455: PV not matching column

Description:
A process variable whose structure does not match the column in the database table was used for a query. The structure of the PV contains variables that do not exist in the database table.

Cause/Solution:

• Specify a PV whose structure matches the structure in the database table.
• For additional information, see "Messages in the command-line interface" in section Diagnostics.

Constant

mpDATABASE_WRN_RESPONSE_TO_PV_MISMATCH