Akıllı Kontrolde Teknoloji Devi

MDC GATEWAY READER

Application Note

- Load Profile Integration

10 / 2025

MIKRODEV_AN_MDC_Gateway_Reader/Load_Profile_Integration

CONTENTS

About Mikrodev

Since 2006, MIKRODEV has been developing and manufacturing industrial control and communication products. MIKRODEV serves the system integrators in the public and private sector, OEM and end users.

Our products are manufactured complying with the quality standards required by the industrial automation industry and the quality of our products are proved on the field for many years

MIKRODEV is one of the few companies in the world that has its own designed IEC 61131-3 compliant library for its programmable logic control devices. In addition, the open, flexible, programmable SCADA solution developed by MIKRODEV is also available to customers.

MIKRODEV products' performance and wide range of applications make them possible for customers to achieve faster, simplified and cost-effective results.

WARNING!

- ✓ Use the programming editor only for Mikrodev Certifed devices
- ✓ When you change your physical hardware configuration, update your development to the appropriate version.
- ✓ The developed program should be tested separately before taking to field service and should be shipped to the field after the tests are successfully completed.
- ✓ Take all accident prevention measures and safety measures identified by local law

Failure to comply with these rules may result in death, serious injury or property damage

1 Load Profile Integration

The **MDC Gateway Reader** software works in conjunction with **PostgreSQL.**

The program creates a new database with the same name as the configuration file generated by the software.

Under the **Schemas** section, there are two schemas named **"logs"** and **"public."**

1.1 Public Schema

Under the **"public"** schema, there are two tables:

- **mdcgateway:** Stores gateway information.
- **mdcmeter:** Stores meter information.

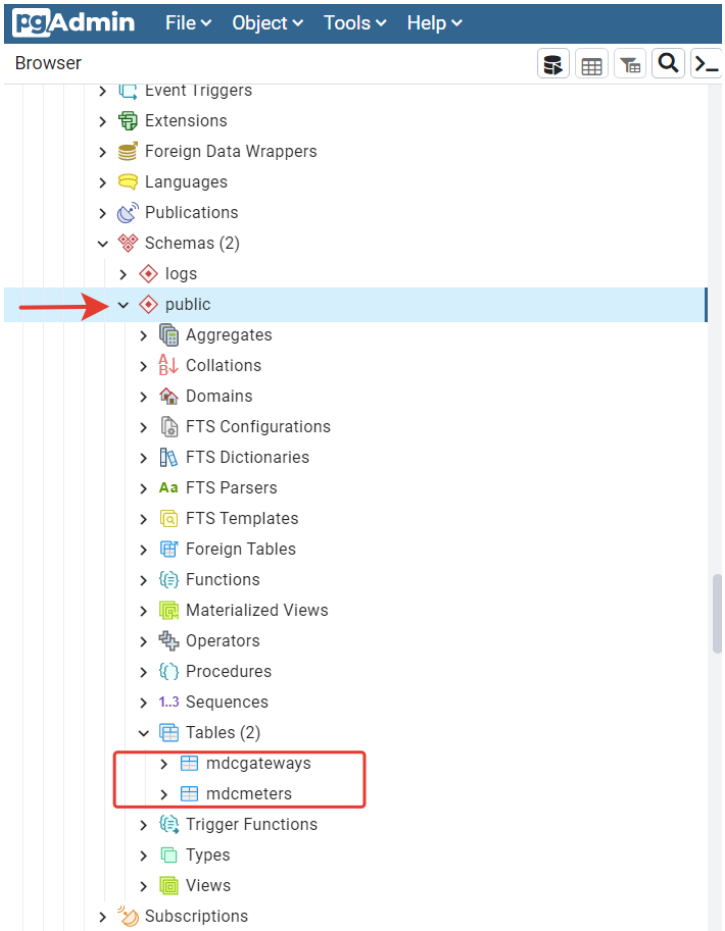

Figure 1 Public Schema

1.1.1 mdcgateways Table

This table stores information about the gateway devices added to the software, such as IP address, port number, and device name.

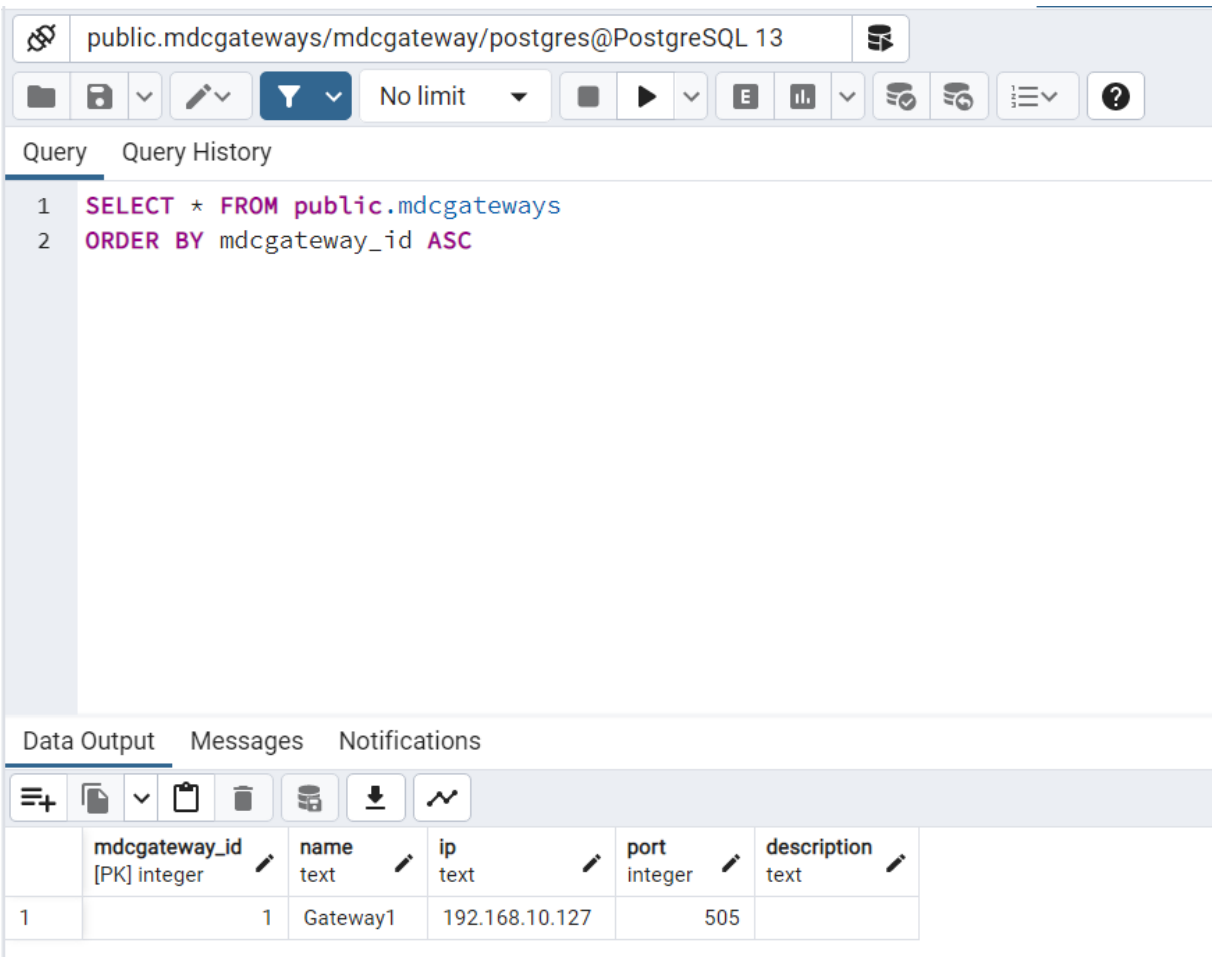

Figure 2 mdcgateways Table

1.1.2 mdcmeters Table

This table stores information such as meter serial number, meter ID, and meter name. Load profile records can be retrieved according to the **meter ID** specified in this table.

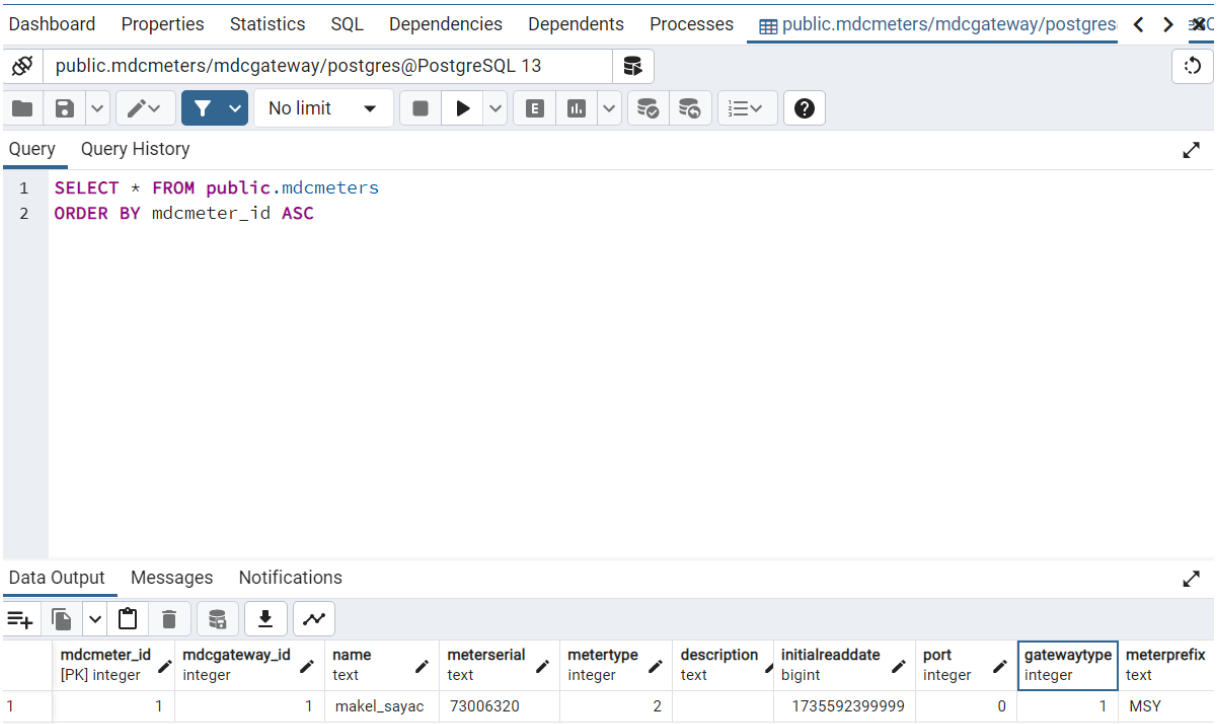

Figure 3 mdcmeters Table

Metertype values:

Value	Meter Model
1	Makel C500 KMY
2	Makel C41 KMY
3	Köhler AEL TF 11
4	Köhler AEL MF 14
5	Köhler AEL TF 19
6	EMH 6 LZQ
7	Makel C520 AMT
8	Köhler AEL TF 09
9	Köhler AEL TF 22
10	Landis LGZ5
11	Elster A1350
12	Elster A1440
13	Makel T600
14	Makel T610

1.2 Logs Schema

The **"logs"** schema stores parsed **readout data**, parsed **load profile records**, and **latest successful load profile records**.

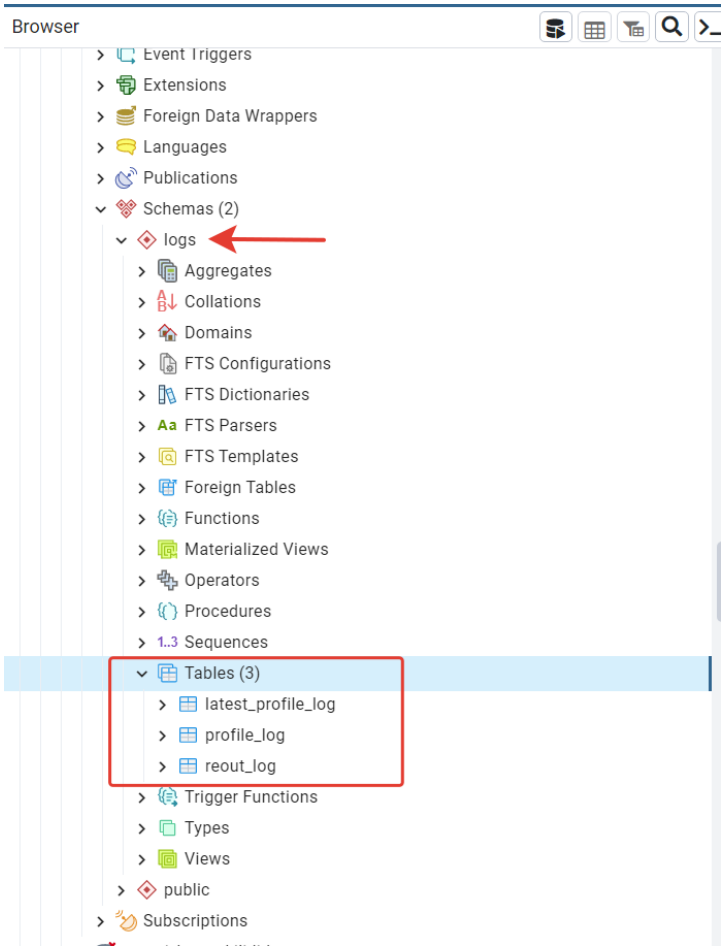

Figure 4 Logs Schema

Under the **logs** schema:

- **logs.latest_profile_log** – The latest successfully read load profile record.
- **logs.profile_log** – Contains all load profile records.
- **logs.reout_log** – Stores predefined index values from the meters' readouts.

1.2.1 profile_log Table

This table stores all load profile data.

The information here is raw data received directly from the meters.

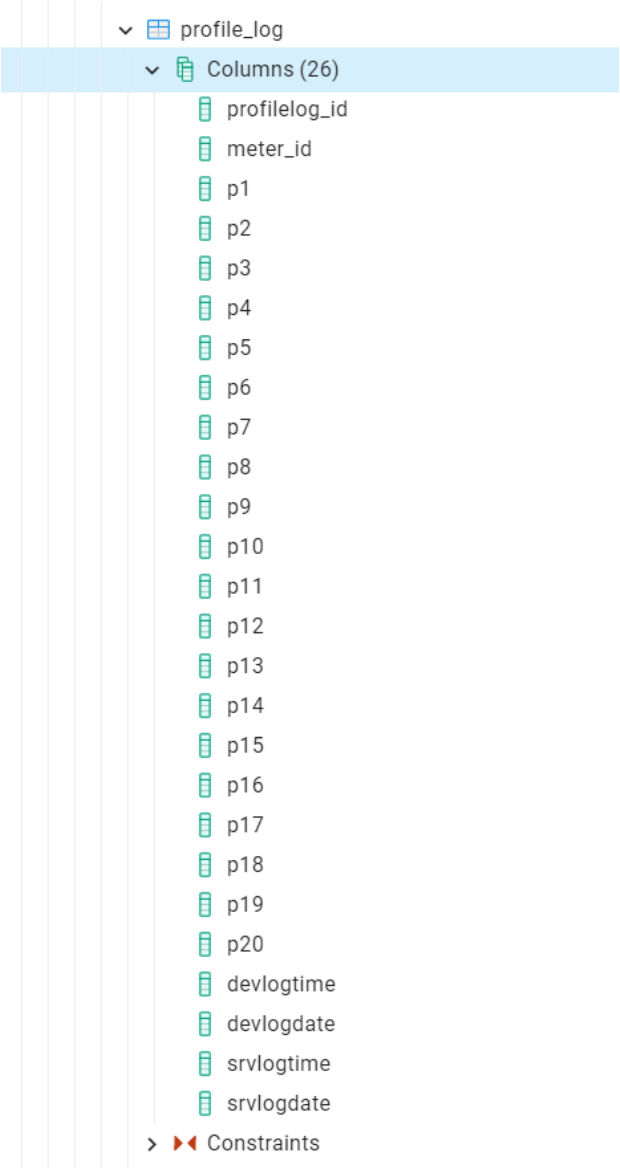

Figure 5 profile_log Table

Column	Description
profilelog_id	Log ID
meter_id	Meter ID
p1	T+ value
p2	Ri+ value
p3	Rc+ value
p4	IrmsA value
p5	IrmsB value
p6	IrmsC value
p7	VrmsA value
p8	VrmsB value
p9	VrmsC value
p10	T− value
p11	Ri− value
p12	Rc− value
p13	Profile index (for Köhler meters)
p14–p19	EMH meter 15-minute differential profile values (T+, Ri+, Rc+, T−, Ri−, Rc−)
devlogtime	Profile timestamp read from the meter
devlogdate	Profile date and time read from the meter
svrlogtime	Timestamp of when the meter was read
svrlogdate	Date and time of when the meter was read

These values apply to all meters.

Because EMH meters have a different load profile structure, their values are stored in different columns.

For some meters, certain parameters may not exist — in such cases, the corresponding column will contain a value of **"-1."**

1.2.2 latest_profile_log Table

This table has the same columns as **logs.profile_log**, but it only stores the **latest load profile record** read from the meter.

The next profile query to be sent to the meter is generated based on the date stored in this table.

Figure 6 latest_profile_log Table

1.2.3 reout_log Table

This table stores the values returned from the meters' **readout** operation. The OBIS codes to be logged in this table are predefined in the program. Below is the correspondence of OBIS codes to the columns in the table.

Figure 7 reout_log Table

Column	OBIS Code	Description
reoutlog_id	–	Readout Log ID
meter_id	–	Meter ID
r0	0.0.0	Meter Serial Number
r1	1.8.0	Cumulative Active Energy (+)
r2	1.8.1	T1 Active (+)
r3	1.8.2	T2 Active (+)
r4	1.8.3	T3 Active (+)
r5	5.8.0	Reactive Inductive (+)
r6	6.8.0	Reactive Capacitive (−)
r7	7.8.0	Reactive Inductive (−)
r8	8.8.0	Reactive Capacitive (+)
r9	2.8.0	Cumulative Active Energy (−)
r10	2.8.1	T1 Active (−)
r11	2.8.2	T2 Active (−)
r12	2.8.3	T3 Active (−)
r13	1.6.0	Active Demand (+) [Value]
r14	1.6.0	Active Demand (+) [Date]
r15	2.6.0	Active Demand (−) [Value]
r16	2.6.0	Active Demand (−) [Date]
r17–r28	Various	Previous Month Start Values (e.g. 1.8.x*1, 5.8.0*1, etc.)
r29–r32	Various	Previous Month Demand Values (1.6.0*1, 2.6.0*1)
r33	0.9.1	Meter Time **
r34	0.9.2	Meter Date **
r35	96.70	Meter Cover Open Date
r36	96.71	Terminal Cover Open Date
r37	96.71	Terminal Cover Opening Count
r38	96.6.1	Low Battery Warning
r39	96.7.0	Power Outage Count
r40–r42	–	Phase Interruption Start Date-Time (R, S, T)
r43–r45	–	Phase Interruption End Date-Time (R, S, T)

Column	OBIS Code	Description
r46–r48	–	Phase Currents (R, S, T)
r49–r51	–	Phase Voltages (R, S, T)
r52–r54	–	Phase Interruption Counts (R, S, T)
r55	–	Three-Phase Interruption Start Date
r56	–	Three-Phase Interruption End Date
r79	–	GSM Signal Strength
svrlogtime	–	Timestamp of meter read time
svrlogdate	–	Date and time of meter read time

**Epoch Time Conversion

To convert the epoch-based meter time to human-readable format: Use the following expression: (r33 / 1000 + r34 / 1000 + 10800)

Example PostgreSQL command:

SELECT to_timestamp(r33 / 1000 + r34 / 1000 + 10800);

Empty columns indicate that the corresponding OBIS value was **not provided** by the meter.