



**The DLMS COSEM plugin  
PRINTED MANUAL**

# DLMS COSEM plugin

© 1999-2024 AGG Software

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: 11/2/2024

## **Publisher**

*AGG Software*

## **Production**

© 1999-2024 AGG Software

*<http://www.aggsoft.com>*

---

# Table of Contents

<b>Part 1 Introduction</b>	<b>1</b>
<b>Part 2 System requirements</b>	<b>1</b>
<b>Part 3 Installing DLMS COSEM</b>	<b>2</b>
<b>Part 4 Glossary</b>	<b>3</b>
<b>Part 5 Requests queue</b>	<b>3</b>
<b>Part 6 Request method</b>	<b>7</b>
<b>Part 7 Cron time format</b>	<b>8</b>

## 1 Introduction

DLMS/COSEM (IEC 62056, EN13757-1) is the global standard for energy and water smart management, advanced control, and innovative metering. The plugin implements some parts of the DLMS/COSEM protocol to read current values.

The protocol specifies an object-oriented data model, an application layer protocol, and media-specific communication profiles.

DLMS/COSEM comprises three key components:

- COSEM - Companion Specification for Energy Metering - the object model capable of describing virtually any application;
- OBIS - Object Identification System, the naming system of the objects; and DLMS - Device Language Message Specification - the application layer protocol that turns the information held by the objects into messages.
- DLMS/COSEM can be used for all utilities/energy kinds, all market segments, all applications and virtually over any communication media.

This module has the following features:

- It can read data from any DLMS-compatible device.
- It can initiate a connection using the IEC 62021 protocol.
- The plugin can work with serial and network interfaces.
- The module can retrieve an objects list from a device.
- It can poll DLMS data by a custom interval.
- It can poll data from multiple devices at the same time.
- The module can connect to password-protected devices.

## 2 System requirements

The following requirements must be met for "DLMS COSEM" to be installed:

**Operating system:** Windows 2000 SP4 and above, including both x86 and x64 workstations and servers. The latest service pack for the corresponding OS is required.

**Free disk space:** Not less than 5 MB of free disk space is recommended.

**Special access requirements:** You should log on as a user with Administrator rights in order to install this module.

The main application (core) must be installed, for example, Advanced Serial Data Logger.

### 3 Installing DLMS COSEM

1. Close the main application (for example, Advanced Serial Data Logger) if it is running;
2. Copy the program to your hard drive;
3. Run the module installation file with a double click on the file name in Windows Explorer;
4. Follow the instructions of the installation software. Usually, it is enough just to click the "Next" button several times;
5. Start the main application. The name of the module will appear on the "Modules" tab of the "Settings" window if it is successfully installed.

If the module is compatible with the program, its name and version will be displayed in the module list. You can see examples of installed modules on fig.1-2. Some types of modules require additional configuration. To do it, just select a module from the list and click the "Setup" button next to the list. The configuration of the module is described below.

You can see some types of modules on the "Log file" tab. To configure such a module, you should select it from the "File type" list and click the "Advanced" button.

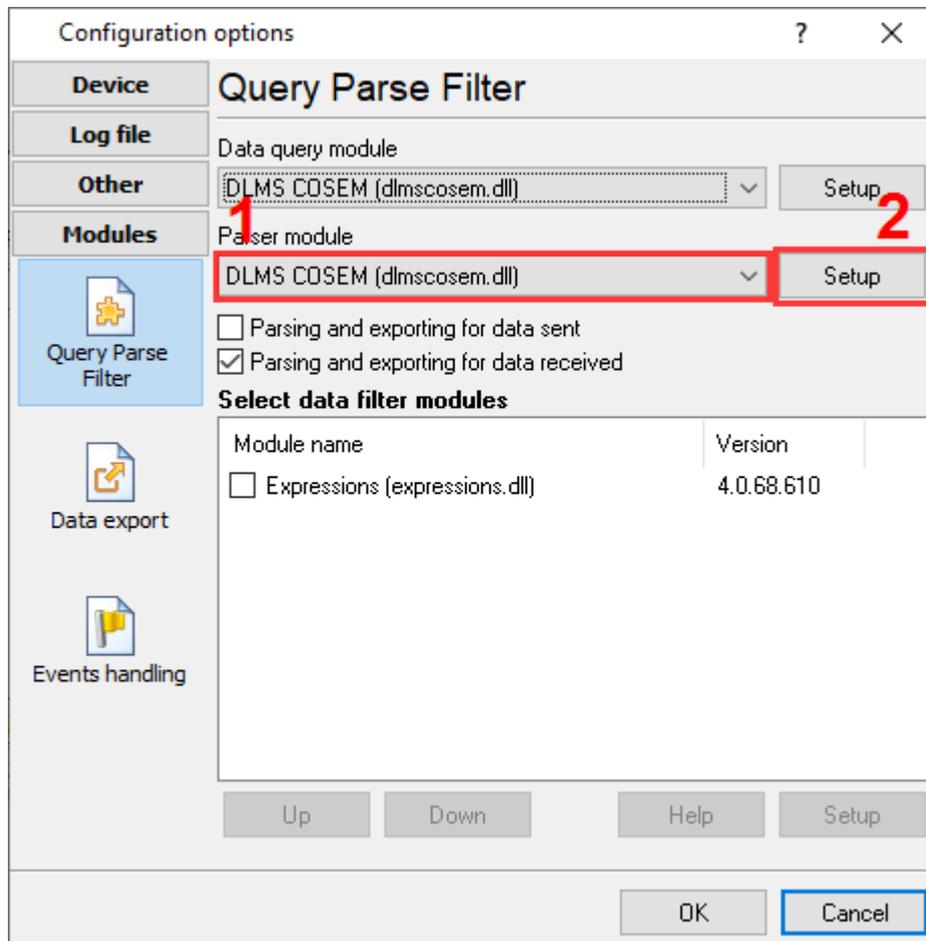


Fig. 1. Example of installed module

## 4 Glossary

**Main program** - it is the main executable of the application, for example, Advanced Serial Data Logger and asdlog.exe. It allows you to create several configurations with different settings and use different plugins.

**Plugin** - it is the additional plugin module for the main program. The plugin module extends the functionality of the main program.

**Parser** - it is the plugin module that processes the data flow, singling out data packets from it, and then variables from data packets. These variables are used in data export modules after that.

**Core** - see "Main program."

## 5 Requests queue

To add a new item, click "Actions - Add new request." The dialog window will be shown (fig.1). Enter a request description that can contain any characters and click the "OK" button.

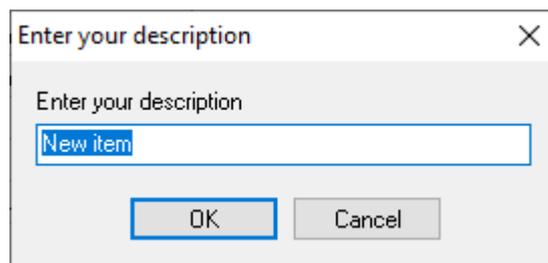


Fig.1. Name dialog

The new DLMS request will appear in the requests tree (fig.2). Each request has few important options:

**Physical address, Logical address** - In DLMS/COSEM, a physical smart meter device can host several so called logical devices. Each logical device has an address in the range [0, 16383]. For example, a meter could consist of one logical device for electricity metering at address 18 and another for a connected gas meter at address 67. No matter what type of smart meter it is, a DLMS/COSEM device must contain a special logical device called the logical management device at address 1. The content of this device may vary, but it must at least have a list of all logical devices inside the physical smart meter. You are always connecting to a single logical device with DLMS/COSEM.

If you poll several identical physical devices on the same bus, you can specify all addresses in one request in the "Physical address" field, like 1,2,3,4.

**Address size** (default: 1) - some devices may require an explicit address size for physical and logical addresses. Even if a device address can be sent, the device may expect the address in two or four bytes.

---

## Protocol mode

DLMS - the program interacts with a device using the DLMS protocol.

DLMS, start with IEC - the program initiates a connection using the IEC 62021 protocol and changes a baud rate if a remote device asks for it. It is mainly used with the optical, RS232 or RS485 interface.

## Authentication

No security - the program connects to a device without a password and uses the 16 (10h) client address. Most devices allow reading data without a password.

Low security - the program connects to a device with the specified password and uses the 32 (20h) client address. The password length is 8 characters.

## Referencing

Logical name - A logical name is actually an OBIS code. Sometimes, it is also called instance ID. It is a 6-byte number that uniquely identifies an object in a logical device. For example, the clock of a smart meter is always reachable under the address "0.0.1.0.0.255".

Short name - the second way to address a value is the so-called short address. Short addresses are used for small devices and should only be used if the connected smart meter cannot communicate using logical names.

**Request timeout** - After reaching the timeout limit, the program will automatically cancel the current request and execute the next request in the queue. The timeout value depends on the media speed on device performance, and if the data transfer rate over media is slow, then the timeout value should be larger.

DLMS COSEM 4.0.2.917

**Requests queue**

Property	Value
<b>Request 1</b>	
<input checked="" type="checkbox"/> Send requests, otherwise parse response only	
Physical address	1
Logical address	1
Address size (bytes)	2
Protocol mode	DLMS
Referencing	Logical name
<b>Authentication</b>	
<input checked="" type="radio"/> No security	
<input type="radio"/> Low security	
Request timeout (ms)	1500
<b>Request method</b>	
<input type="radio"/> Once, on the program startup	
<input checked="" type="radio"/> Polling	
Interval (ms)	1000
Interval units	Millisecond
<input type="radio"/> At specified time	
<input type="radio"/> Time using Unix Cron schedule	
<input type="radio"/> Event	
<b>Response items</b>	
<b>Item #1</b>	
Export name	ITEM
Data type	Auto
OBIS code	0.0.1.0.0.255
Scale (numbers only)	1

Action ▼

Minimal interval between data packets (ms)

OK Cancel

Fig.2. DLMS request

## Response items

This group allows you to add one or more items that the program should read from the specified device.

---

You can add a new parser item (variable) to the request by clicking "Actions - Add response item." Before, you should select a caption of the corresponding request. A new parser item (variable) will appear in the "Response items" group (fig.2).

Each response item has few important options:

**Export name** - it is the name of the parser variable. This name you'll bind with fields in data export plugins.

**OBIS code** - the logical name of the necessary value. Even if a device uses short name referencing, you should specify the logical name here, and the program will automatically convert it to a short name.

**Data type** - the data type of the value.

**Scale** - this optional parameter allows you to scale a numerical read value.

### Obtaining object list from a device

1. Configure a connection with your device in the main program and check that the program can successfully connect to the configured interface (a COM port or network).
2. Click the "Actions - Read object list" menu item to open the dialog window will be shown (fig.3).
3. Specify connection parameters (server and client addresses)
4. Click the "Read object list" button. The reading process may take a while.
5. After a successful reading, the program will fill the list with OBIS codes for your reference and save it to a template file. Later, you can add requests to a queue using that template by clicking the "Action - Add using preset" menu.

OBIS	Name	Unit	Data type
------	------	------	-----------

Fig.3. OBIS codes

## 6 Request method

The plugin can send requests in the following mode:

**Once, on program startup** - the program will send a request once when the program starts.

**Polling** - the program will send a request periodically based on an interval specified. The interval between requests depends on the network on which master (program) and slave (device) is running. If the network is slow, then the time for each request will be larger and vice versa. Because the program executes all requests in the queue one by one, the time between requests depends on the number of requests in the queue.

**At the specified time** - the time of the day using the 24hr format (e.g., 18:00:00). You may specify several time points separated by a semicolon (e.g. 11:00:00;11:20:00;11:40:00).

**Time, using Unix Cron schedule** - a flexible schedule format that allows sending requests periodically or at the specified time. You can find detailed information about this format and see examples in the "Cron time format" section. The default is 0 0 12 \* \* \*, which means "every week, every day at 12:00:00".

**Event** - the program executes the corresponding request when the plugin receives an external event. These events can be generated by our other plugins, like "Event generator," "Script execute," "Expressions,"

**Request method**

---

Once, on the program startup

---

Polling

Interval (ms)	10000
Interval units	Millisecond

---

At specified time

---

Time using Unix Cron schedule

---

Event

Fig. 2. Request methods

If you have added several requests to the queue, you can move them up or down. To do it, select a request, click the "Action" button, and select an action ("Move up" or "Move down").

You can also click this button to change a request's description or remove a request from the queue.

You can also perform the same actions by using the context menu that pops up when you right-click items in the request tree.

## 7 Cron time format

The CRON format is a simple yet powerful way to describe time and operation periodicity. The traditional (inherited from the Unix world) CRON format consists of five fields separated with spaces:

<Second> <Minutes> <Hours> <Month days> <Months> <Weekdays>

Any of the five fields can contain the \* (asterisk) character as its value. It stands for the entire range of possible values. For example, every minute, every hour and so on. In the first four fields, you can also use the proprietary "?" (w/o quotes) character. See its description below.

Any field can contain a list of comma-separated values (for example, 1,3,7) or an interval (subrange) of values defined by a hyphen (for example, 1-5).

You can use the / character after the asterisk (\*) or after an interval to specify the value increment. For example, you can use 0-23/2 in the "Hours" field to specify that the operation should be carried out every two hours (old version analog: 0,2,4,6,8,10,12,14,16,18,20,22). The value \*/4 in the "Minutes" field means that the operations must be carried out every four minutes. 1-30/3 is the same as 1,4,7,10,13,16,19,22,25,28.

You can use three-word abbreviations in the "Months" (Jan, Feb, ..., Dec) and "Weekdays" (Mon, Tue, ..., Sun) fields instead of numbers.

### Examples

Note: the <Second> field equal 0 in all examples

Format	Description
* * * * *	every minute
59 23 31 12 5	one minute before the end of the year if the last day in the year is Friday
59 23 31 Dec Fri	one minute before the end of the year if the last day in the year is Friday (one more variant)
45 17 7 6 *	every year on the 7th of June at 17:45
0,15,30,45 0,6,12,18 1,15,31 * 1-5 *	00:00, 00:15, 00:30, 00:45, 06:00, 06:15, 06:30, 06:45, 12:00, 12:15, 12:30, 12:45, 18:00, 18:15, 18:30, 18:45, if it is the 1st, 15th or 31st of any month and only on workdays
*/15 */6 1,15,31 * 1-5	00:00, 00:15, 00:30, 00:45, 06:00, 06:15, 06:30, 06:45, 12:00, 12:15, 12:30, 12:45, 18:00, 18:15, 18:30, 18:45, if it is the 1st, 15th or 31st of any month and only on workdays (one more variant)
0 12 * * 1-5 (0 12 * * Mon-Fri)	at noon on workdays
* * * 1,3,5,7,9,11 *	every minute in January, March, May, July, September, and November
1,2,3,5,20-25,30-35,59 23 31 12 *	on the last day of the year at 23:01, 23:02, 23:03, 23:05, 23:20, 23:21, 23:22, 23:23, 23:24, 23:25, 23:30, 23:31, 23:32, 23:33, 23:34, 23:35, 23:59
0 9 1-7 * 1	on the first Monday of every month at 9 in the morning
0 0 1 * *	at midnight on the 1st of every month
* 0-11 * *	every minute till noon
* * * 1,2,3 *	every minute in January, February, and March
* * * Jan, Feb, Mar *	every minute in January, February, and March
0 0 * * *	every day at midnight
0 0 * * 3	every Wednesday at midnight

You can use the proprietary "?" character in the first four fields of the CRON format. It stands for the start time, i.e., the question mark will be replaced with the start time during the field processing: minute for the minute field, hour for the "Hours" field, month day for the month day field, and month for the month field.

For example, if you specify:

? ? \* \* \*

The task will be run at the moment of startup and will continue being run simultaneously (if the user does not restart the program again, of course) – the question marks are replaced with the time the program was started at. For example, if you start the program at 8:25, the questions marks will be replaced like this:

```
25 8 * * * *
```

Here are some more examples:

- ? ? ? ? \* - run `_only_` at startup;
- ? \* \* \* \* - run at startup (for example, at 10:15) and continue being run in exactly one hour: at 11:15, 12:15, 13:15 and so on;
- \* ? \* \* \* - run every minute during the startup hour;
- \*/5 ? \* \* \* - run on the next day (if CRON is not restarted) at the same hour every minute and so on every day, once in five minutes, during the startup hour.