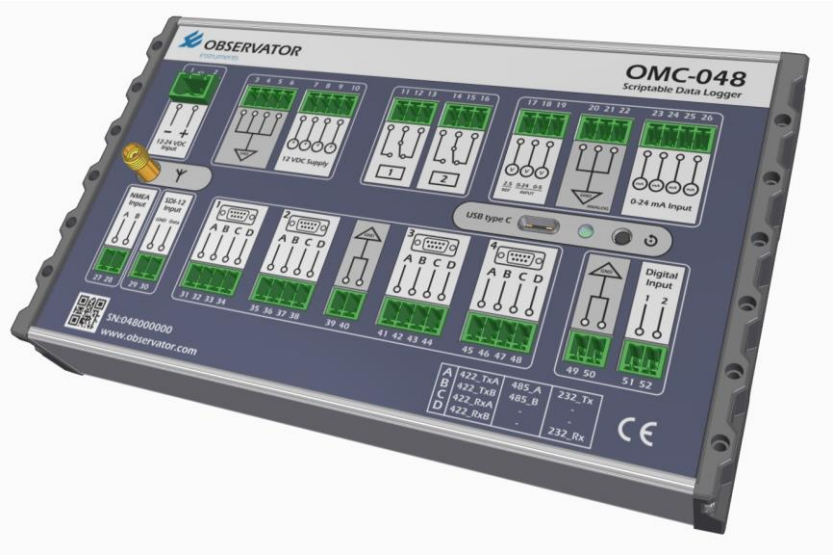
OBSERVATOR
instruments



Users' Manual

OMC-048

Scriptable Data logger

Version: 1.0
Status: Released
Date: 16 November 2020
Author: Observator

Document history

The Observator range is in continuous development and so specifications may be subject to change without prior notice. When in doubt about the accuracy of this document, contact the Observator Group.

Revision history

Date	Amendments	Company, position
2020-7-1	Initial document creation	Observator, TDC
2020-8-4	Pre- release	Observator, TDC
2020-11-16	First Release	Observator, TDC

Preface

This manual is intended for the user of the OMC-048 Scriptable Data logger.
You will find all information related to setting up the basics, connecting your sensors, as well as how to update and maintain your logger.
Detailed and up to date information concerning drivers and firmware is available in the OMC-048 web based manual, which is included on the loggers sd card and will be available via the Observator.com website soon.
It does not include explanation of the Python scripting language. It assumes Observator Instruments or a third party supplies the Python application for your logger.

For correct functioning of this system the OMC-048 Data logger and connected sensors must be installed and commissioned according installation instructions.

Note the correct power supply voltage:

This a 12-24VDC system
Use external fuse!

After end of life dispose this product according local regulations or return to manufacturer.

What's in the box

1		OMC-048 Scriptable Data Logger
2		Micro SD card 32GB (installed in the logger
3		Backup battery 3V coin cell (installed in logger)
4		USB cable C-type
5		This manual

Table of contents

1 Introduction

This versatile data logger is the successor of the successful OMC-045-3 data logger. It has a large number of inputs and outputs, including SDI-12, ModBus and analog. The OMC-048 uses the latest technology and includes a powerful processor as well as a state-of-the-art LTE ‘world wide’ modem for connecting to the cellular network.
Thanks to the advanced processor and the availability of a large collection of input and output drivers, it is possible to write your own programs (Python scripts) for maximum flexibility. For common applications, scripts will be available for you to download, so most users will be able to use the logger without any programming.

2 First use

2.1 Requirements

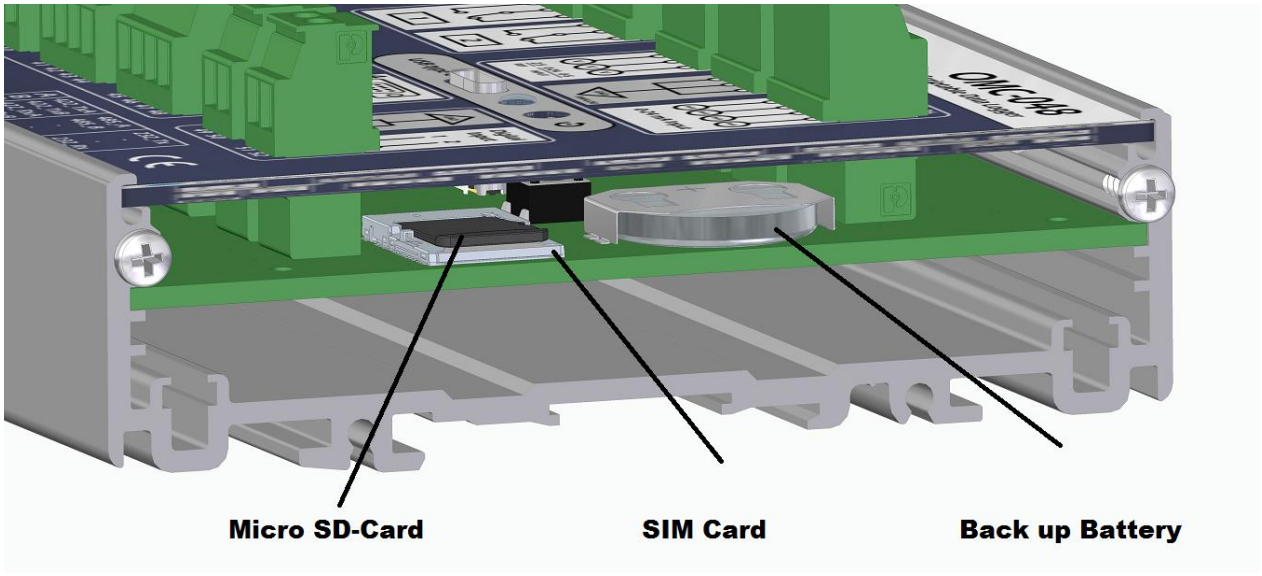
For setting up your OMC-048 Scriptable Data Logger you will require the following:

1. USB –C cable (included)
2. PC / laptop
3. Terminal program
4. External power supply (optional)
5. Cellular antenna (suitable for the band you will use)
6. Sensors (optional)
7. Phillips screwdriver
8. 2mm flat screwdriver

2.2 Installation SIM card & activation backup battery

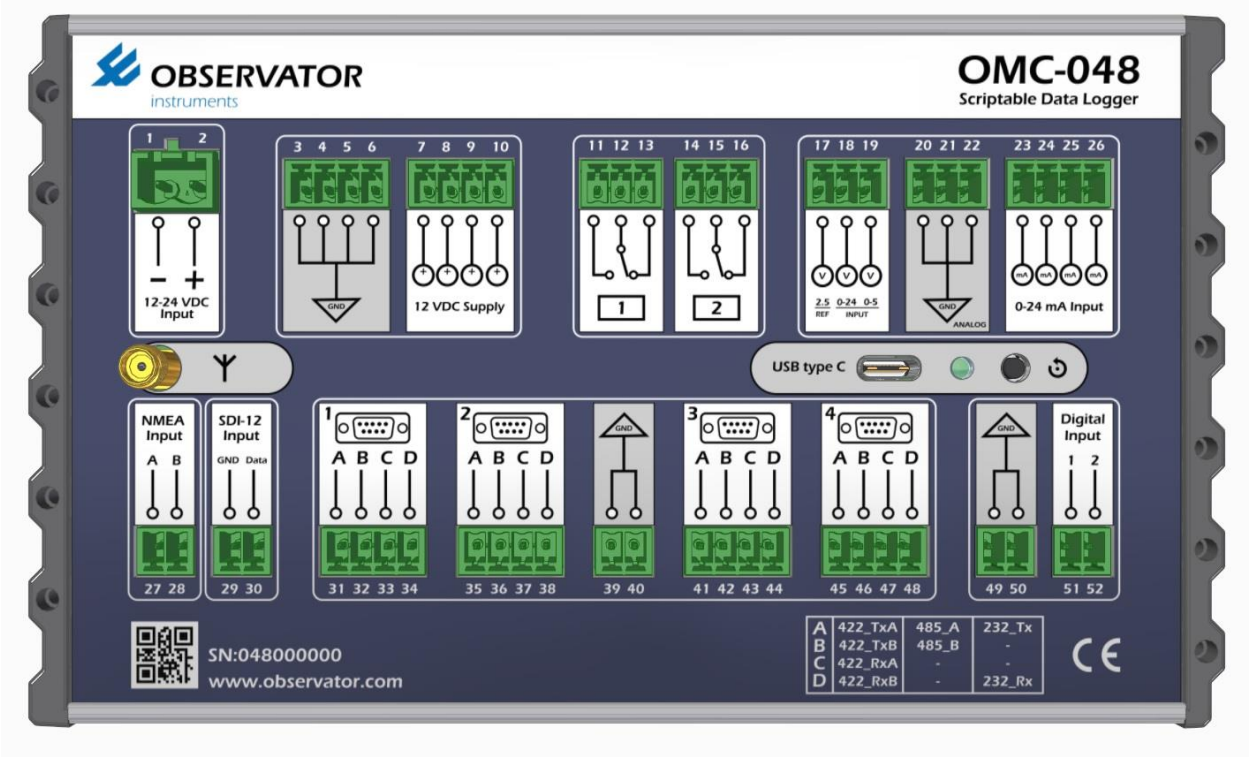
When you receive your OMC-048 Scriptable Data Logger, it might or might not already have been configured by Observator Instruments. For the following steps we assume it has not.

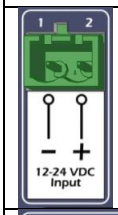
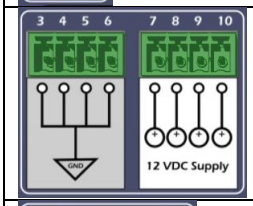
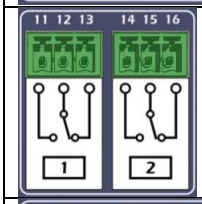
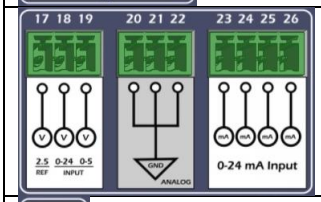
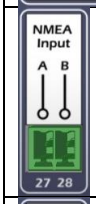
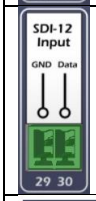
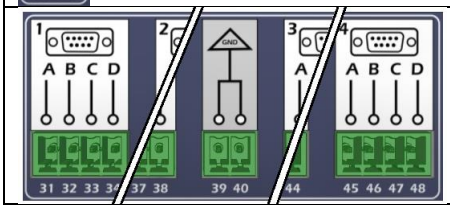
1. Use a suitable Philips screwdriver to remove the 2 screws of the right cover.
2. Remove the right cover
3. Remove plastic strip from coin cell (if not already removed)

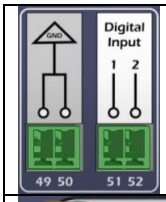
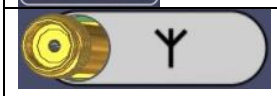


4. Place SIM card (without pin code, check and disable with a phone).
5. Reinstall cover
6. Tighten the 2 screws.

3 I/O Connections



Connector	Description	Remarks
	Power supply input 12- 24 VDC	Note: Power input is protected but not internally fused. Use an external fuse
	Switchable power supply 12VDC Max 200mA nominal per output Peak power 500mA (1 port!)	For sensor use.
	Relay outputs Max 1A @ 30V	Note: relays retain their last position after power down.
	Analogue inputs 0-24VDC 0-5VDC 2.5V reference voltage output 0-24mA input (4 pce)	
	NMEA input Optical isolated	Wind sensors Meteolink NMEA type sensors.
	SDI-12 input	
	Serial inputs (4 pce) RS-232 RS-422 RS-485	Each port can be selected for either type.

	Digital input Contact input (2 pce)	
	Antenna connection for GPRS/UMTS/LTE (2.5G/3G/4G).	
	USB (type C) connector Status LED Reset Button	

3.1 Ground connections

- The negative input power connection (terminal 1) is filtered and therefore not equal to ground!
- The analogue ground is decoupled so not equal to ground!
- The SDI-12 ground is dedicated to the SDI 12 input.

.

Basic ground connection rule:

Use the ground (GND) and signal connections within same section (= white lined box) and you will be fine!

4 Connection

4.1 USB connection

The logger can be powered via the USB connection, this is very useful for configuration.
However, your USB port might not supply enough power when sensors are connected and / or the modem is active.

Connect external power before connecting the USB cable in that situation, otherwise you might not be able to connect via USB.

Connect external power (12-24VDC) before connecting the USB cable when sensors are connected or the logger already has been configured for transmitting data!

The OMC-048 Scriptable Data Logger will create 2 connections with your PC:

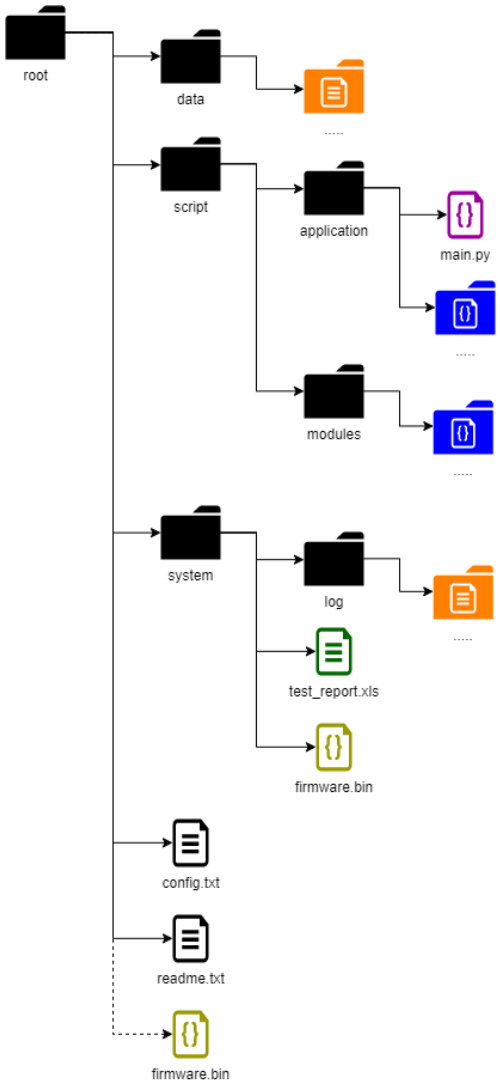
1. Mass Storage device
2. USB Serial device (Com port)

Default the OMC-048 will check for USB connection every 10s. However this is user configurable.
If this check is disabled and the logger is already running, you might have to reset the logger to get an USB connection.

4.2 Mass Storage device

The OMC-048 will act as a USB drive connected to your PC.
This will make it very easy to transfer files.

File structure:



4.2.1 Root

In the root you will find the config.txt, which contain all user settings and a readme.txt file.
The readme.txt file is a helpfile which explains the basics.

4.2.2 Data

In the data folder all logged data will be stored.

4.2.3 Script

The Script folder contains 2 sub folders.

- *Application* contains your main script (main.py)
- *Modules* contains libraries the application requires.

This manual will not go into detail about these files.

4.2.4 System

The system folder contains system log files in the subfolder ’ log’ and a copy of the (previous) installed firmware(s).
The log files can contain important debug information.

4.3 USB Serial Device

The OMC-048 Scriptable Data Logger will also create a comport.
With this connection you can communicate directly with the OMC-048 via a terminal program.

This will allow you to monitor the logger, test Python code, test communication and sensors.

You can use any terminal program. A free tool is available on the Observator website: OMC-Terminal.
All examples assume OMC-terminal is used, but with other terminals will work in a similar way.

4.3.1 Monitoring

When you connect via the (virtual) comport with a terminal program (baud rate etc. settings are not important since it’s usb), you can monitor the actions of the OMC-048 data logger.

It will show:

Sensor Power ON or OFF
Sensor read
Creation of Log (Data & System, including time)
Warnings when actions fail

Example of start-up:

```
2020-08-04 14:02:15 [APPLICATION] INFO Application started: data_bouy
2020-08-04 14:02:15 [APPLICATION] INFO Init NEP5000 Driver 1
2020-08-04 14:02:15 [APPLICATION] INFO Init GPS Driver
2020-08-04 14:02:15 [APPLICATION] INFO NEP5000 parameter detect
2020-08-04 14:02:42 [NEP5000] WARN NEP5000 parameters of sensor: 1 could not be obtained!
2020-08-04 14:02:42 [APPLICATION] INFO GPS parameter detect
2020-08-04 14:02:42 [GPS] INFO Navilock MD6 parameters obtained
2020-08-04 14:02:42 [DATALOG045] INFO Started logging in 'data/OMC-048_Scrilo_Weekend_200804_140242.txt'
2020-08-04 14:02:42 [SCHEDULER] INFO Enabled sleeping in between tasks
2020-08-04 14:02:43 [APPLICATION] INFO pre sleep functions execute
2020-08-04 14:02:43 [SCHEDULER] INFO sleep at 16949758 for 16798, (2020, 8, 4, 2, 14, 2, 43, 251)
2020-08-04 14:02:59 [SCHEDULER] INFO awake at 16966557, (2020, 8, 4, 2, 14, 2, 59, 47)
2020-08-04 14:02:59 [APPLICATION] INFO post sleep functions execute
2020-08-04 14:03:00 [APPLICATION] INFO NEP 1 power on (20 sec startup time)
2020-08-04 14:03:00 [APPLICATION] INFO GPS power on
2020-08-04 14:03:00 [APPLICATION] INFO GPS timeout: 120 seconds
2020-08-04 14:03:21 [APPLICATION] INFO NEP 1 succesfully received data
2020-08-04 14:03:21 [APPLICATION] INFO NEP 1 power off
2020-08-04 14:03:21 [APPLICATION] INFO Initiate nep5000 log
```

4.3.2 Time sync

Time sync can be automated via GPS or time server, but you can also manually adjust the date & time.

For this you will need to go into the Python interpreter, we will not go into detail, but show you step by step how to correct the date & time.

While connected via the terminal type:

CTRL-C (CTRL-SHIFT-C when using OMC-terminal)

The logger should respond with something like:

```
2020/08/04 16:22:50----------------------------------------
Observator instruments OMC-048 Scriptable Data Logger
SN: 49000000 FW: 0.01B653 BL: 1.00B0 uPY: 1.12.0
Application:
Type "help()" for more information.
------------------------------------------------------------
```

Type:

```
import omc048
rtc = omc048.RTC()
rtc.datetime()
```

response of logger (example):

```
(2020, 8, 4, 2, 16, 23, 15, 103)

(Year, Month, Day, Weekday*, Hour, Minute, Second, Sub second)
```

* Weekday: 1 = Monday – 7 = Sunday

Correct the date & time, in this example 4th of August 2020 16:29:00:

Type:

```
rtc.datetime((2020,8,4,2,16,29,0,0))
```

Mind the double () !

```
Check if the update went ok type:

rtc.datetime()

Reponse of logger:

(2020, 8, 4, 2, 16, 29, 3, 217)
```

If this is the correct date and time you're done otherwise repeat above actions.

Return to normal operation with soft-reset:

CTRL-D

5 Software

5.1 Application

The application folder contains the program (script) of your application.
The program file main.py written in Python controls the logger. It might require specific libraries.
Within the application settings can be changed, they are stored in the configuration file.
The configuration file can be easily adapted by the user to change for example intervals of measurement, storage or transmission. No programming skills are required.

The application does require Python programming skills and is not explained in this manual.

5.2 Configuration (config.txt)

The config.txt file contains all parameters which can be changed by the user.
It is located in the root of the sd card. Without a valid config.txt file the python program will not run and switch to the REPL mode.

It is written in YAML, what is YAML (quotes from yaml.org):

- 'YAML: YAML Ain't Markup Language'
- 'YAML is a human friendly data serialization standard for all programming languages.'

Please refer to the yaml.org site for detailed information.

An example config will be included with each firmware update, if you make changes (especially in the field) we recommend you make a copy of the file first, so you can always go back.

You will find an example is on the next page.

A hashtag ('#') in front of text makes it just text. You can also use this to disable a setting

Actually any character in front will disable the setting, but for readability and consistency we do recommend to use the hashtag!

For example:

To disable the FTP driver in this config simply change Ftp: to #Ftp:

Always use the config.txt file included with the installed firmware as reference!

All names of drivers, parameters, settings etc. must be spelled correct, a typo will invalid the line. Luckily you will be notified with clear error messages when you start the application in that case and end up in the REPL.

5.2.1 General settings

The first line of each driver contains the name of the driver followed by a colon (':').
(in the previous example NEP5000:)

The next line always the id:. This name tag will be send with the corresponding parameters and is free to choose, however we do recommend to keep it default especially when using OMC-DOL.
Most drivers will have a port setting to define the connected i/o port and interval setting.

Other settings are driver specific, check the example config and OMC-048 web manual of the corresponding firmware for details.

5.2.2 Example config.txt

```
# ----System---- #
Omc048:
  system_id: Scrilo
  application: test
  file_log_level: info
  repl_log_level: info
  utc_time_offset_hours: +1
  utc_time_offset_minutes: +0

Usb_wake:
- id: usb_control
  wake_interval: "0,10,20,30,40,50 * * * *"

# ----Modem-Settings---- #
Modem:
  id: onboard_modem
  port: modem
  sim_username: KPN
  sim_password: gprs
  apn: internet

# ----FTP-Settings---- #
Ftp:
- id: data_ftp
  url: ftp.omc-data-online.com
  port: 21
  username: OMC-test
  password: omc-test
  transmit_interval: "30 0,5,10,15,20,25,30,35,40,45,50,55 * * *"

# ----Sensor-Settings---- #
Gps:
- id: GPS_A
  port: serial1
  mode: RS232
```

```
  baudrate: 9600
  supply_port: 4
  sample_interval: "0 * * * *"
  response_timeout: 180
  utc_time_sync: True

ysi_6_series:
- id: 600OMS
  port: serial2
  sample_interval: "0 * * * *"
  wiper_interval: "0 * * * *"
  supply_port: 1
  timeout: 10
  supply_port_always_on: False

Onboard:
- id: board_sensors0
  sample_interval: "0 * * * *"

Analog_voltage:
- id: analog_voltage1
  port: 1
  sample_interval: "0 * * * *"
  min_in: 0
  max_in: 24000
  min_out: 0
  max_out: 24
  log_name: Voltage1
  log_unit: V
  log_tag: VOLT1

# ----Data-log-settings---- #
Data_file:
- id: data
  create_interval: "0 0,5,10,15,20,25,30,35,40,45,50,55 * * *"

Log_parameters:
  GPS_A:
  - Latitude
  - Longitude
  board_sensors0:
  - Temperature
  - Humidity
  - Coin cell voltage
```

The config file in this chapter is for example use only and may not work correct with the firmware installed on your OMC-048 datalogger.

Again: always use the config file included with the installed firmware as reference!

5.2.3 System

ID of the system & substation and name of the application.

In this example:

System_id	= Scrilo	will be included in the logfile name
Application	= test	for your reference only
file_log_level: info		level of stored information in system log file
repl_log_level: info		level of displayed information in REPL

for both file and repl log_level ‘info’ is default, possible levels (maximum data to minimal / off) are:

debug < info < warn < error < fatal < off

utc_time_offset_hours: +1	Offset to UTC in hours
utc_time_offset_minutes: +0	Offset to UTC in minutes

utc_time_pffset is used when sync to gps or timeserver is active and you want to use local time for example (time sync is with utc time).

Usb_wake:	USB connection check routine.
- id: usb_control	
wake_interval: "0,10,20,30,40,50 * * * *"	every 10s check USB, see chapter Interval settings for details.

5.2.4 Modem Settings

Selection of the modem and the Provider login details (APN).

ID = onboard_modem
Port = modem (do not change)
sim_username: KPN
sim_password: gprs
apn: internet

Change the username, password & apn to the ones required for your provider.

5.2.5 FTP-settings

id = data_ftp
url = ftp server address
port = portnumber of the ftp-server
username = username of ftp server
password = password of ftp server
transmit_interval: transmit time schedule, see chapter Interval settings for details.

5.2.6 Sensor Settings

Example Sensor config

Gps:
- id: GPS_A tag id of sensor

port: serial1	Used port
mode: RS232	mode of serial port (RS232, RS422 or RS485)
baudrate: 9600	Baudrate setting
supply_port: 4	Used power-switch (1 – 4) for sensor power
sample_interval: "0 * * * *"	see chapter interval settings for details.
response_timeout: 180	wait max 180s for data
utc_time_sync: True	True = use GPS for time sync (False = do not)

onboard: Onboard sensors (Temperature, Humidity & Coin cell voltage)
- id: board_sensors
  sample_interval: "0 0,10,20,30,40,50 * *" see chapter Interval settings for details.

5.2.7 Data_log settings

Data_file:
- id: data
  create_interval: "0 0,5,10,15,20,25,30,35,40,45,50,55 * * *"

Set the interval on which time a data file is created (see chapter Interval settings for details).

Note especially for (former) OMC-045 users:

The OMC-048 behaves different from the OMC-045(x) when it comes to sample and storage rate!

The data_file create_interval only sets the interval time on which a new file is created.
The sample_interval of each sensor sets the number off samples taken and stored!

5.2.8 Interval settings

The intervals for samples, storage and transmission can be configured in the similar way as cronjobs (Cron is a time schedular for jobs in Unix based software languages) are configured. A interval description is built up out of 5 fields separated by white spaces that represent a set of times at which to execute the task. The function runs when the current date matches the interval description. An interval description looks like this:

```
┌──────────── second (0 - 59)
│ ┌────────── minute (0 - 59)
│ │ ┌──────── hour (0 - 23)
│ │ │ ┌────── day of the month (1 - 31)
│ │ │ │ ┌──── day of the week (1 - 7) (Monday to Sunday)
│ │ │ │ │
* * * * *
```

Special characters can be used to select a range or multiple values for a field. The following special characters are allowed.

* Asterisk - The asterisk is used to select all allowed values for that field.

, Comma - The comma defines a list of values. For example, 0,14,29,44 in the first field would mean seconds 0, 14, 29, and 44.

- Dash - The dash defines a range of values. For example, 5-10 indicates all minutes between 5 and 10, inclusive. (5, 6, 7, 8, 9, 10)

Example intervals:

Expression	**Meaning**
* * * * *	Every second
0 * * * *	Every minute at hh:mm:00
2 * * * *	Every minute at hh:mm:02
0 0 * * *	Every hour at hh:00:00
5 1 * * *	Every hour at hh:01:05
0 0 0 * *	Every day at 00:00:00
0,29 * * * *	Every 30 seconds
0 0,10,20,30,40,50 * * *	Every 10 minutes
0 30 9 * 7	Every Sunday at 9:30:00
0 0 12 1 *	Every first day of the month at 12:00:00
0 0 9-17 * 1-6	Hourly between 9:00:00 and 17:00:00 except for Sunday

An Important note concerning Interval settings:

All intervals are time based. This does offer great flexibility and control, but also forces you to choose the intervals wisely:
Setting all intervals identical for samples, storage and transmission for example will result in transmission of data from a previous cycle (the actual data won't be available yet, since transmitting and sampling will start at the same moment).
This might not be an issue when you transmit frequently, but might not be what you expect when transmitting is set to once a day.

Set the FTP scheduler delayed with enough time to get a sample if you want the most recent data transmitted!

6 Firmware Update

6.1 Introduction

Observator products are in continuous development during their product lifetime.
Official firmware updates are published on the Observator website.

We recommend you only update your firmware when advised by Observator Instruments.

Note that firmware updates may change the behavior of your data logger and sometimes the programming might need adjustment!

6.2 Firmware update

If you do need to update:

1. Download the latest version from the support page of the Observator.com webpage.
2. Connect your OMC-048 Data logger to your PC using the USB cable
3. The OMC-048 data logger will appear as a mass storage device
4. Copy the firmware file to the root of the OMC-048
5. Reset the OMC-048 (press reset button)
6. The OMC-048 will restart and during startup check for a valid Firmware file in the root. When found it will update and move the firmware file to the system folder. This way you can confirm the update has been performed, but you will also find the previous firmware versions in this folder. If for some reason you would like to go back to a previous version, simply move the desired version to the root and restart the OMC-048 (press reset button).

Appendix A: Specifications:

Power

- Supply voltage nominal 12 Vdc to 24 Vdc (min 9Vdc, max 32Vdc).
- Power consumption in sleep mode: <1 mA at 12 V.
- Power consumption in active mode: <30 mA at 12 V (excluding modem and power outputs).
- Peak power consumption (including modem): 250 mA at 12 V (excluding power outputs).
- Max power output per switched power output: 0.5 A peak (one output), 200 mA continuously (4 outputs each).

Note: Switched power output is maximized to 12V for supply voltages above 12V.

Environmental

- Temperature: -25 ... +70 °C.
- Humidity: 10 - 90% RH, non-condensing.
- Enclosure: IP40.

Dimensions

- Width x depth x height: 177 x 105,5 x 50 mm.
- Weight: 410 g.
- Package dimensions: t.b.d.
- Package weight: t.b.d.

SIM & SD cards

- Micro SIM.
- Micro SD (Max 32GB)

Appendix B: Declaration of Conformity

OBSERVATOR
instruments

Observator Instruments B.V.

Rietdekkerstraat 6
2984 BM Ridderkerk
The Netherlands

P.O. Box 60
2980 AB Ridderkerk
The Netherlands

Tel.: +31 (0)180 463411
Fax.: +31 (0)180 463530

Email: info@observator.com
Internet: www.observator.com
CoC: 24172722

EU DECLARATION OF CONFORMITY

(1) Apparatus model : **OMC-048**

(2) Manufacturer :
Observator instruments B.V.
Rietdekkerstraat 6
2984 BM Ridderkerk
The Netherlands

(3) This declaration of conformity is issued under the sole responsibility of the manufacturer.

(4) Object of the declaration :

OMC-048 Script Data Logger

(5) The object of the declaration described above is in conformity with the relevant Union harmonisation legislation:
- Directive 2014/30/EU of the European Parliament and of the Council of 26 February 2014 on the harmonisation of the laws of the Member States relating to electromagnetic compatibility
- Directive 2011/65/EU of the European Parliament and of the Council of 8 June 2011 on the restriction of the use of certain hazardous substances in electrical and electronic equipment

(6) References to the relevant harmonised standards used:

EN IEC 60945:2002 +C1 EN IEC 61326-1:2013
EN 50581:2012

(7) -

(8) Signed for and on behalf of:

Ridderkerk, 01 November 2019,

Observator Instruments

Dr. Ir. R. de Vries
CEO

SGS

Since 1924 Observator has evolved to be a trend-setting developer and supplier in a wide variety of industries. Originating from the Netherlands, Observator has grown into an internationally oriented company with a worldwide distribution network and offices in Australia, Germany, the Netherlands, Singapore and the United Kingdom.