23 Data Export

General

Various protocols are made to transmit data points as they are generated. This is enough for a lot of systems (e.g. SCADA) that have their databases and devices only have to buffer fairly recent messages in case of connection or transmission errors. However, there is frequently a need to save and keep the data in files, grouped in batches, and later transmit these batches to a remote server via HTTP(S) or FTP(S). For this purpose, a dedicated protocol has been created and called **Data export.**

> ✅ Data export functionality is available since firmware version v1.5.0, of WCC Lite.

Overview

Data export service gathers information from other protocols and puts it into files (optionally compressing them) after a timeout or when data buffers fill up; eventually periodically sending them to a server. HTTP(S) and FTP(S) servers with optional authentication are supported. A user can optionally choose between ISO8601 and UNIX timestamp time formats (the latter being the default value). More than one instance can be set up, for instance, some of the information can be sent to an FTP server, while others could be transmitted to the HTTP server which can handle POST requests.

Using WCC Lite for data export

To configure WCC Lite to use a data export server a user can fill in the needed parameters in Excel configuration. These parameters are shown in the two tables below. Default values are shown in bold font.

Data export (data-export) parameters for Devices tab table:

Parameter	Type	Description	Required	Default value (when not specified)	Range	
					Min	Max
name	string	User-friendly device name	Yes			
device_alias	string	Device alias to be used in configuration	Yes			
enable	boolean	Enabling/disabling of a device	No	1	0	1
protocol	string	Selection of protocol	Yes		Data Export	
timeout	integer	The time frame during which transmission to the remote server has to be completed (**in seconds**)	No	5		
type	string	Selection of file format	No	csv-simple	cvs-periodic csv-simple, json-simple, json-samples	
host	string	A URL of the remote server where files should be sent	Yes			
upload_rate_sec	integer	Frequency of generated file uploads (**in seconds)**	No	60		

Parameter	Type	Description	Required	Default value	Min	Max
records_buffer_size	integer	A maximum amount of data change entries to hold before initiating the logging mechanism	No	100		
logging_period_sec	integer	Describe how frequently the data buffer of records_buffer_size is saved to the file	No	10	1	3600
log_folder	string	A folder in the WCC Lite file system to save generated files (**"/var/cache/data-export"**)	No			
timestamp	string	Selection of time format	No	unixtimestamp	unixtimestamp, iso8601	
compress	string	Selection of file compression mechanism	No	none	none, gz, tar.gz	
compress_password	string	Enable the feature of file password protection	No		yes, true	
csv_field_separator	string	Columns separator in .csv file format	No	"," - (comma)	"," - (comma) ";" - (semicolon) "." - (dot) " " - whitespace) "|" - (pipe)	
csv_decimal_separator	string	Decimal separator in values	No	"." - (dot)	"." - (dot) "," - (comma)	

> ⓘ The same symbols cannot be selected for both csv_field_separator and csv_decimal_separator. In such case both of them will be set to default values ".." and "," respectively.

The data generation rate may be bigger than what the data buffer can hold (controlled by *records_buffer_size* and *logging_period_sec*). To make sure that no data loss occurs there's an additional data logging call made in case the data buffer reaches a *records_buffer_size* value.

Signals to be sent are configured differently than signals for most other protocols. As data export service only transmits signals and does no data processing, usual signal logic is not used for them. That means that:

• Signals for data export service are not seen in the *Imported Signals* tab;
• Signals for data export service are configured in a different Excel sheet called DataExport

The parameters to be filled in the DataExport sheet are shown in the table below.

Data export (data-export) parameters for the DataExport tab

Parameter	Type	Description	Required	Default value (when not specified)	Range	
					Min	Max
device_alias	string	Device alias to be used in configuration Yes	Yes			
device_name	string	User-friendly device name as in the Device sheet	Yes			
tag_name	string	User-friendly signal name	Yes			
source_device_alias	string	device_alias of a source device	Yes			
source_signal_alias	string	source_alias of a source signal	Yes			

enable	boolean	Enabling/disabling of a measurement to be transmitted and logged	No	1	0	1
attribute	string	An additional attribute to be attached to a signal	No			

Debugging data export service

If the configuration for the Data export service is set up, a handler for the protocol will start automatically. If the configuration is missing parameters or contains errors, the protocol will not start. It is done intentionally to decrease unnecessary memory usage.

Data export (data-export) command-line debugging options

> ℹ **The below-described parameters for debugging are accessible over the console (SSH).**

`-h [--help] Display help information`

`-c [--config] Configuration file location`

`-V [ —version ] Show version`

`-d<debug level> [ —debug ] Set debugging level`

`-R [ —readyfile] Ready notification file`

`-p [ —http ] Show HTTP messages`

`-r [ —redis ] Show Redis output`

If the Data export service does not work properly (e.g. data is corrupted, etc.), a user can launch a debug session from the command line interface and find out why it is not functioning properly. To launch a debugging session, a user should stop data-export processes and run the data-export command with respective flags as in the table above.

Host URL format rules

The host parameter is highly configurable and might contain a considerable amount of information:

• *Protocol* - FTP or HTTP (encrypted and encrypted);

• *URL address* - both resolved and non-resolved;

• *Authentication* - pair of users and/or passwords;

• *Port* - useful when non-standard value is used;

• *Endpoint* - a place in the server to which a call is made

The format for the host parameter can be summarized as:

```
[ h t t p ( s ) / f t p ( s ) ] : / / [ [ u s e r ] : [ p a s s w o r d ]@] [ URL ] [ : p o r t ] / [ e n d p o i n
]
```

Options are printed in square brackets. A protocol has to be selected, otherwise HTTP will be used as a default. The user and password pair is optional, but if the user: password pair is used, it should proceed with the @ sign.

HTTP and FTP use default or user-assigned ports. By default HTTP uses port 80, while HTTPS uses port 443, FTP sends data over port 21, FTPS - over port 990. Make sure that these ports are open in the firewall on both the server and client side, otherwise, data will not be sent successfully.

Finally, a POST request (for HTTP) or upload (for FTP) can be made to a specific place (endpoint). This endpoint should be described after a URL and port (if used).

Format of exported data

For a server to interpret data, a set of rules for a file format has to be established.

Csv-simple format applies to all files by default and is used as in this example:

```
###DUID:3182121xx
#device name; tag name; value; quality; timestamp; atribute
inv1;Ia;236.9,1;1599137189963;Pa
```

Example of additional format *csv-periodic:*

```
###DUID:318212xxx
##DEVICE:inv1
#Time;Upv1;Upv2;Upv3;Upv4;Upv5;Upv6;Ipv1;Ipv2;Ipv3;Ipv4;Ipv5;Ipv6;Status;Error;Temp;cos;fac;Pac;Qac;Eac;E-Day;E-
Total;Cycle Time
2020-09-
02T15:45:00Z;462.3;462.3;370.2;370.2;371.2;371.2;1.40;1.43;1.35;1.47;1.21;1.26;512;0;26.3;1.000;50.00;3.217;-
0.029;0.28;17.41;54284.53;5;
2020-09-
02T15:40:00Z;462.3;462.3;370.2;370.2;371.2;371.2;1.40;1.43;1.35;1.47;1.21;1.26;;512;0;26.3;1.000;50.00;3.217;-
0.029;0.28;17.41;54284.53;5;
##DEVICE:meter
#Time;Uab;Ubc;Uca;P;Q;S;F;eTOT;Cycle Time
2020-09-02T15:45:00Z;421.3;421.3;421.3;15000;100;15600;50;246894;5;
2020-09-02T15:40:00Z;421.3;421.3;421.3;15000;100;15600;50;246895;5;
```

Example of additional format json-simple*:*

```
{
  "metadata": {
    "duid": "318xxxxx",
    "name": "hostname",
    "loggingPeriod": "15min",
    "format": "json"
  },
  "data": [
    {
      "tag_name": "Ia",
      "device_name": "inv1",
      "attribute": "Pa",
      "last": { "value": 12.2, "timestamp": 1213123 },
      "min": { "value": 12, "timestamp": 1213123 },
      "max": { "value": 12, "timestamp": 1213123 },
      "avg": { "value": 12, "timestamp": 1213123 }
    },
    {
      "tagName": "Ib",
      "deviceName": "inv1",
      "attribute": "Pb",
      "last": { "value": -12.3, "timestamp": 1213123 },
      "min": { "value": 12, "timestamp": 1213123 },
      "max": { "value": 12, "timestamp": 1213123 },
      "avg": { "value": 12, "timestamp": 1213123 }
    },
  ]
}
```

Example of additional format json-sample*:*

```
{
    "metadata": {
        "duid": "318xxxxx",
        "name": "hostname",
        "loggingPeriod": "15min",
        "format": "json-samples"
```

```
      },
      "data": [
        {
          "tag_name": "Ia",
          "device_name": "inv1",
          "attribute": "Pa",
          "timestamp":{
              "first": 123123,
              "last": 123236
          },
          "first": { "value": 12.2, "timestamp": 1213123 },
          "last": { "value": 12.2, "timestamp": 1213123 },
          "min": { "value": -12, "timestamp": 1213123 },
          "max": { "value": 12, "timestamp": 1213123 },
          "avg": { "value": 12, "timestamp": 1213123 },
          "samplesCount": 2,
          "samples": [
              {"value": 12, "timestamp": 1213123, "quality": true},
              {"value": -12.3, "timestamp": 1213123, "quality": true}
          ]
        }
      ]
    }
```

```
      },
      "data": [
        {
          "tag_name": "Ia",
          "device_name": "inv1",
          "attribute": "Pa",
          "timestamp":{
              "first": 123123,
              "last": 123236
          },
          "first": { "value": 12.2, "timestamp": 1213123 },
          "last": { "value": 12.2, "timestamp": 1213123 },
```