Data formats OTT ecoLog 1000

The OTT ecoLog 1000 supports various data formats for local readout as well as IP transmission, details and explanations of the individual data formats are summarized in this document.

Content:

1 OTTML Data Format 2

2 OTT MIS Format 4

3 CSV Format..... 5

4 ecoLog 1000 integration in Aquarius Time Series 7

5 ZRXP data exchange format (V3.0) 9

6 ZRXP data exchange format (legacy)..... 10

7 SMS data push..... 11

8 Data and Station Identifiers - UUID 11

9 IP test transmission 12

10 Test SMS..... 13

11 MQTT Support..... 14

1 OTTML Data Format

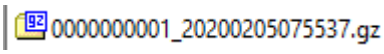
The OTTML data format is based on XML format (extensible markup language). The format is easily readable with a standard text editor and is a used standard format for exchanging data between different application, platform independent.
OTTML provides extensive meta data of the system besides the measured data. The expression “data” is very generic and comprises the actual measured values, as well as additional information, like events that occurred in the logger or metadata, like current firmware version of the logger.

Example:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<StationDataList>
 <StationData stationId="0000000001" name="OTT Formats" timezone="+01:00"
ref="020C0B3340547EE594A0B22A12"  >
  <StationInfo time="2020-02-04T15:10:56" firmware="V1004" configtime="2020-01-30T20:53:05"
paramtime="2020-02-04T14:59:59" batteryVoltage="3.60" batteryMin="3.48" usedBattery="1708.5"
temperature="25.30" humidity="30"  lon="10.322513" lat="47.711428" deviceType="OTT ecolog 1000"
providerName="Vodafone.de" gsmSignal="4" ipAddress="100.89.223.243" transmissionCycle="300"
transmissionOffset="0" configuredTransmissionCycle="300" />
  <ChannelData channelId="0001" name="Distance to Water" unit="m" samplingInterval="60"
configuredSamplingInterval="60" storageInterval="60" configuredStorageInterval="60"  >
  <Values>
   <VT t="2020-02-04T15:05:00">2.348</VT>
   <VT t="2020-02-04T15:06:00">2.348</VT>
   <VT t="2020-02-04T15:07:00">2.348</VT>
  </Values>
  </ChannelData>
  <ChannelData channelId="0002" name="Water Temperature" unit="&#176;C" samplingInterval="60"
configuredSamplingInterval="60" storageInterval="60" configuredStorageInterval="60"  >
  <Values>
   <VT t="2020-02-04T15:05:00">24.1</VT>
   <VT t="2020-02-04T15:06:00">24.1</VT>
   <VT t="2020-02-04T15:07:00">24.1</VT>
  </Values>
  </ChannelData>
  <ChannelData channelId="0006" name="Processed Value" unit="m" samplingInterval="60"
configuredSamplingInterval="60" storageInterval="60" configuredStorageInterval="60"  >
  <Values>
   <VT t="2020-02-04T15:05:00">714.348</VT>
   <VT t="2020-02-04T15:06:00">714.348</VT>
   <VT t="2020-02-04T15:07:00">714.348</VT>
  </Values>
  </ChannelData>
  <ChannelData channelId="PBAT" name="Power Consumption" unit="mAh" samplingInterval="900"
configuredSamplingInterval="900" storageInterval="900" configuredStorageInterval="900"  >
  <Values>
   <VT t="2020-02-04T15:00:00">1707</VT>
  </Values>
  </ChannelData>
  <ChannelData channelId="RSSI" name="Signal Strength" unit="" samplingInterval="900"
configuredSamplingInterval="900" storageInterval="900" configuredStorageInterval="900"  >
  <Values>
   <VT t="2020-02-04T15:00:00">4</VT>
```

```
  </Values>
  </ChannelData>
  <ChannelData channelId="UBAT" name="Supply Voltage" unit="V" samplingInterval="900"
configuredSamplingInterval="900" storageInterval="900" configuredStorageInterval="900"  >
  <Values>
   <VT t="2020-02-04T15:00:00">3.70</VT>
  </Values>
  </ChannelData>
 </StationData>
</StationDataList>
```

To reduce the size of the OTTML file it is possible to zip it in the ecoLog 1000 and transmit it as zipped file.



The receiving server needs to unzip the data, OTT Hydromet software like Hydras 3 does this automatically.

2 OTT MIS Format

The OTT MIS data format is an ASCII format. The format is easily readable with a standard text editor. The format contains the measurement data of the system without meta data.
Every sensor within the file has its own block of data, which starts with a header line, which specifies the station and sensor, to which the data belong. The header line is followed by the data lines, where every line consists exactly of the date, the time and the corresponding measured value.

Example:

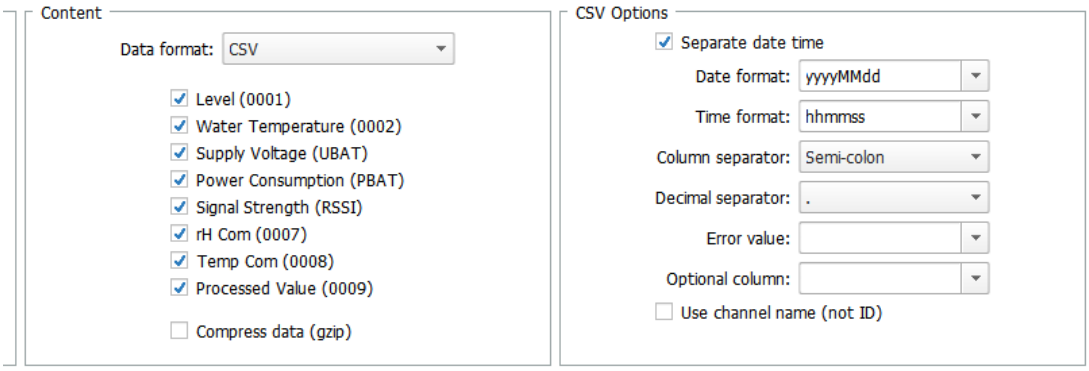
<STATION>0000000001</STATION><SENSOR>0001</SENSOR><DATEFORMAT>YYYYMMDD</DATEFORMAT>
20200204;145000;2.348
20200204;145200;2.348
20200204;145400;2.348
<STATION>0000000001</STATION><SENSOR>0002</SENSOR><DATEFORMAT>YYYYMMDD</DATEFORMAT>
20200204;145000;24.1
20200204;145200;24.1
20200204;145400;24.1
<STATION>0000000001</STATION><SENSOR>0006</SENSOR><DATEFORMAT>YYYYMMDD</DATEFORMAT>
20200204;145000;2.348
20200204;145200;2.348
20200204;145400;2.348
<STATION>0000000001</STATION><SENSOR>PBAT</SENSOR><DATEFORMAT>YYYYMMDD</DATEFORMAT>
20200204;144500;1702
<STATION>0000000001</STATION><SENSOR>RSSI</SENSOR><DATEFORMAT>YYYYMMDD</DATEFORMAT>
20200204;144500;8
<STATION>0000000001</STATION><SENSOR>UBAT</SENSOR><DATEFORMAT>YYYYMMDD</DATEFORMAT>
20200204;144500;3.71

3 CSV Format

The file format CSV stands for Comma-Separated-Values and describes the structure of a text file for storing or exchanging simply structured data. The file name extension is *.csv.
There is no general standard for the CSV file format, but it is described in RFC 4180. The character encoding to be used is also not specified.

The ecoLog 1000 support various possibilities to configure via LinkComm the csv data format in rows or columns as well as the csv options.

Example for csv – row:

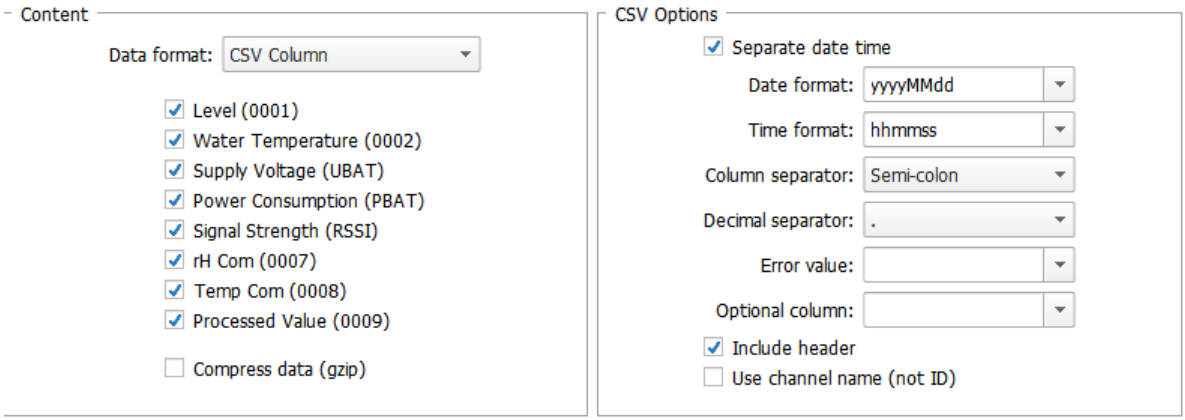


File format

Station ID; Channel ID; Date; Time; Value

0000000001;0001;20200204;145000;2.348
0000000001;0001;20200204;145100;2.348
0000000001;0001;20200204;145200;2.348
0000000001;0001;20200204;145300;2.348
0000000001;0001;20200204;145400;2.348
0000000001;0001;20200204;145500;2.348
0000000001;0002;20200204;145000;24.1
0000000001;0002;20200204;145200;24.1
0000000001;0002;20200204;145400;24.1
0000000001;0006;20200204;145000;2.348
0000000001;0006;20200204;145200;2.348
0000000001;0006;20200204;145400;2.348
0000000001;PBAT;20200204;144500;1702
0000000001;RSSI;20200204;144500;8
0000000001;UBAT;20200204;144500;3.71

Example for csv – column:

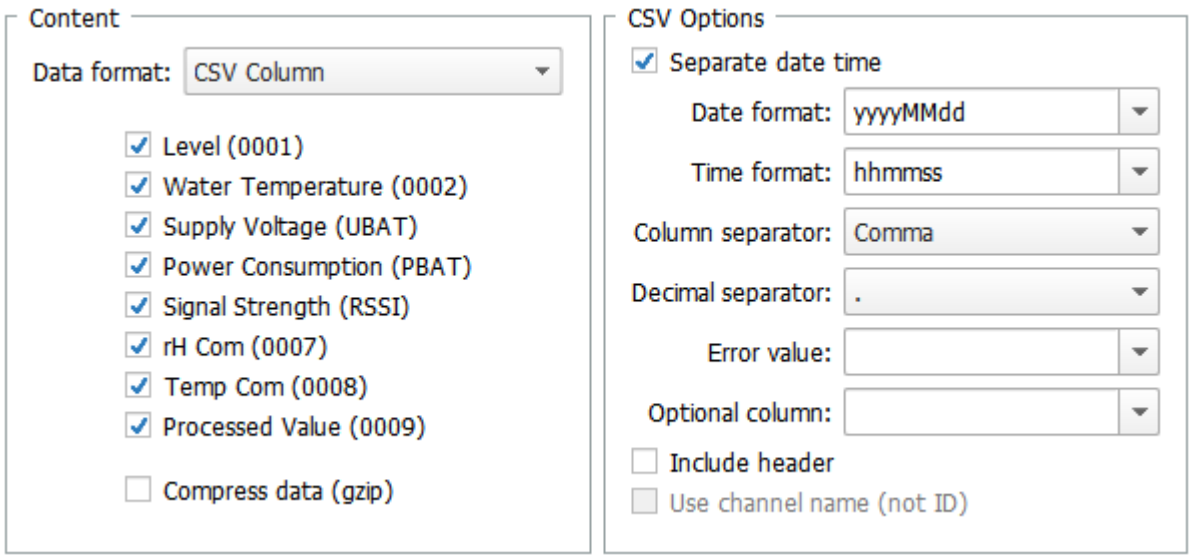


File format

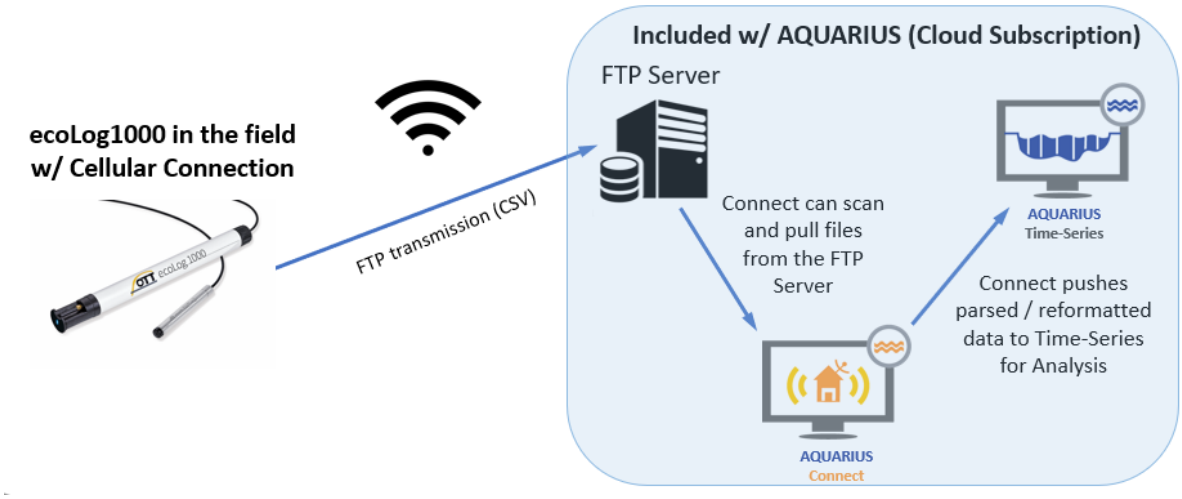
```
Date;Time;0001;0002;0007;0008;0009;PBAT;RSSI;UBAT
20201211;084500;15.1;18.47;;;0.151;12556;6;3.69
20201211;084600;;18.48;;;0.151;;;
20201211;084700;;18.48;;;0.151;;;
20201211;084800;;18.48;;;0.151;;;
20201211;084900;;18.49;;;0.151;;;
20201211;085000;15.1;18.49;;;0.151;;;
20201211;085100;;18.50;;;0.151;;;
20201211;085200;;18.50;;;0.151;;;
20201211;085300;;18.49;;;0.151;;;
20201211;085400;;18.49;;;0.151;;;
20201211;085500;15.1;18.49;;;0.151;;;
20201211;085600;;18.49;;;0.151;;;
20201211;085700;;18.49;;;0.151;;;
20201211;085800;;18.49;;;0.151;;;
20201211;085900;;18.50;;;0.151;;;
20201211;090000;15.1;18.49;41;21.2;0.151;12556;6;3.71
```

4 ecoLog 1000 integration in Aquarius Time-Series

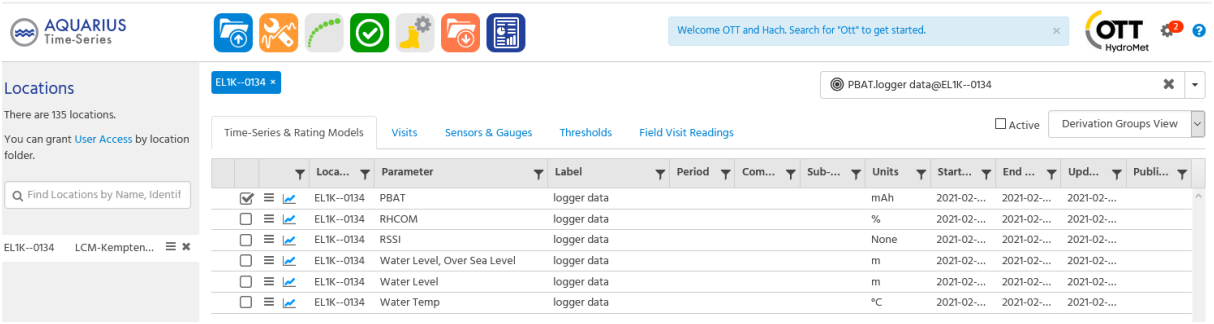
The OTT ecoLog 1000 allows to transmit data to Aquarius Software and visulaize the data in Aquarius Time-Series.
The used data format is the csv format:

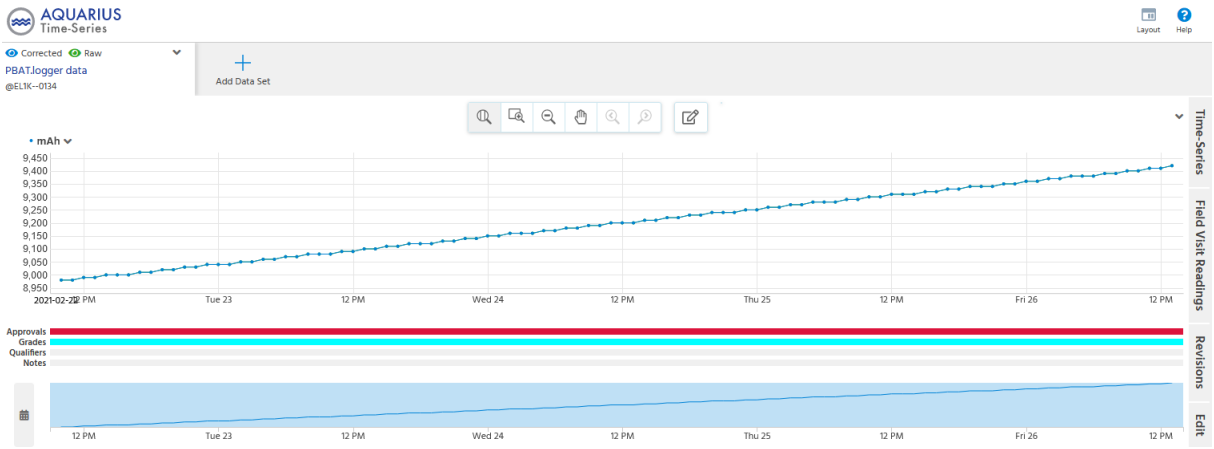


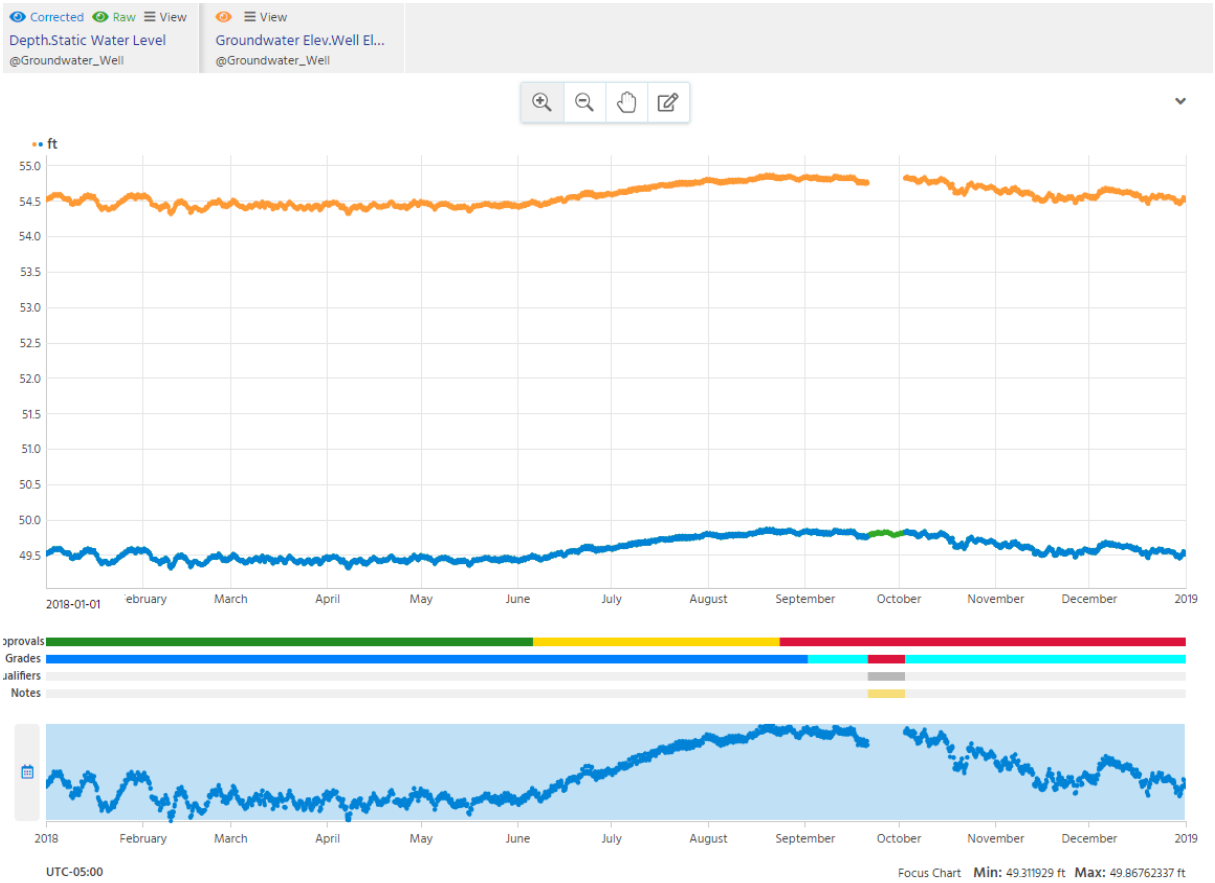
Data is sent to an Aquarius FTP server and from there pulled into Aquarius Time-Series via Aquarius Connect:



Aquarius Time-Series visulaizes the data:







5 ZRXP data exchange format (V3.0)

The data format ZRXP is a line-oriented text file format having ISO-8859-1 encoding. A file in ZRXP format consists of one or several segments (blocks) with each segment being divided into a basic data header and a time series value block.
Each segment always begins with a basic data header. At least one block with time series value(s) must follow the basic data header. After each block the file can end, or a further segment can follow. Empty lines and comments are ignored; they can stand in any place in the file.
With the implementation of ZRXP 3.0 in the OTT ecoLog 1000 it is possible to add to every measured value a so called REXCHANGE value/string. This allows to import the zrxp without modification into a Kisters Wiski 7 data base (bypassing a Kisters SODA) Use case: push ZRXP V3.0 directly via IP to a HTTP/HTTPS Wiksi 7 Server. Alternatively, it could be sent to an FTP server and then directly imported into the Wiski 7 data base.

Example:

```
#ZRXPVERSION3014.03|*|ZRXPCREATORKiIOSystem.Manual|*|
#REXCHANGE1131802900064.W_Gw.Dasa1.O|*|
#SANR0000000001|*|
#CDASA1|*|
#CDASANAMEOTT Formats|*|
#CCHANNELNO1|*|CUNITm|*|
#RINVAL-777|*|
#LAYOUT(timestamp,value)|*|
20200204154000 2.348
20200204154100 2.348
20200204154200 2.348
#ZRXPVERSION3014.03|*|ZRXPCREATORKiIOSystem.Manual|*|
#REXCHANGE|*|
#SANR0000000001|*|
#CDASA1|*|
#CDASANAMEOTT Formats|*|
#CCHANNELNO2|*|CUNIT°C|*|
#RINVAL-777|*|
#LAYOUT(timestamp,value)|*|
20200204154000 24.1
20200204154100 24.1
20200204154200 24.1
#ZRXPVERSION3014.03|*|ZRXPCREATORKiIOSystem.Manual|*|
#REXCHANGE|*|
#SANR0000000001|*|
#CDASA1|*|
#CDASANAMEOTT Formats|*|
#CCHANNELNO6|*|CUNITm|*|
#RINVAL-777|*|
#LAYOUT(timestamp,value)|*|
20200204154000 714.348
20200204154100 714.348
20200204154200 714.348
#ZRXPVERSION3014.03|*|ZRXPCREATORKiIOSystem.Manual|*|
#REXCHANGE|*|
#SANR0000000001|*|
#CDASA1|*|
```

```
#CDASANAMEOTT Formats|*|
#CCHANNELNOPBAT|*|CUNITmAh|*|
#RINVAL-777|*|
#LAYOUT(timestamp,value)|*|
20200204153000 1716
20200204154500 1719
#ZRXPVERSION3014.03|*|ZRXPCREATORKiIOSystem.Manual|*|
#REXCHANGE|*|
#SANR0000000001|*|
#CDASA1|*|
#CDASANAMEOTT Formats|*|
#CCHANNELNORSSI|*|CUNIT|*|
#RINVAL-777|*|
#LAYOUT(timestamp,value)|*|
20200204153000 5
20200204154500 -777
#ZRXPVERSION3014.03|*|ZRXPCREATORKiIOSystem.Manual|*|
#REXCHANGE1131802900064.U_Betrieb.Dasa1.O |*|
#SANR0000000001|*|
#CDASA1|*|
#CDASANAMEOTT Formats|*|
#CCHANNELNOUBAT|*|CUNITV|*|
#RINVAL-777|*|
#LAYOUT(timestamp,value)|*|
20200204153000 3.71
20200204154500 3.73
```

6 ZRXP data exchange format (legacy)

Before the support of V3.0, as mentioned under §4, a different zrxp format was used and supported. This can be can be figured as well.

Example:

```
#SANRBETAUSPETE;*;
#RINVAL-777.0;*;;RNR1440;*;
#CNR0001;*;CMW1440;*;CTYPEn-min-equi;*;
20210108173100 15.1
20210108173200 15.1
20210108173300 15.1
20210108173800 15.1
#SANRBETAUSPETE;*;
#RINVAL-777.0;*;;RNR1440;*;
#CNR0002;*;CMW1440;*;CTYPEn-min-equi;*;
20210108173100 18.3
20210108173200 18.3
20210108173300 18.3
20210108173800 18.3
#SANRBETAUSPETE;*;
#RINVAL-777.0;*;;RNR17280;*;
#CNR0006;*;CMW17280;*;CTYPEn-min-equi;*;
20210108173030 0.151
20210108173035 0.151
20210108173040 0.151
```

7 SMS data push

SMS (short message service) is a text messaging service component of most telephone, Internet, and mobile device systems. It uses standardized communication protocols to enable mobile devices to exchange short text messages. In the ecoLog 1000 it is used to send data to a receiving station.

Attention: the used data format is an OTT binary protocol which is limited to 16bit which limits transmitted values to ±32.767. This might cause limitations in some applications. This data format is not displayed on a mobile device.

8 Data and Station Identifiers - UUID

In the ecoLog 1000 it is possible to add identifiers to the station and each measuring channel. This information is than transmitted to a server in the following formats.

- OTTML (Station ID and Channel ID)
- MIS (Channel ID only)
- ZRXP (described under 4. as REXCHANGE value)

OTTML Example

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<StationDataList>
 <StationData stationId="0000000001" name="OTT Formats" timezone="+01:00"
ref="0207053D4B5A75EB9FAEB974E3"  UUID="StationExchangeID04022020" >
  <StationInfo time="2020-02-04T16:05:56" firmware="V1004" configtime="2020-01-30T20:53:05"
paramtime="2020-02-04T15:59:21" batteryVoltage="3.61" batteryMin="3.48" usedBattery="1728.5"
temperature="25.60" humidity="29"  lon="10.322513" lat="47.711428" deviceType="OTT ecolog 1000"
providerName="Vodafone.de" gsmSignal="4" ipAddress="100.67.51.217" transmissionCycle="300"
transmissionOffset="0" configuredTransmissionCycle="300" />
  <ChannelData channelId="0001" name="Distance to Water" unit="m" samplingInterval="60"
configuredSamplingInterval="60" storageInterval="60" configuredStorageInterval="60"
UUID="Channel1ExchangeID04022020" >
  <Values>
   <VT t="2020-02-04T16:00:00">2.348</VT>
   <VT t="2020-02-04T16:01:00">2.348</VT>
   <VT t="2020-02-04T16:02:00">2.348</VT>
  </Values>
  </ChannelData>
```

OTT MIS Example

```
<STATION>0000000001</STATION><SENSOR>0001</SENSOR><DATEFORMAT>YYYYMMDD</
DATEFORMAT><UUID>Channel1ExchangeID04022020</UUID>
20200204;160000;2.348
20200204;160100;2.348
20200204;160200;2.348
```

9 IP test transmission

To test the transmission to a server for each transmission a test transmission is available. The ecoLog 1000 will terminate the Bluetooth connection and perform a test transmission. After the test transmission is finished the ecoLog 1000 reconnects automatically via Bluetooth and the result of the test transmission is displayed.
A test transmission is independent from the chosen data format always an OTTML data format without data, this makes it easy to identify a test transmission.

Example:

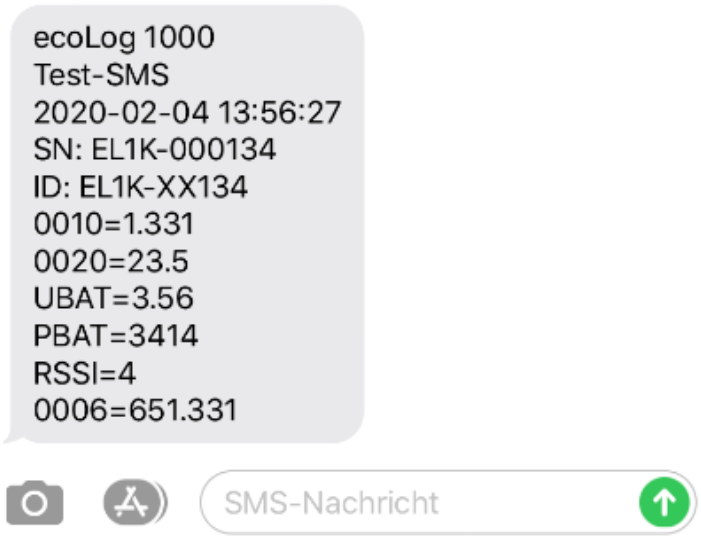
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<StationDataList>
 <StationData stationId="0000000001" name="OTT Formats" timezone="+01:00"
ref="02050D35495277E39DA6BBB161"  >
  <StationInfo time="2020-01-30T21:08:33" firmware="V1004" configtime="2020-01-30T20:53:05"
paramtime="2020-01-30T21:07:59" batteryVoltage="3.66" batteryMin="3.34" usedBattery="1613.4"
temperature="13.60" humidity="45"  lon="10.322513" lat="47.711428" deviceType="OTT ecolog 1000"
providerName="Vodafone.de" gsmSignal="6" ipAddress="100.66.72.104" transmissionCycle="900"
transmissionOffset="0" configuredTransmissionCycle="900" />
  <StationEvents>
  </StationEvents>
  <ChannelData channelId="0001" name="Distance to Water" unit="m" samplingInterval="180"
configuredSamplingInterval="180" storageInterval="180" configuredStorageInterval="180"  >
  </ChannelData>
  <ChannelData channelId="0002" name="Water Temperature" unit="&#176;C"
samplingInterval="180" configuredSamplingInterval="180" storageInterval="180"
configuredStorageInterval="180"  >
  </ChannelData>
  <ChannelData channelId="0006" name="Processed Value" unit="m" samplingInterval="180"
configuredSamplingInterval="180" storageInterval="180" configuredStorageInterval="180"  >
  </ChannelData>
  <ChannelData channelId="PBAT" name="Power Consumption" unit="mAh" samplingInterval="900"
configuredSamplingInterval="900" storageInterval="900" configuredStorageInterval="900"  >
  </ChannelData>
  <ChannelData channelId="RSSI" name="Signal Strength" unit="" samplingInterval="900"
configuredSamplingInterval="900" storageInterval="900" configuredStorageInterval="900"  >
  </ChannelData>
  <ChannelData channelId="UBAT" name="Supply Voltage" unit="V" samplingInterval="900"
configuredSamplingInterval="900" storageInterval="900" configuredStorageInterval="900"  >
  </ChannelData>
 </StationData>
</StationDataList>
```

As an alternative an immediate data transmission can be issued regardless of the scheduled transmission. In this case data in the chosen format will be transmitted.

10 Test SMS

The test SMS allows to check if the SMS connection is available, different to data SMS in OTT binary format as described in “5”, the test SMS contains clear text with date, serial number and actual readings. This can be read on any mobile phone.

Example:



11 MQTT Support

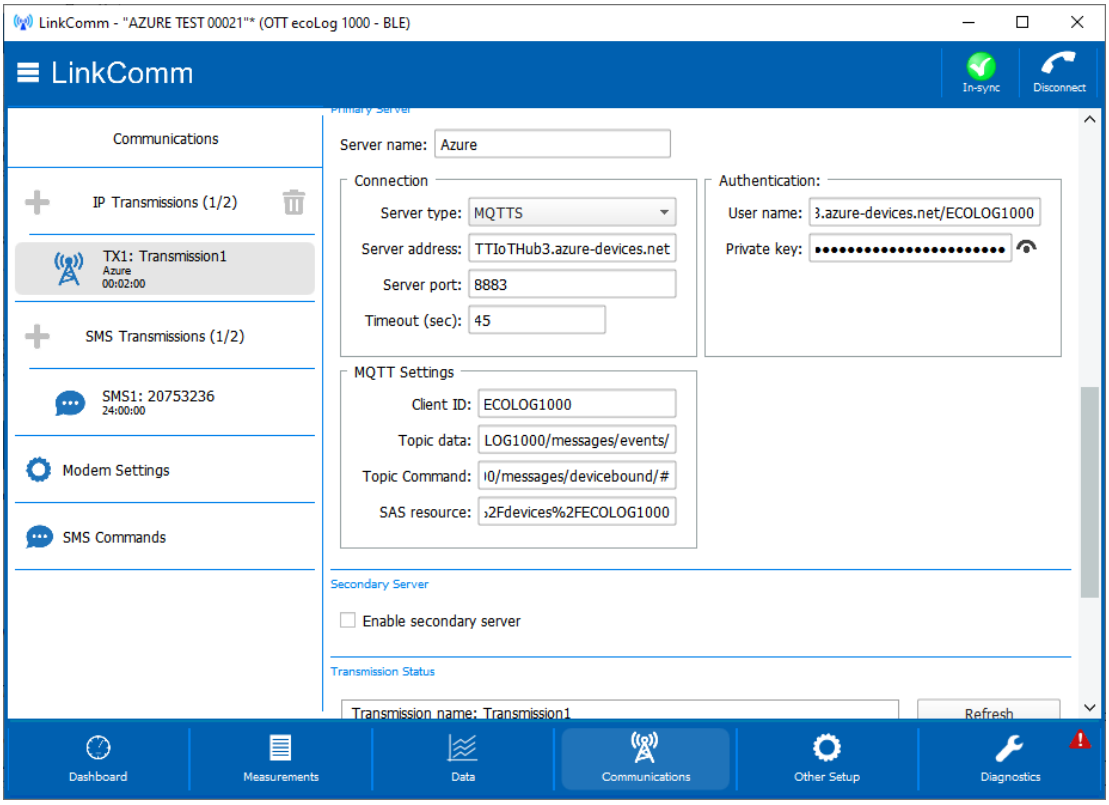
Starting with firmware version 1.00.6, MQTT and MQTTS protocols are supported for data transfer.
Support for MQTT/MQTTS can be used with existing LinkComm versions, provided that the following changes to the configuration are made:

1) For secure MQTT implementation the URL specified under HTTPS configuration needs to start with mqtts:// prefix. For example: mqtts://my.test.com.In
this case it is recommended to use the default port 8883.

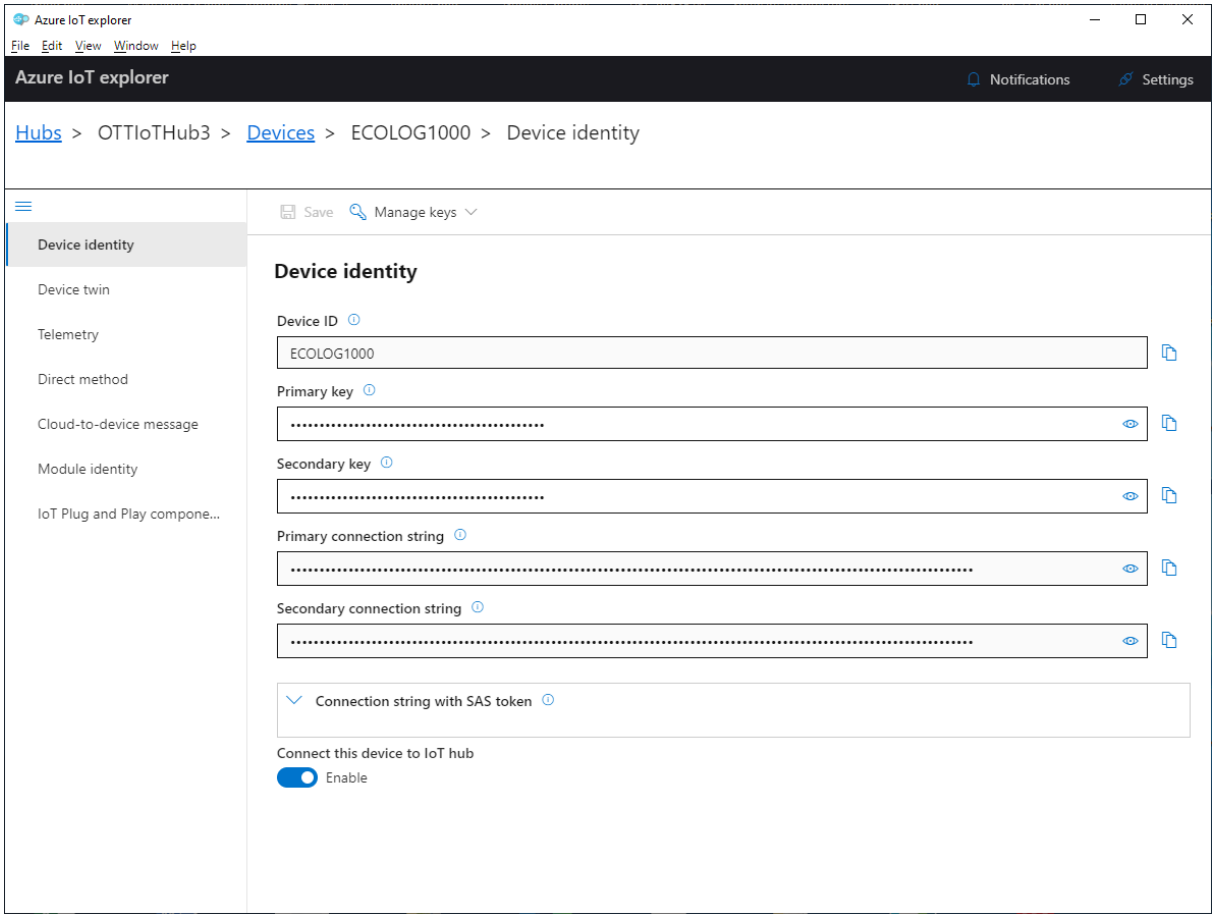
2) For non-encrypted MQTT implementation the URL specified under HTTP configuration needs to start with mqtt:// prefix. For example: mqtt://my.test.
com. In this case it is recommended to use the default port of 1883.

In case there is a need to apply username and password protection for MQTT, the credentials for HTTP can be used to specify the exact inputs. Example
screenshot below shows a set of configuration parameters for MQTTS with use of username and password. Key parameters set are:

- Type of server MQTTS (For Azure and other TLS enabled brokers it is necessary to select MQTTS. For non-TLS connections MQTT should be selected).
- Server address mqtts://OTTIoTHub2.azure-devices.net. Please note that this should be changed to the respective Azure IoT Hub server address.
- Server port should be set to 8883 for MQTTS and 1883 for MQTT connection.
- Timeout which can be modified depending on the connection speed. Typically, a value of 20-30 seconds should be sufficient. Please note that ecoLog 1000 automatically calculates the time-till-expiration value of the SAS token, depending on this configuration value.
- "Enable basic authentication" has to be checked for Azure servers.
- User name containing the user name required to authenticate with the MQTT broker. For example the user name could be OTTIoTHub2.azure devices.net/ECOLOG1000 . Please note that user name is not escaped. User name depends on the device defined in IoT Hub and the URL.
- Password containing the credential details required to authenticate with the MQTT broker. For Azure the password should be the secret master key in base64 encoded format (as it is give by Azure and/or Azure IoT Explorer). Please note that total length of the base64 encoded password should be less than 64 bytes!



In case there is no authentication required, then username and password can be left empty, and "Enable basic authentication" can be left unchecked. This however is not the case for Azure servers, where Authentication has to be checked and username and password have to be set. In order to make sure that correct authentication is used, the primary key can be obtained from the IoT explorer.



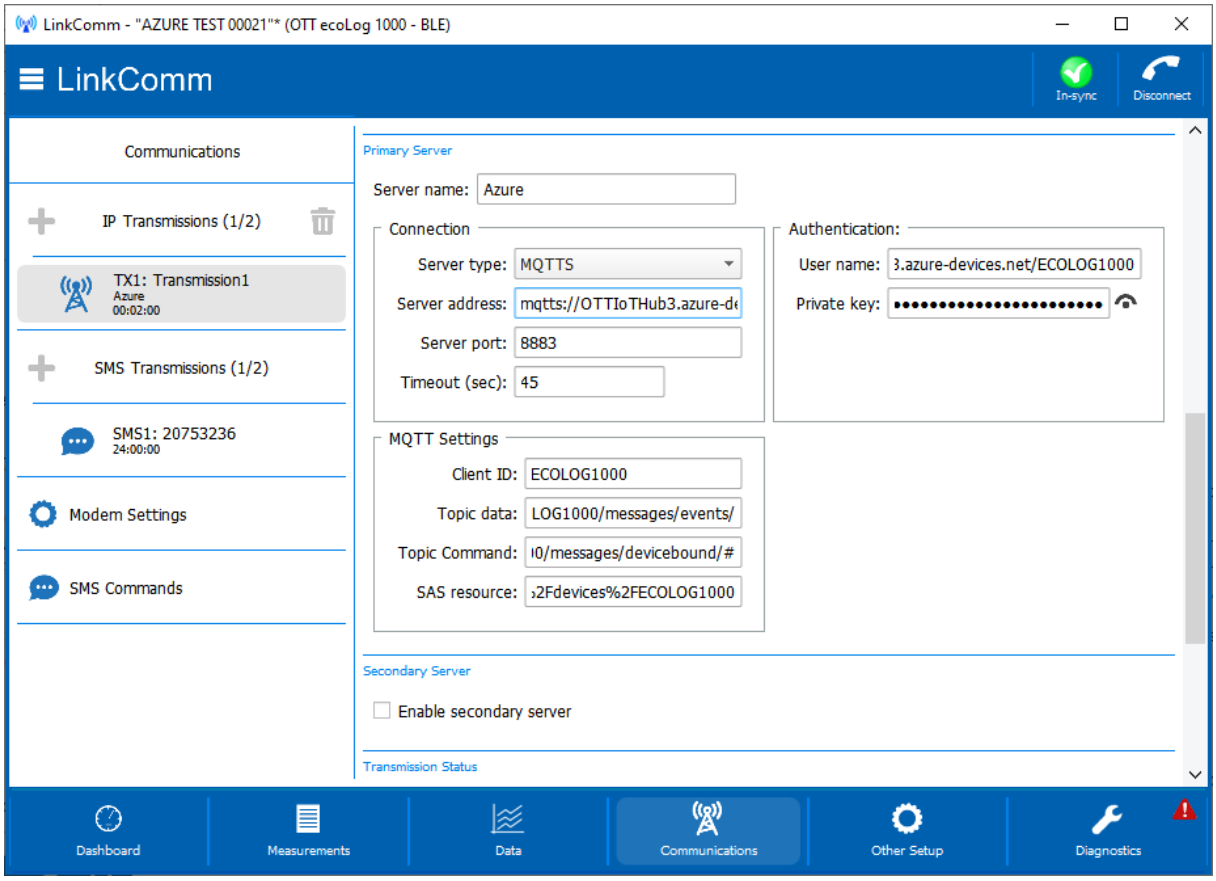
Azure-specific options

MQTT/MQTTS support supports Azure SAS authentication. In order to generate a proper SAS authentication string, the following options need to be configured:

- Enable basic authentication should be checked.
- Username should be the one expected from Azure. For example: OTTIoTHub2.azure-devices.net/ECOLOG1000.
- Password should be the primary or secondary key, without the sig= prefix. Please note that this key should be in base64 encoded format (as
- IoT explorer shows it) and up to 64 bytes long (base64 encoded string should be up to 64 bytes long).
- Topic data should correspond to the topic to submit data to. In this example case Azure would expect a user name like devices/ECOLOG1000/messages/events/.
- Topic command should correspond to the client identifier used for submitting to MQTT broker. Please note that according to the standard the broker may allow multiple clients with the same ID, but it is generally recommended to have separate identifiers.

- SAS resource should contain the resource string including the "sr=" prefix. For example sr=OTTIoTHub2.azure-devices.net%2Fdevices% 2FECOLOG1000. Please note that characters are already escaped in the configuration parameter (%2F instead of /). Please note that SAS resource depends on the URL and device identifier as defined in the IoT Hub.

Further reference SAS tokens can be found at: https://docs.microsoft.com/en-us/rest/api/eventhub/generate-sas-token



MQTT/MQTTS connection relies on the server side to interpret the data being sent. Therefore, all data types that are supported can be used.