Brief overview of "LTX"

“LTX” (“LTX Server” and “LTX Legacy”) stands for two collections of scripts that the associated data loggers can use to communicate with the Internet.

“LTX Server” uses an SQL database, “LTX Legacy” is a purely file-based solution and a subset of the “LTX Server”.

Data loggers often only transmit in longer intervals (e.g. every few hours), but each transmission can contain many measurements. All of this must run as reliably and quickly as possible: for example, faulty transmissions must be repeated and errors must be detected (such as empty batteries, moisture penetrating, connection interruptions). Data loggers must also be able to receive commands or updates from the server, for example. Often a large number of devices and different authorizations must also be managed.

“LTX” was developed for this purpose. Its primary focus is not on long-term storage of data, but as a fast and flexible entry portal (the “red carpet” to the cloud, so to speak).

“LTX” is completely open source, low-maintenance (it is designed to maintain itself) and can be installed on (almost) any minimal LAMP server. It only requires a web server with PHP and, for the “LTX server” variant, a small SQL database.

Since “LTX” only needs to store data temporarily (note: referred to as “quota” in the documentation), a large number of devices can be managed even with a small database. To save energy, “LTX” uses asynchronous scripts. Data loggers are thus not held up by potentially slow response times of the underlying systems (such as database or transfers). At the same time, “LTX server” and “LTX legacy” can be easily decoupled (“LTX legacy” is simply in front of the “LTX server”, so to speak).

The data loggers usually transmit via mobile Internet (2G, 4G, LTE-M/-NB) or via non-terrestrial networks.

Occasionally it is desired to include additional data, such as weather data or measurements from sensors without memory or gateways. This is also easily possible with “LTX”.

Upload with GET-URL

For example, simple weather stations often use a fairly simple system to pass on data by passing the measured values via URL, which are then simply packed into an HTTP request with URL parameters.

The whole thing could then look like this for a weather station, for example (in one line):

```
http://server.abc/script.php?
ID=0011223344556677&PASSWORD=SECRET&temp=25.9&humidity=61.72
```

The block „`ID=0011223344556677&PASSWORD=SECRET&temp=25.9&humidity=61.72`“ contains the access credentials (ID and PASSWORD) and the values. (‚temp‘ and ‚humidity‘).

In the simplest case and because, for example, the data is transferred frequently, the server's response may even not be evaluated.

For upload with GET-URL, "LTX" contains the script ".../lxu_wug_v1.php".

Example: EcoWitt WS90 – Weatherstation / „Wunderground“-Protocol

Many manufacturers offer GET-URL and almost identical protocols. Unofficially, another name for this is the "Wunderground" protocol, but there does not seem to be an officially defined standard for it. However, it is very easy to add your own sensors to "lxu_wug_v1.php" at any time. The currently implemented identifiers can be found in the appendix.

EcoWitt WS90 – Wireless Weatherstation

The EcoWitt WS90 is an inexpensive and compact weather station and measures all important parameters. It contains no mechanical parts (wind parameters are measured using ultrasonic, precipitation using a piezo sensor). It is solar powered and wireless. The measured values are sent every few seconds via radio (868 MHz or 915 MHz) and can be picked up by a display and/or a gateway, for example. Both the display and the gateway have WiFi and can be parametrized using an APP.

This also makes it possible to forward the data to a server. The format used is GET-URL (setting "Protocol like 'Wunderground'").

As listed in the header of the „lxu_wug_v1.php“ script:

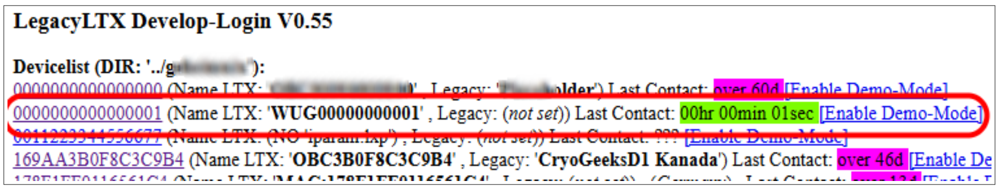
```
/* lxu_wug_v1.php - Script for Wunderground-aehnliche Daten per GET-URL
* ------------------------------------------------------
* Example
* server.abc/ltx/sw/lxu_wug_v1.php?ID=0011223344556677&PASSWORD=XXXXX&&tempf=61.70
*
* Setup for EcoWitt in the EcoWitt-APP:
* - Select Weather Service
* - Protocol: Wunderground
* - Server/IP(http:) Hostname, e.g. 'server.abc' or IP
* - Path: inkl. '/'und'?', e.g..: '/ltx/sw/lxu_wug_v1.php?'
* - ID: a (random) 16-digit HEX-number (e.g. '0123456789ABCDEF')
* - Key: 'D_API_KEY' (from File './conf/api_key.inc.php',
*         may also be dynamically per device './conf/check_dapikey.inc.php'
* - Port: normally: 80
* - Upload-Interval: e.g. 1 minute
* ------------------------------------------------------*/
```

That's it. When uploading for the first time, "LTX" automatically creates a configuration file from it. This is called 'iparam.lxp' and controls the measurement data acquisition on data loggers. Here, however, it only controls the way in which measurement values are imported into "LTX". The device type is defined as 950 (an internal assignment. Devices with type <= 999 have no memory, so they are sensors or gateways or - as here - a weather station). The types above are "real" data loggers.

It is also possible not to import individual channels, check alarms and linearize measurement values. But first of all, the data is on the server.

As mentioned, "LTX" consists of 2 layers:

- Legacy (always included). This is a predominantly text-based interface for administrator applications (here marked in red with the automatically generated device:



LegacyLTX Develop-Login V0.55

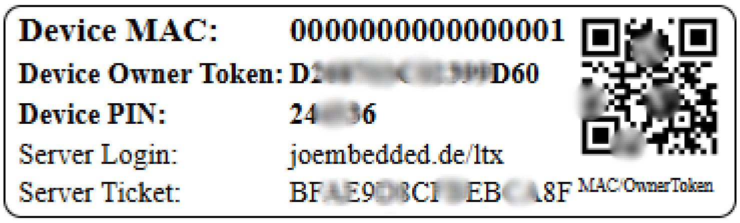
Devicelist (DIR: '../g[redacted]'):
0000000000000000 (Name LTX: '[redacted]' , Legacy: '[redacted]older') Last Contact: over 60d [Enable Demo-Mode]
0000000000000001 (Name LTX: '**WUG00000000001**' , Legacy: (*not set*)) Last Contact: 00hr 00min 01sec [Enable Demo-Mode]
0011223344556677 (Name LTX: (NO 'iparam.lxp') , Legacy: (*not set*)) Last Contact: ??? [Enable Demo-Mode]
169AA3B0F8C3C9B4 (Name LTX: '**OBC3B0F8C3C9B4**' , Legacy: '**CryoGeeksD1 Kanada**') Last Contact: over 46d [Enable De
[illegible]

- Server with GUI and SQL. This interface is intended for users.

The access data for the GUI is generated in the legacy area (“.../ltx/legacy/index.html”). The device is already listed there.

Some settings (e.g. the “quota” or how data is “pushed” further can only be set in the legacy area).

“Generate Key Badge” is used to generate a “Device Owner Token”:

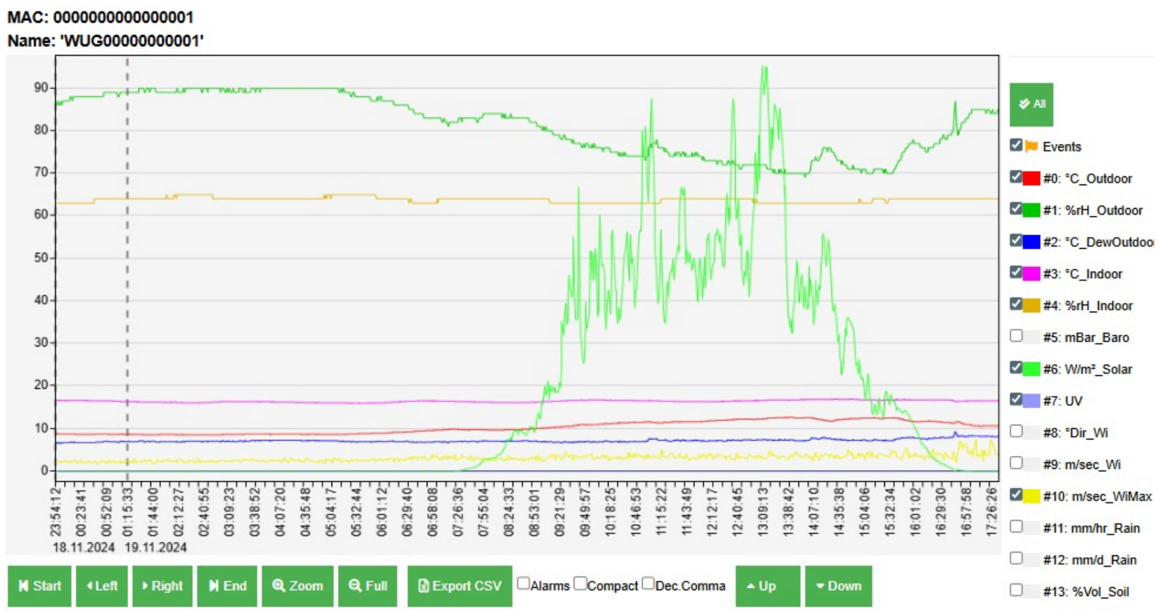


Device MAC: 0000000000000001
Device Owner Token: D2[redacted]D60
Device PIN: 24[redacted]6
Server Login: joembedded.de/ltx
Server Ticket: BF[redacted]8F

MAC/OwnerToken

This means that it can be accessed via the “Server Login” (on which the “LTX server” runs).

The data can also be released to other users or for external data retrieval via the “Device Owner Token”, see separate documentation for “LTX”.



Several channels of the EcoWitt WS90 in the graphic viewer of “LTX“

“LTX” does not offer any special display options other than a simple graph; the primary importance is to have the data available on the server; the precise evaluation can be carried out later on the target system.

Exceptions

Normally, "lxu_wug_v1.php" assumes that all parameters contained in the configuration file ('iparam.lxp') are also supplied. However, several problem states are conceivable. In these cases, the following happens:

- Only some of the parameters are provided: Apparently, the EcoWitt Station, for example, tends to simply omit individual parameters when there are transmission problems. In this case, the script sets the values to the error code "ErrNoValue".
- Parameters are invalid: For example, it can happen that a missing value is simply returned as an unrealistic value as an error code (e.g. -999°C for a temperature sensor).

Links

Github „LTX-Server“: https://github.com/joembedded/LTX_server

GitHub „OSX-Sensors and Gateways“:
https://joembedded.de/x3/ltx_firmware/index.php?dir=./Open-SDI12-Blue-Sensors

List of the defined GET-parameters

In "lxu_wug_v1.php" the parameters are defined as JSON. Each parameter is assigned to an index, for example for the 4 humidity in the interior. The configuration file 'iparam.lxp' can only get the data with the desired indices. Additional parameters can easily be added in the script.

Some parameters can occur multiple times, e.g. for temperatures. Others simply have a number appended. For example "soiltemp, soiltemp1, soiltem2, ..".

If there are any uncertainties or additional sensors: there is a "debug" mode (can be set in "Legacy"), then all GET parameters are written to the file ".../dbg/indata.log".

Some measured values also come in unusual units, such as the temperatures, which in "Wunderground" are probably mostly in '°F' instead of '°C'. These are also converted automatically.

List of GET-parameters, initial version of the script:

```
{
      "tempf": {
            "six": 0,
            "unit": "°C_Outdoor",
            "offset": 32.0,
            "multi": 0.555555555556,
            "digits": 3,
            "rem": "Outdoor Temp (raw: in °F)"
      },
      "humidity": {
            "six": 1,
            "unit": "%rH_Outdoor",
            "offset": 0.0,
            "multi": 1.0,
            "digits": 2,
            "rem": "Outdoor Humidity in %"
      },
      "dewptf": {
            "six": 2,
            "unit": "°C_DewOutdoor",
            "offset": 32.0,
            "multi": 0.555555555556,
            "digits": 3,
            "rem": "Dewpoint (raw: in °F)"
      },
      "indoortempf": {
            "six": 3,
            "unit": "°C_Indoor",
            "offset": 32.0,
            "multi": 0.555555555556,
            "digits": 3,
            "rem": "Indoor Temp (raw: in °F)"
      },
	"indoorhumidity": {
		"six": 4,
		"unit": "%rH_Indoor",
		"offset": 0.0,
		"multi": 1.0,
		"digits": 2,
		"rem": "Outdoor Humidity in %"
	},
	"baromin": {
		"six": 5,
		"unit": "mBar_Baro",
		"offset": 0.0,
		"multi": 33.8637526,
		"digits": 1,
		"rem": "Baro (raw in inch)"
	},
	"solarradiation": {
		"six": 6,
		"unit": "W/m²_Solar",
		"offset": 0.0,
		"multi": 1.0,
		"digits": 3,
		"rem": "Solar Radiation"
	},
	"UV": {
		"six": 7,
		"unit": "UV",
		"offset": 0.0,
		"multi": 1.0,
		"digits": 1,
		"rem": "UV-INdex"
	},
	"winddir": {
		"six": 8,
		"unit": "°Dir_Wi",
		"offset": 0.0,
		"multi": 1.0,
		"digits": 0,
		"rem": "0-360°"
	},
	"windspeedmph": {
		"six": 9,
		"unit": "m/sec_Wi",
		"offset": 0.0,
		"multi": 0.44704,
		"digits": 1,
		"rem": "WindSpeed"
	},
	"windgustmph": {
		"six": 10,
		"unit": "m/sec_WiMax",
		"offset": 0.0,
		"multi": 0.44704,
		"digits": 1,
		"rem": "Boe"
	},
	"rainin": {
		"six": 11,
		"unit": "mm/hr_Rain",
		"offset": 0.0,
		"multi": 25.4,
            "digits": 2,
            "rem": "Rain per last hr (raw: inch/hr)"
      },
      "dailyrainin": {
            "six": 12,
            "unit": "mm/d_Rain",
            "offset": 0.0,
            "multi": 25.4,
            "digits": 2,
            "rem": "Daily rain (raw: inch/d)"
      },
      "soiltempf": {
            "six": 13,
            "unit": "°C_Soil",
            "offset": 32.0,
            "multi": 0.555555555556,
            "digits": 3,
            "rem": "Soil Temp (raw: in °F)"
      },
      "soilmoisture": {
            "six": 14,
            "unit": "%Vol_Soil",
            "offset": 0.0,
            "multi": 1.0,
            "digits": 0,
            "rem": "Soil Moisture (Vol%)"
      }
}
```

***