webdyn

flexitron group

APPLICATION NOTE

WebdynEasy W M-Bus

Helper Scripts Application Note

Table of Contents

Introduction 3
Prerequisite 3
json2bson.py 4
bson2json.pu 4
data2csv.py 5
filters.py 5
whiteList.py 5
csv2whitelist.py 8
dataDecode.py 9

1 Introduction

This application note presents a series of tiny python scripts intended to help users that are not familiar with WebdynEasy original file formatting or those who don't have a dedicated portal that can facilitate WebdynEasy usage, or to introduce portal developers to the manipulation of WebdynEasy files.

Scripts are available for download from Webdyn Website:

https://www.webdyn.com/download/WebDynEasyScripts.zip

Also useful, the WebdynEasy UserManual available on our support page:

English version

French version

Deutsch version

Spanish version

2 Prerequisite

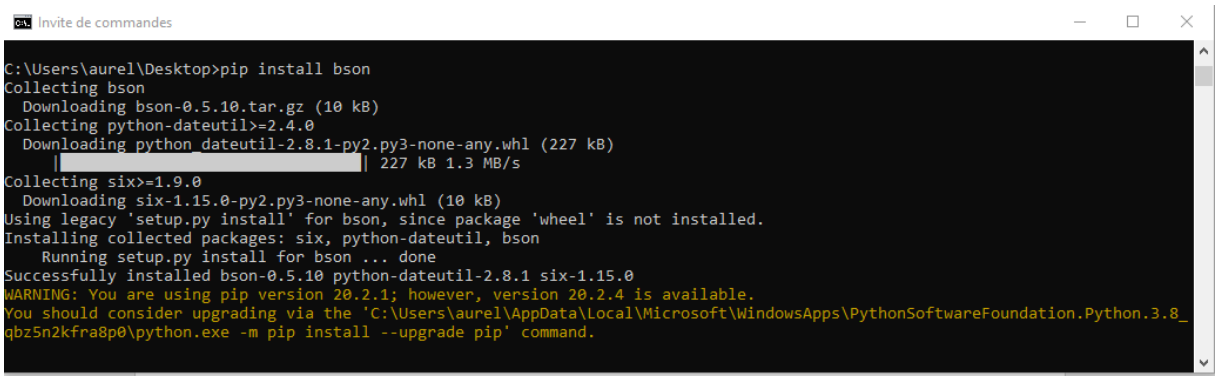
a) Windows

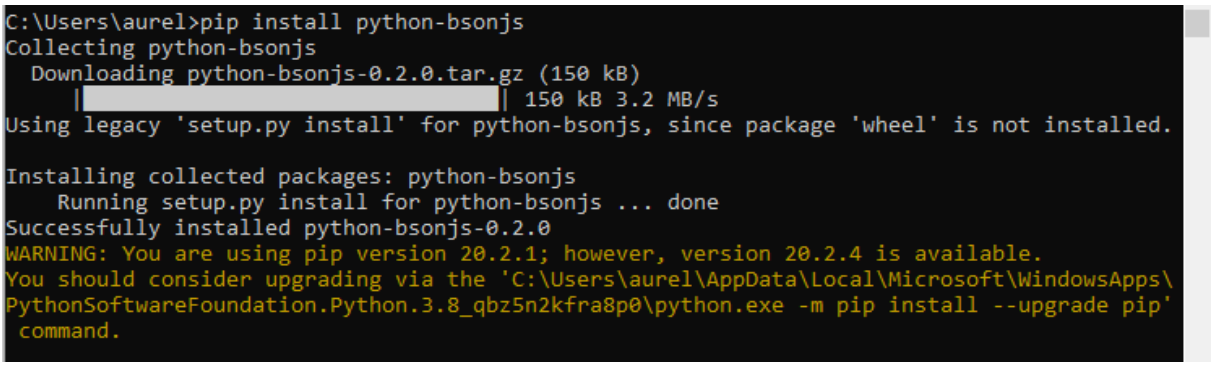
Install python version 3.8 or above:

https://www.python.org/ftp/python/3.8.6/python-3.8.6-amd64.exe

Then install following modules from command prompt:

- pip install bson
- pip install python-bsonjs





b) Linux

```
sudo apt install python3 sudo
apt  install  python3-pippip3
install bson
pip3 install python-bsonjs
```

3 json2bson.py

This script performs conversion from a json file to a bson file with crc

See script: https://www.webdyn.com/download/json2bson.py

```
usage:

     json2bson  inputfile.json  outputfile.bson

if the first arg is '-' stdin is used as input
if the second arg is '-'stdout is used as output
if the second arg is ommited the ouput file name will be: inptufile.bson
```

4 bson2json.py

This script performs conversion from a bson file to a json file.
See script: https://www.webdyn.com/download/bson2json.py

```
usage:

     bson2json  intpufile.bson  outputfile.json

if the first arg is '-' stdin is used as input
if the second arg is '-'stdout is used as output
if the second arg is ommited the ouput file name will be: intpufile.json
```

5 data2csv.py

This script performs conversion from a bson data file produced by the WebdynEasy to a csv file containing 1 telegram per line.

See script: https://www.webdyn.com/download/data2csv.py

```
usage:

     data2csv [-d] [-m] inputfile.data outputfile.csv

'-d' (optional) converts timestamp to a readable date
'-m' (optional) manufacturer code is converted to a readable valueif
the first arg is '-' stdin is used as input
if the second arg is '-'stdout is used as output
```

6 filters.py

This script creates a configuration file to be sent to the **WebdynEasy** to activate filter upon mediumand/or manufacturers.

See script: https://www.webdyn.com/download/filters.py

```
usage:

     filters.py "ITR,SAP,ZRI" "03,07,0c" ouputfileName.bson only

medium filter: filters.py "" "03,07,0c" ouputfileName.bson
only manufacturer filter: filters.py "ITR,SAP,ZRI" ""  ouputfileName.bson
```

7 whiteList.py

This script creates, from files containing meter list, a bson Whitelist file to be sent to theWebdynEasy. (See manual §4.2.2.1 page 44)

Naming the output file after this format: “<uid>-wlFilter.bson” will allow you to pass it directly to the /INBOX/uid folder for this whitelist to be taken into account after next connection.

See script: https://www.webdyn.com/download/whiteList.py

```
usage:

     whiteList.py ouputfileName.bson file1.wl file2.wl ...

the input files contain:
     first line:      Manufacturer code (3 letters ex: WDN)
     second line:     Generation or version code in hexadecimal (1 byte)
     third line:      Medium code in hexadecimal (1 byte)
     forth line:      List of the radio ID of the meters separated by comma
```

Example:

```
python3 whiteList.py out-wlFilter.bson sap.wl itr.wl
```

With sap.wl:

```
SAP
c0
01
20992a41,2099293d,209929c7,20992a3d,209929c1,20992a36
```

and itr.wl:

```
ITR
1e
03
11000310,11000311,11000312,11000313
```

Result: out-wlFilter.bson (converted to json):

```
{
  "wlFilter": [
    {
      "id": {
        "$binary":    "MExBKpkgwAE=",
        "$type": "00"
      }
    },
    {
      "id": {
        "$binary": "MEw9KZkgwAE=",
```

```
        "$type": "00"
      }
    },
    {
      "id": {
        "$binary": "MEzHKZkgwAE=",
        "$type": "00"
      }
    },
    {
      "id": {
        "$binary": "MEw9KpkgwAE=",
        "$type": "00"
      }
    },
    {
      "id": {
        "$binary": "MEzBKZkgwAE=",
        "$type": "00"
      }
    },
    {
      "id": {
        "$binary": "MEw2KpkgwAE=",
        "$type": "00"
      }
    },
    {
      "id": {
        "$binary": "kiYQAwARHgM=",
        "$type": "00"
      }
    },
    {
      "id": {
        "$binary": "kiYRAwARHgM=",
        "$type": "00"
      }
    },
    {
      "id": {
        "$binary": "kiYSAwARHgM=",
        "$type": "00"
      }
    },
    {
      "id": {
        "$binary": "kiYTAwARHgM=",
        "$type": "00"
      }
    }
  ],
  "crc": -1381163023
}
```

8 csv2whiteList.py

This script creates, from a csv file, a bson Whitelist file to be sent to the WebdynEasy. (See manual §4.2.2.1 page 44)

csv file must have the same format as the one obtained when using data2csv.py (§5), whatever theoption used for date format (-d) or manufacturer format (-m).

This allows you to retrieve a first data file from site, convert it thanks to the script data2csv.py (§5) modify the csv file as wanted (remove telegrams from unwanted devices eventually add some) andgenerate a dedicated WhiteList to the specific site.

Useful tip: magnet option 1 (§4.2.1.1) will trigger a Diagnostic command when magnet is used to gofrom storage mode to run mode. This diagnostic command is retrieving radio telegrams before doingthe connection to the FTP.

Naming the output file after this format: "<uid>-wlFilter.bson" will allow you to pass it directly to the /INBOX/uid folder for this whitelist to be taken into account after next connection.
See script:

https://www.webdyn.com/download/csv2whiteList.py

```
usage:

     csv2whiteList.py inputfile.csv outputfile.bson

if the first arg is '-' stdin is used as input if
the second arg is '-'stdout is used as output
```

Example:

```
csv2whiteList.py 456383-1606379932-data.csv 456383-wlFilter.bson
```

with 456383-1606379932-data.csv:

```
date,rssi,length,C field,manufacturer,radio id,genration,medium,CI field,payload

2000-01-01 01:01:09,-
70,138,44,SON,20591301,31,01,7a,2d00a025822adbcf1552ed4711f34792e90

2000-01-01 01:01:10,-
71,138,44,SON,20591302,31,01,7a,2d00a025822adbcf1552ed4711f34792e90
```

result:

```
{
  "wlFilter": [
    {
      "id": {
        "$binary": "7k0BE1kgMQE=",
        "$type": "00"
      }
    },
    {
      "id": {
        "$binary": "7k0CE1kgMQE=",
        "$type": "00"
      }
    }
  ],
  "crc": 1829404939
}
```

9 dataDecode.py

This script deciphers the encrypted telegram according to the keys listed in the Key.json file and it extracts data from all unencrypted or newly decrypted telegrams received in the WebdynEasy data files if they are compliant with the MBus format.
Warning: Once decrypted, telegram format is not always compliant with MBUS, in this case data can't be extracted with this script, retrieving the format from the manufacturer is then necessary to create an appropriate parser.

Few additional python modules need to be installed to run this script:
pip install pycryptodome
pip install bson
pip install pyMeterBus

Link towards the script:
https://www.webdyn.com/download/dataDecode.py

```
usage:

     dataDecode.py key.json data.bson outputfile.csv

if the third arg is '-' stdout is used as output
```

3 arguments shall be passed to the script:
- `key.json` : file containing the keys to decipher encrypted telegrams.
- `data.bson` : original data file from the WebdynEasy.

- `outputfile.csv`: file containing the extracted data from the telegrams. CSV format with semi column as default separator (this can be changed line 11)

Key file description:

The key file contains the AES keys listed in a json format

There are two types of keys:

- product keys (item: "id")
- manufacturer keys (item: "manuf").

Sample key file:


```
{
 "id":{"00001949":"00000000000000000000000000000000","21460457":"00000000000000000000000000000000","21460459":"00000000000000000000000000000000"},
 "manuf":{"MAD":"00000000000000000000000000000000"}
}
```


Both items must be present, even if there is no product key:


```
{
 "id":{"":""},
 "manuf":{"MAD":"00000000000000000000000000000000"}
}
```


Example:

python dataDecode.py keyfile.json 412603-1661962552-data.bson outputfile.csv

The files used in this example can be found in the following zip file:
https://www.webdyn.com/download/DataDecode.zip

The result file (outputfile.csv), which can be found in the DataDecode.zip file, shows three possible states:

-Unencoded telegram doesn't need a key (not needed in the key file), therefore the data is present and visible (example: ITW)

-Encoded telegram using the key file, therefore the data is present and visible (example: WDN)

-Encoded telegram which can't be read since we don't have the corresponding key (needed in the key file) (example: DME)

Links describing MBus data encoding used in WMBus:
https://m-bus.com/documentation-wired/06-application-layer
https://m-bus.com/documentation-wired/08-appendix

Link to the 2 reference files cited in the comments:
https://oms-group.org/fileadmin/files/download4all/specification/Vol2/4.2.1/OMS-Spec_Vol2_Primary_v421.pdf

https://www.compass-security.com/fileadmin/Datein/Research/Praesentationen/blackhat_2013_wmbus_security_whitepaper.pdf