webdyn
Flexitron group

WebdynEasy Wireless M-bus
Application Note

HTTP(s) Specifications

Table of contents

1 HTTP(s) Specifications

1.1 Introduction

This document presents the use of the HTTP protocol by the WebdynEasy Wireless M-bus gateway for file transfer.

In order to be compatible with portals that use cloud services where the FTP protocol is not available we introduce HTTP transfert protocol in addition to FTP.

The HTTP protocol will be used to upload files on the server (data,supervision,alarms,...) and to download configuration and commands files.

The HTTP client implementation is provided by the BG95 modem. The HTTPS is also supported.

The HTTP/HTTPS is based on current FTP file contents, only the transport layer is updated, payload is still the same. Features and associated files are remapped to HTTP/HTTPS requests.

> **Warning**
>
> It is not possible to configure the product to use both protocols simultaneously; their use is exclusive.

1.2 Configuration

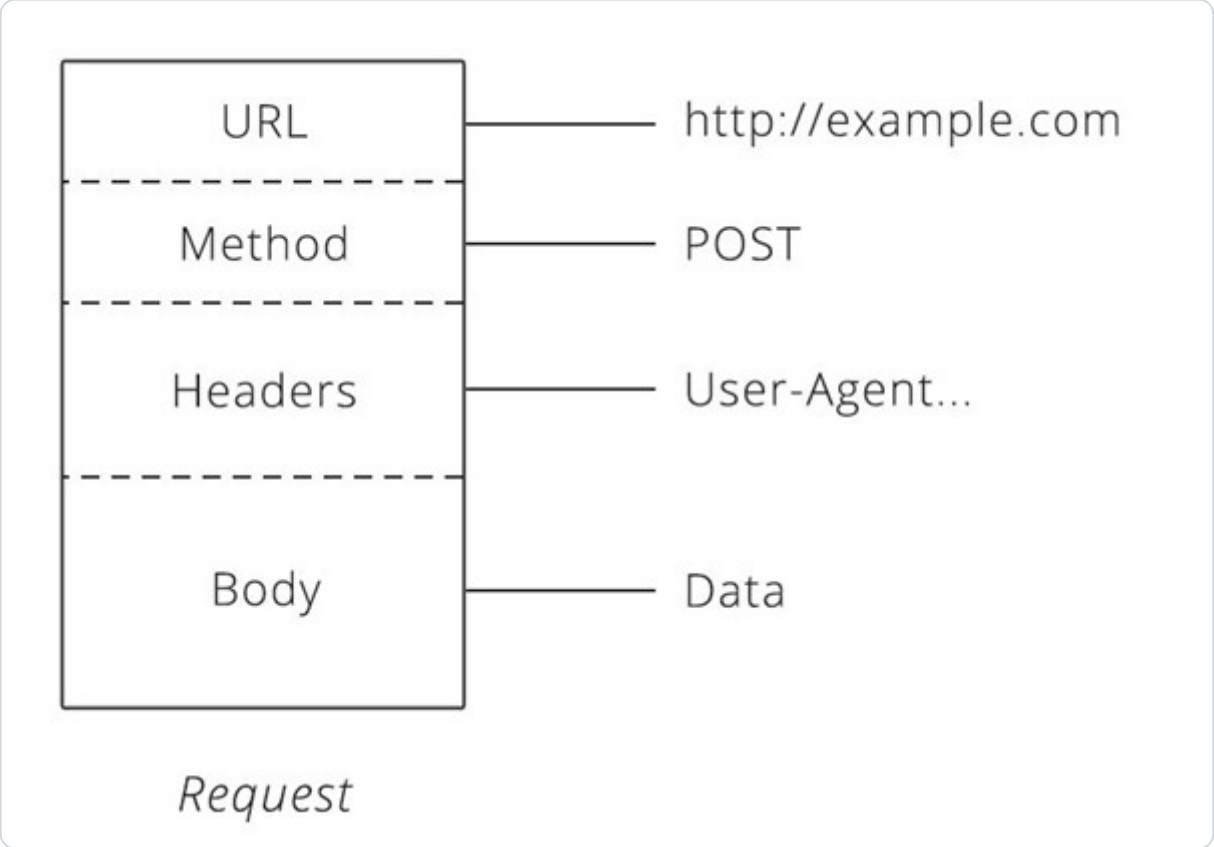
For compatibility reasons the HTTP parameters will be stored in the `configuration.remote.ftp` object of the gateway configuration.

Directory	Type	Value
/remote/ftp/mode	Integer	**0**: FTP **1:** FTPS **2**: FTPS+certificates **3**: HTTP **4**: HTTPS **5**: HTTPS+certificates
/remote/ftp/addr	String	Server address. It can be numerical or symbolic If the port is different from 80 it must be add at the end of the address after ":". The maximum length if 128 characters.
/remote/ftp/user	String	username for HTTP authentication
/remote/ftp/pass	String	password for HTTP authentication
/remote/ftp/dir	String	Path of the URL The maximum length if 128 characters.
/remote/ftp/cacert	String	Root Certificate of Authority certification
/remote/ftp/clientcert	String	Signed certificate for client
/remote/ftpclientkey	String	private key for client

1.3 HTTP/HTTPS

1.3.1 Requests

HTTP/HTTPS requests are as following:



http request

WebdynEasy W-MBUS respect the following conventions:

- **URL:** Identify by a unique address unique objects or containers
- **METHOD:** the following methods are implemented:
  - **HTTP/GET**: Retrieve a resource identified the URL
    - Request BODY : NONE
    - Response BODY : BSON for requested object or Container contents list for containers
  - **HTTP/PUT**: Modify or clear a resource
    - Request BODY : JSON for modified data (will be merged with existing object)
    - Response BODY : JSON from modified object
  - **HTTP/POST**: Create a resource
    - Request BODY : BSON for object to be created
    - Response BODY :
- **HEADERS**: the following headers are provided by WebdynEasy W-MBUS
  - Content-Type: application/octet-stream
  - Authorization : Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ== (User and password as per basic Authentication specification)
  - Content-Length: <length of the payload>

The following headers must be provided by the server

- Content-Length: <length of the payload>
- Content-Type: application/octet-stream

In HTTPS, TLS and cypher suite are detailed in annex. The BG95 modem supports the SSL/TLS protocol for secure communication. It is possible to use simple user/password authentication or to use certificates.

> **Note**
>
> If certificates and keys are used, the files must first be loaded into the modem's memory using the "certificate" command.

> **Note**
>
> Please note that HTTP / HTTPS is provided by Quectel layer, no change can be done in HTTP/HTTPS layer.

1.3.2 Security

The BG95 modem supports the SSL/TLS protocol for secure communication. It is possible to use simple user/password authentication or to use certificates.

> **Note**
>
> If certificates and keys are used, the files must first be loaded into the modem's memory using the "certificate" command.

1.3.3 Error Management

The HTTP commands have a time out of 300 seconds. If a command failed 1 retry is done.

1.4 Functional Description

1.4.1 Push data to the server

Request

```
POST /DATA/<UID>-<TimeStamp>-data.bson
POST /DATA/<UID>-<TimeStamp>-data.bson.aes   (if AES encryption is used)
```

Parameters

- **UID**: Gateway Unique Identifier
- **TimeStamp**: data Timestamp

Example

```
POST /DATA/499198-1690876878-data.bson
```

Request Payload

```
<Bson file contents>
```

Reply Payload

```
<empty>
content-length=0
```

Example of a file upload (TCP capture)

Request

```
POST /DATA/499198-1690876878-data.bson HTTP/1.1
Host: 172.20.30.1:8000
Accept: */*
User-Agent: QUECTEL_MODULE
Connection: Keep-Alive
Content-Type: application/octet-stream
Content-Length: 6832
```

Reply

```
HTTP/1.0 200 OK
Server: minimal_dav Python/3.8.10
Date: Thu, 11 Jan 2024 14:49:25 GMT
Last-modified: Thu, 11 Jan 2024 13:22:31 GMT
Content-length: 0
```

1.4.2 Push alarm to the server

Request

```
POST /ALARM/<UID>-<TimeStamp>-alarm.bson
POST /ALARM/<UID>-<TimeStamp>-alarm.bson.aes   (if AES encryption is used)
```

Parameters

- **UID**: Gateway Unique Identifier
- **TimeStamp**: data Timestamp

Request Payload

```
<Bson file contents>
```

Reply Payload

```
<empty>
content-length=0
```

1.4.3 Push supervision to the server

Request

```
POST /SUPERVISION/<UID>-<TimeStamp>-supervision.bson
POST /SUPERVISION/<UID>-<TimeStamp>-supervision.bson.aes   (if AES encryption is used)
```

Parameters

- **UID**: Gateway Unique Identifier
- **TimeStamp**: data Timestamp

Request Payload

```
<Bson file contents>
```

Reply Payload

```
<empty>
content-length=0
```

1.4.4 Push log to the server

Request

```
POST /SUPERVISION/<UID>-<TimeStamp>-log.bson
POST /SUPERVISION/<UID>-<TimeStamp>-log.bson.aes (if AES encryption is used)
```

Parameters

- **UID**: Gateway Unique Identifier
- **TimeStamp**: data Timestamp

Request Payload

```
<Bson file contents>
```

Reply Payload

```
<empty>
content-length=0
```

1.4.5 Check the configuration known by the server

> **Note**
>
> Configuration synchronization is done in two steps :
>
> - check the configuration know by the server

- update the configuration if needed

Request

```
GET /CONFIG/<UID>
```

Parameters

- **UID**: Gateway Unique Identifier

Reply Payload

Empty if no configuration is known by the server

```
<empty>
content-length=0
```

Or configuration known by the server with the same naming convention that FTP

```
<UID>-<TimeStamp>-config.bson
<UID>-<TimeStamp>-config.bson.aes (if AES encryption is used)
```

Info

If the reply is empty (content-length=0) or the provided file name is not matching the local configuration THEN the configuration is posted to the server

Example

```
499198-1690876878-config.bson
```

1.4.6 Update the configuration known by the server

Note

Configuration synchronization is done in two steps :

- check the configuration know by the server
- update the configuration if needed

Request

```
POST /CONFIG/<UID>-<TimeStamp>-log.bson
POST /CONFIG/<UID>-<TimeStamp>-log.bson.aes (if AES encryption is used)
```

Parameters

- **UID**: Gateway Unique Identifier
- **TimeStamp**: data Timestamp

Request Payload

```
<Bson file contents>
```

Reply Payload

```
<empty>
content-length=0
```

1.4.7 Check the inbox contents

Inbox content handling done by following those steps :

- check the inbox contents (files names) Then for each files
- GET the inbox file
- Process the file as per WebdynEasy documentation
- PUT the acknowledgement.

When the acknowledgement is posted, the server must consider the command has handled and must remove the command file from the inbox contents

Request

```
GET /INBOX/<UID>
```

Parameters

- **UID**: Gateway Unique Identifier

Reply Payload

Empty if no inbox file is pending

```
<empty>
content-length=0
```

Or list of pending inbox files separated by a semicolon. filenames follow FTP naming convention.

```
<UID>-<TimeStamp>-cfg.bson
<UID>-<TimeStamp>-cmd.bson
<UID>-<TimeStamp>-xxx.bson

<UID>-<TimeStamp>-cfg.bson.aes (if AES encryption is used)
<UID>-<TimeStamp>-cmd.bson.aes (if AES encryption is used)
<UID>-<TimeStamp>-xxx.bson.aes (if AES encryption is used)
```

Info

If the reply is empty (content-length=0) or the provided file name is not matching the local configuration THEN the configuration is posted to the server

Example

```
499198-1690876878-cfg.bson
```

1.4.8 Get the inbox file

Request

```
GET /INBOX/<filename>
```

Parameters

- **filename**: retrieved by the previous request. As described it follows the FTP name rule

Reply Payload

```
<Bson file contents>
```

Reply payload

```
<empty>
content-length=0
```

1.4.9 Put the acknowledgement

Request

```
PUT /INBOX/<filename>
```

Parameters

- **filename**: retrieved by the previous request. As described it follows the FTP name rule

Reply Payload

```
<Bson file contents>
```

Reply payload

```
<empty>
content-length=0
```

1.5 Examples

1.5.1 Command and configuration update

In this example, there is 2 files in the inbox subdirectory of the gateway.

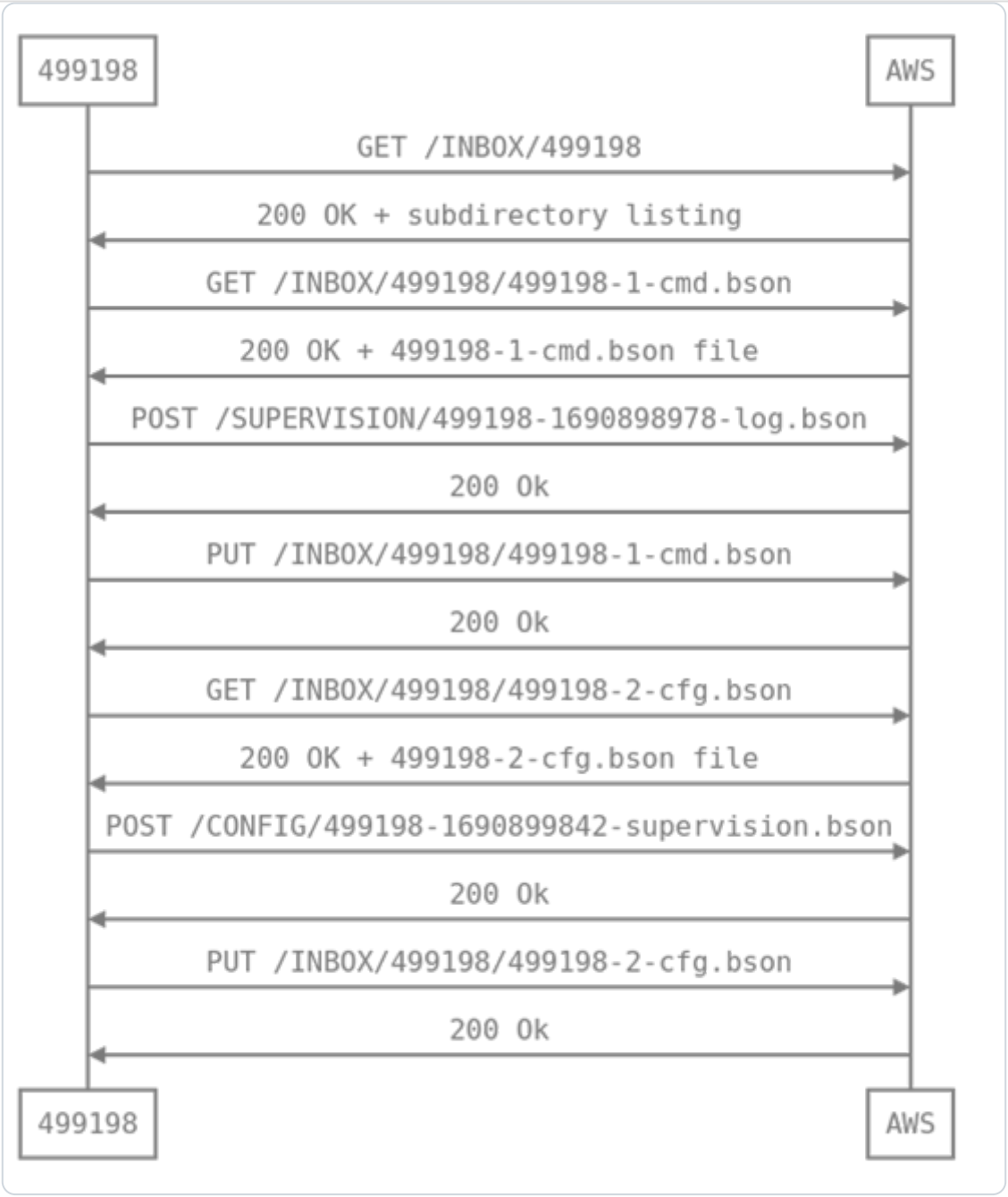
`499198-1-cmd.bson`


```
{
  "cmd": [
    {
      "type": "logGet",
      "cid": "1"
    }
  ],
  "crc": 2040707386
}
```


This command asks the gateway to upload the log file. `499198-2-cfg.bson`


```
{
  "config": {
    "radio": {
      "manufFilter": {
        "$binary": "jlw=",
        "$type": "00"
      }
    }
  },
  "crc": -1156637560
}
```


This is a simple parameter update in the configuration. Sequence diagram:



http request2

Inbox Listing Request:

```
GET /INBOX/499198 HTTP/1.1
Host: 172.20.30.1:8000
Accept: */*
User-Agent: QUECTEL_MODULE
Connection: Keep-Alive
```

Subdirectory listing answer:

```
HTTP/1.0 200 OK
Server: minimal_dav Python/3.8.10
Date: Thu, 11 Jan 2024 14:49:25 GMT
Last-modified: Thu, 11 Jan 2024 13:22:31 GMT
Content-length: 38
```

```
499198-1-cmd.bson;499198-2-cfg.bson
```

Command file download request:

```
GET /INBOX/499198/499198-1-cmd.bson HTTP/1.1
Host: 172.20.30.1:8000
Accept: */*
User-Agent: QUECTEL_MODULE
Connection: Keep-Alive
```

Answer:

```
HTTP/1.0 200 OK
Server: minimal_dav Python/3.8.10
Date: Thu, 11 Jan 2024 14:49:25 GMT
Last-modified: Thu, 11 Jan 2024 13:22:31 GMT
Content-length: 60

<499198-1-cmd.bson file>
```

1.6 Annex

1.6.1 SSL version supported

- **TLS 1.2**

1.6.2 Supported SSL Cipher Suites

Code of Cipher Suites	Name of Cipher Suites
0x0035	TLS_RSA_WITH_AES_256_CBC_SHA
0x002F	TLS_RSA_WITH_AES_128_CBC_SHA
0x0005	TLS_RSA_WITH_RC4_128_SHA
0x0004	TLS_RSA_WITH_RC4_128_MD5
0x000A	TLS_RSA_WITH_3DES_EDE_CBC_SHA
0x003D	TLS_RSA_WITH_AES_256_CBC_SHA256
0xC002	TLS_ECDH_ECDSA_WITH_RC4_128_SHA
0xC003	TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
0xC004	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
0xC005	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
0xC007	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
0xC008	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
0xC009	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
0xC00A	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
0xC011	TLS_ECDHE_RSA_WITH_RC4_128_SHA
0xC012	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
0xC013	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
0xC014	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
0xC00C	TLS_ECDH_RSA_WITH_RC4_128_SHA
0xC00D	TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
0xC00E	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
0xC00F	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
0xC023	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
0xC024	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
0xC025	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
0xC026	TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
0xC027	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
0xC028	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

Code of Cipher Suites	Name of Cipher Suites
0xC029	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
0xC02A	TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
0xC02F	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
0xFFFF	Support all cipher suites above