GET / PUT HTTPS request using BASIC Script or Java

1. Introduction

The aim of this document is to explain how to execute GET/PUT HTTP**S** requests using BASIC 2 or Java (Flexy only).

As a side effect, it will also be explained how to provide your own certificates.

2. Setting up the environment

This chapter describes a configuration as an example. We will explain how to generate a certificate, configure the server side and the client side.

2.1. Generate a certificate

The eWON is loaded with the latest CA root certificates from the Mozilla foundation which means that generating a certificate is only necessary if the server you are reaching is not officially recognized (self-made certificate or unknown custom server for example).

Here is an example of a command using OpenSSL tool to generate certificates:

```
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem
-days 360
```

By running this command, it will create:

- a certificate expiring in 360 days
- a private certificate: cert.pem that you will provide to your server
- a public key: key.pem that you will provide to your server and your clients (= eWONs).

- Note -

eWON only accepts certificates with a TLS 1.0

2.2. Server setup

The server can use a standard Apache with the SSL plugin enabled or you can use OpenSSL tools if you just want to test your keys.

We decided to go for the second option:

```
sudo openssl s_server -accept 443 -cert cert.pem -key key.pem
```

Providing your *cert.pem* and your *key.pem*, you can set up a simple HTTPS server listening on your computer to any destination (0.0.0.0) on the port 443.

The *sudo* is mandatory on most of Linux distribution because simple users are not allow to listen on ports 443.

2.3. Client setup

Before checking how to set up your eWON to talk to your HTTPS server, we will try a standard tool: curl.

```
curl --verbose --cert key.pem --form "param1=https-curl"
"localhost:443"
```

Providing your key.pem, you execute a simple POST on your HTTPS server posting the form "param1=https-curl".

You can replace localhost by any accessible IP/hostname/etc. on your network.

3. eWON setup

On the eWON, a SYS config variable that defines the path to the folder where your certificates are stored is provided: **HttpCertDir**.

You can upload in the defined folder, using an FTP client, an unlimited amount of certificates but the eWON will only read the first 20 (lexicographical order).

The default value is empty. We recommend using */usr/certificates*.

You can change it at any time, just make sure that this name is valid and pointing to an directory accessible to any eWON user. If this is not the case, the default value would be restored.

According to the previous exposed HTTPS server setup, if you want your eWON to put HTTPS on your server, simply drop in /usr/certificates your *key.pem*. Now your eWON is able to talk HTTPS (encryption + certificate system) to your server.

4. Basic

The GET / PUT HTTPS feature is available for Flexy devices running under BASIC 2 (available since firmware v8.2s0).

We kept it simple by using existing BASIC2 function: *PUTHTTP*.

By prefixing your server address with *https://*, the eWON will detect that you are trying to

connect using an HTTPS connection.

If you don't prefix your server address by https://, your eWON will use standard http connection on port 80.

Once the HTTPS protocol detected, the eWON will parse the HttpCertDir system config variable, list certificates available in this directory and try to negotiate an SSL connection.

4.1. Example:

As we suppose your server is accessible on the IP 192.168.58.13 and your eWON can reach this IP, this is to be typed in the BASIC IDE section of the eWON web interface:

```
PUTHTTP "https://192.168.58.13","/","param1=value”,"POSTERROR"
```

You should now see something like that on your server:

```
-----BEGIN SSL SESSION PARAMETERS-----
…
-----END SSL SESSION PARAMETERS-----
Shared ciphers:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-
AES256-SHA:DHE-DSS-AES256-SHA:DHE-RSA-CAMELLIA256-SHA:DHE-DSS-
CAMELLIA256-SHA:ECDH-RSA-AES256-SHA:ECDH-ECDSA-AES256-SHA:AES256-
SHA:CAMELLIA256-SHA:ECDHE-RSA-DES-CBC3-SHA:ECDHE-ECDSA-DES-CBC3-
SHA:EDH-RSA-DES-CBC3-SHA:EDH-DSS-DES-CBC3-SHA:ECDH-RSA-DES-CBC3-
SHA:ECDH-ECDSA-DES-CBC3-SHA:DES-CBC3-SHA:ECDHE-RSA-AES128-SHA:ECDHE-
ECDSA-AES128-SHA:DHE-RSAAES128-SHA:DHE-DSS-AES128-SHA:DHE-RSA-SEED-
SHA:DHE-DSS-SEED-SHA:DHE-RSACAMELLIA128-SHA:DHE-DSS-CAMELLIA128-
SHA:ECDH-RSA-AES128-SHA:ECDH-ECDSAAES128-SHA:AES128-SHA:SEED-
SHA:CAMELLIA128-SHA:ECDHE-RSA-RC4-SHA:ECDHEECDSA-RC4-SHA:ECDH-RSA-
RC4-SHA:ECDH-ECDSA-RC4-SHA:RC4-SHA:RC4-MD5:EDHRSA-DES-CBC-SHA:EDH-
DSS-DES-CBC-SHA:DES-CBC-SHA:EXP-EDH-RSA-DES-CBCSHA:EXP-EDH-DSS-DES-
CBC-SHA:EXP-DES-CBC-SHA:EXP-RC2-CBC-MD5:EXP-RC4-MD5


CIPHER is ECDHE-RSA-AES256-SHA
Secure Renegotiation IS supported
POST / HTTP/1.1
Host: 192.168.58.13
Connection: TE, close
TE: trailers
Pragma: no-cache
Content-Type: multipart/form-data;
boundary=--------------------------7d624e177018a
```

```
Transfer-Encoding: chunked



0069
------------------------------7d624e177018a
Content-Disposition: form-data;


…
https
…


DONE
shutting down SSL
CONNECTION CLOSED
```

4.2. RequestHTTPX & ResponseHTTPX

You can also manage HTTPS Headers with RequestHTTPX & ResponseHTTPX BASIC command. For more information see PRI-(BASIC2) document on eWON Developer website.

5. Java

You should expect the same behaviour than in BASIC2: use existing *GetHttp* function, but put https:// in front of your URL.

This example executes a GET https on httpbin.org domain on the url /get with the parameter show_env=1 and store the result in /usr/javas.txt:

```
try {
    com.ewon.ewonitf.ScheduledActionManager.GetHttp("https://httpbin.org",
    "/usr/javas.txt",
    /get?show_env=1");
} catch (EWException ex) {
    System.out.println(ex.getMessage());
}
```

6. Common errors

HTTPS: SSL server does not present a certificate signed by a trusted CA	It's probably a self signed certificate
HTTPS: The certificate was not issued for the specified server	The certificate specifies a host that doesn't match your server hostname (may be a man in the middle attack)
HTTPS: The certificate was not signed by a trusted Certificate Authority	The certificate wasn't signed by a CA (may be a self signed certificate or a man in the middle attack)
HTTPS: The certificate validity expired	Your certificate is no longer valid
HTTPS: The certificate validity is in the future	Your certificate will be valid in the future. May be a problem with your eWON clock or simply OK by design.

Revision

Revision History

Revision Level	Date	Description
1.0	29/04/2016	Original Document

Document build number: 15

Note concerning the warranty and the rights of ownership:

The information contained in this document is subject to modification without notice. Check http://ewon.biz/support for the latest documents releases.

The vendor and the authors of this manual are not liable for the errors it may contain, nor for their eventual consequences.

No liability or warranty, explicit or implicit, is made concerning the quality, the accuracy and the correctness of the information contained in this document. In no case can the manufacturer's responsibility be called for direct, indirect, accidental or other damage occurring from any defect of the product or mistakes coming from this document.

The product names are mentioned in this manual for information purposes only. The trade marks and the product names or marks contained in this document are the property of their respective owners.

This document contains materials protected by the International Copyright Laws. All reproduction rights are reserved. No part of this handbook can be reproduced, transmitted or copied in any way without written consent from the manufacturer and/or the authors of this handbook.

eWON sa