eWON
MACHINES CAN TALK

Reference Guide

RG 004-0-EN / Rev. 1.3

M2Web API

Talk2M SDK - 1.3.0.26452

Reference Guide explaining how to use the M2Web API.

ewon.biz/support

Table of Contents

1. Introduction

The M2Web API exposes a set of HTTPS web services based on the Talk2M M2Web HTTPS service.

The API exposes web services aimed at querying M2Web information (basically the contents found in the M2Web portal) and access to the eWON web server (and possible web servers on eWON LAN).

Examples of features of the API:

- Stateless M2Web sessions: one can *POST* or *GET* to the eWON using a single API call.

- Stateful requests (explicit session management) remain supported.

- Large volume applications: The API allows concurrent access to different eWONs such as *simultaneously* sending a single request to a large set of eWONs. The API (*stateless* mode) is not subject to concurrent M2Web connections.

- The information available in the portal can be retrieved in a machine-friendly format (JSON). This mostly consists in a list of eWONs and their state (online, offline, send wake-up SMS,...).

1.1. Keywords used in this document.

- ESP: eWON Solution Provider. A third-party supplier (hardware, software, services,...) that delivers business solutions involving eWONs.

- DevID: Talk2M Developer ID. A unique identifier owned by every developer of a program or application that uses the M2Web API. DevID example: *21EC2020-3AEA-1069-A2DD-08002B30309D*.

2. Talk2M Developer

2.1. Registration

All Talk2M users or third parties (such as ESPs) that wish to use a Talk2M API such as the M2Web API must first register as a Talk2M developer.

To do so, fill in the web form that can be found on eWON Developers website. As a result of the registration process, the registrant receives a unique identifier.

In this document, we call it the Talk2M Developer ID, or DevID for short version.

The DevID must be specified in every M2Web API call in order to identify the 3rd party. This is vital as M2Web needs to validate the DevID to send back data: API calls with missing or incorrect DevID are obviously rejected.

2.2. Talk2M SDK

In the Talk2M SDK you downloaded, two different folders can be found: DataMailbox and M2Web API.

2.2.1. DataMailbox

The eWON pushes its historical data to the Data Mailbox running on Talk2M servers. This historical data is temporarily stored and is then available using the DMWeb protocol. This HTTP based protocol allows third party applications to retrieve data from the Data Mailbox in an easy way.

For more information, please refer to eWON Developers website.

2.2.2. M2Web API

The M2Web API is explained in the present document. The M2Web API folder from the Talk2M SDK should contain:

- **This Reference Guide**

- **NET**

  This is actually a library that helps you create .NET (C#/VB) programs using the M2Web API. It was programmed using Visual Studio 2010. Open the Solution file (M2Web.sln) and read the contents of Example1/Program.cs to see how to use the library.

- **PHP**

  Sample M2Web Library source code. Delivered as M2Web API sample code.

3. General Principles

3.1. HTTPS API

The M2Web API is a set of HTTPS requests.

The API supports both *GET* and *POST* methods. API parameters can be put either through the query string part of the URL (*GET*) or through the request body (*POST*).

The API exposes services that can either return information about M2Web itself (account information, eWON connectivity status,...) or forward requests to eWONs (or eWON LAN devices).

3.1.1. GET or POST

Both GET and POST protocols are secure when using the M2Web API. Even so, we recommend using POST requests since they add another layer of security.

Get request is still a valid choice: GET is being used through SSL connections which assures a secure environment. But alternative methods can be used by hackers to steal credentials such as browser history, server logs… POST requests do resist better to those kinds of attack.

- Note -

In the following examples of this document, we use the GET method to ease the comprehension of the different queries but in practice POST requests should be used, especially when sending credential parameters such as t2mdevid, t2mpassword and t2mdevicepassword.

3.2. Stateless requests vs Sessions (aka stateful requests)

Programs must login exactly as a human user of M2Web would do which implies the creation of a session.

M2Web API users may choose between two session models:

1. The traditional login/logout session where a **sessionId** is returned by the login service. This **sessionId** must be in every subsequent request.

    Sessions are closed by calling the logout service or after an inactivity timeout.

    This is known as *stateful requests*.

2. Stateless requests: There is no login/logout. Instead of passing a session id to the web services, the consumer sends its credentials (account, username and password) in every call. This model frees the developer from maintaining sessions.

Both models allow using the same set of API web services. The consumer simply chooses one model or the other by passing either credentials or a session id to those web services.

3.3. API Request Structure

As a reminder to GET or POST chapter, we recommend using the POST query.

In order to ease the comprehension of the following examples, these ones are explained in the GET method.

3.3.1. URL

```
https://m2web.talk2m.com/t2mapi/...
```

All API calls start with **t2mapi/**

3.3.2. Request Parameters

API request parameters are passed as query string arguments. Their name starts with **t2m** in order to avoid collisions with application parameters that must be forwarded to the eWON.

All requests require a mandatory **t2mdeveloperid** parameter that identifies the 3rd party (not the T2M account!). Requests with missing or incorrect DevID are rejected by the M2Web server. To request a Talk2M Developer ID, please refer to Talk2M Developer Registration.

All requests require credential parameters (either a session id in case of stateful request or identity parameters in case of stateless or login requests).

3.4. API Response structure

3.4.1. Response

All the responses generated by the M2Web server itself (as opposed to responses sent by the eWON and forwarded by the M2Web server) are formatted in:

- JSON (http://www.w3schools.com/json/).
- MIME type = *application/json*
- Encoding = UTF-8

All responses contain a boolean **success** parameter.

Example of successful response (HTTP Status code = 200):

```
{
  "success" : true,
  "t2msession": "13abcd456"
}
```

3.4.2. Errors

In case of error, the response is sent with an HTTP status code that reflects the error.

The response created by the M2Web server (as opposed to responses sent by the eWON and forwarded by the M2Web server) indicate errors as follows:

1. HTTP Status Code indicates error (!= 2xx)

2. The response body consists in a JSON object that contains detailed error information:

   • "success" = false

   • "code" = x (Same as HTTP Status code)

   • "message" = "human readable error message"

Example of error response (HTTP Status code != 2xx):

```
{
  "success": false,
  "code": 403,
  "message": "Invalid credentials"
}
```

The full list of error codes is documented in the Appendix 1.

3.5. Stateless Requests

Those requests do the following in one single HTTPS request:

   • Log in M2Web and create the session.

   • Send the request to the eWON (or to the portal, or to the device behind the eWON) and return the reply.

   • Close the session.

Any M2Web request can be turned into a stateless request by adding the following parameters in the query string:

```
?t2maccount=...&t2musername=...&t2mpassword=...?
```

Those parameters are caught on-the-fly by M2Web and removed from the query string forwarded to the eWON. M2Web validates those parameters and forwards the request to the eWON as if there were a session cookie.

- Note -

Those parameters must be encoded as UTF-8 strings.
The way to "percent encode" a UTF-8 character is to encode each bytes of its UTF-8 sequence (e.g.: "é" becomes "%C3%A9", "ω" becomes "%E2%8D%B5"

3.6. Session Management (aka stateful requests)

When the session model is used (as opposed to stateless requests), the session id obtained from the *login* API call must be passed as a query string parameter to every subsequent API call:

```
?t2msession=...?
```

3.7. Talk2M Developer ID

In addition to Talk2M user credentials, all API calls must contain the t2mdeveloperid=... query string parameter that identifies the program author as a registered Talk2M developer.

The Talk2M Developer ID is a unique string received upon registration. From more information, please refer to [Talk2M Developer Registration](#) section.

DevID example: 21EC2020-3AEA-1069-A2DD-08002B30309D

- Note -

The M2Web API uses the GUID notation without curly braces.

Every M2Web API call must contain this DevID as a query string parameter (in addition to other query string parameters such as credentials):

```
?t2mdeveloperid=...?
```

3.8. No rewriting of eWON response

The regular M2Web service rewrites absolute links in HTTP responses (HTML, CSS,...) returned by eWONs to make those links work through M2Web.

Since the M2Web API is aimed at providing machine-to-machine communication rather being used by a browser, there is no such rewriting mechanism.

4. Services

All URLs start with

```
https://m2web.talk2m.com/t2mapi/
```

In order to simplify the notation in this document, we no longer indicate the https protocol and hostname part of the URL.

4.1. Session management

4.1.1. Login

This service is similar to the existing one used by the M2Web UI. However, the response is M2M-oriented: It returns a success/error indication (+session id) rather than redirecting to the account page.

Request:

```
t2mapi/login?t2maccount=...&t2musername=...&t2mpassword=...
```

Response:

```
{
  "success": true,
  "t2msession": "123abc456"
}
```

or

```
{
  "success": false,
  "code": 403,
  "message":"Invalid credentials"
}
```

- Note -

The values for t2msession is an example. Its actual value of course varies from call to call.

The session can be used only from the IP address that created it.

4.1.2. logout

Closes an existing stateful API session.

- Note -

Stateless requests do not require to logout.

Request:

```
t2mapi/logout?t2msession=...
```

Response:

```
{
  "success": true
}
```

- Note -

success can be false such as in case of a wrong t2msession value. See above for the structure of a response that indicates an error.

4.2. Portal Services

All requests must contain either a *t2msession* parameter (stateful request) or the *t2maccount*, *t2musername* and *t2mpassword* credentials parameters (stateless request). In addition they must also contain the *t2mdeveloperid* parameter.

4.2.1. getaccountinfo

- Retrieves basic account information (reference, name, company).
- Retrieves the set of pools visible by user : pairs name/id
- Retrieves the name of each custom attribute.

Request:

```
t2mapi/getaccountinfo?t2msession=...
```

- Note -

As all API calls, it is also available as a stateless call:

```
t2mapi/getaccountinfo?t2maccount=...&t2musername=...&t2mpassword=...
```

Response:

```
{
  "success": true,
  "accountReference": "8435",
  "accountName": "ACME",
  "company": "ACME, Inc.",
  "customAttributes": [ "attribute1", "", "third attribute" ],
  "pools": [
      { "id": 8735, "name": "site 1" },
      { "id": 9541, "name": "site 2" },
      { "id": 723, "name": "devices" }
      ],
  "accountType": "Pro"
}
```

There are always 3 custom attributes listed. Some or all of them may be empty.

The "accountType" attribute will either be "Free" (for non paying account) or "Pro" (for paying account).

4.2.2. getewons

Retrieves the set of eWONs visible by user along with their properties: Displayable names, link names, status, description, the 3 custom attributes, preferred m2web server hostname (currently always m2web.talk2m.com).

Request:

```
t2mapi/getewons?t2msession=...[&pool=...]
```

where *pool* is the optional numerical id of the pool whose eWONs should be retrieved. Pool

id's are retrieved using a **getaccountinfo** call.

If *pool* is not specified, all visible eWONs are listed in the response.

Response:

```
{
  "success": true,
  "ewons": [
      { "id": 14235,
        "name": "IP Camera",
        "encodedName": "IP%20Camera",
        "status": "online",
        "description": "IP Camera displaying Hall 47",
        "customAttributes": [ "Omron CJ1G", "LAN", "192.168.140.3" ],
        "m2webServer": "m2web.talk2m.com"
      },
      { "id": 14758,
        "name": "Robot-T800",
        "encodedName": "Robot-T800",
        "status": "offline",
        "description": "Robot – 1st generation terminator",
        "customAttributes": [ "Arnold", "", "No problemo" ],
        "m2webServer": "m2web.talk2m.com"
      }
      ]
}
```

- *encodedName* is the URL-friendly version of name intended to be used in the *get* service (see below).
- *m2webServer*: hostname to be used when using the get service to retrieve information from this eWON. (As of this writing, the value is always *m2web.talk2m.com* but is expected to change in the future to reflect the worldwide distributed Talk2M servers farm).

4.2.3. getewon

Retrieves the same set of information as *getewons* but is limited to one single eWON.

Request:

```
t2mapi/getewon?name=...&t2msession=...
```

where **name** is the name of the eWON.

Another possibility to identify the eWON is to use the *id* parameter instead of *name*. The

eWON Id is returned by *getewons*.

Response:

```
{
  "success": true,
  "ewon" : {
       "id": 14235,
       "name": "IP Camera",
       "encodedName": "IP%20Camera",
       "status": "online",
       "description": "IP Camera displaying Hall 47",
       "customAttributes": [ "Omron CJ1G", "LAN", "192.168.140.3" ],
       "m2webServer": "m2web.talk2m.com"
     }
}
```

4.2.4. wakeup

Sends a wake-up SMS to the eWON.

The eWON to wake-up is indicated either through its name or its id.

Request:

```
t2mapi/wakeup?name=...&t2msession=...
```

or

```
t2mapi/wakeup?id=...&t2msession=...
```

where *name* is the name of the eWON or *id* is the eWON id returned by *getewons*.

Response:

```
{
  "success": true
}
```

4.2.5. sendoffline

Sends a GPRS eWON offline.

The eWON is indicated either through its name or its id.

Request:

```
    t2mapi/sendoffline?name=...&t2msession=...
```

or

```
    t2mapi/sendoffline?id=...&t2msession=...
```

where *name* is the name of the eWON or *id* is the eWON id returned by *getewons*.

Response:

```
{
  "success": true
}
```

4.3. Reaching eWONs

4.3.1. URL

The URL used to reach an eWON page either using a GET or POST is different from the browser-based M2Web URL scheme used when browsing the eWON (stateful):

```
  t2mapi/get/<ewon name>/<path>?<query>&t2msession=...
```

or (stateless)

```
  t2mapi/get/<ewon name>/<path>?<query>&t2maccount=...
```

The URL part as of *<path>* is forwarded to the eWON. All *t2mxxx* parameters are deleted from the requested URL before forwarding it to the eWON.

Reaching devices behind the eWONs is done using the same URL syntax as in a browser:

```
  t2mapi/get/<ewon name>/proxy/<deviceIp>/<path>?<query>&t2msession=...
```

- Note -

> - *This get service is used for both GET and POST requests.*
>
> - *As indicated in a former paragraph, the eWON response is not rewritten. It is therefore important to make sure that the eWON response does not contain any absolute URL.*

4.3.2. Hostname

The hostname for M2Web is

```
https://m2web.talk2m.com
```

However, the Talk2M servers farm contains several M2Web servers. Talk2M decides which server should be used to reach each eWON. Its hostname is contained in the *m2webServer* property of the *getewon* and *getewons* services documented above. The default value remains *m2web.talk2m.com* but may change at any time in the future.

Example:

```
"m2WebServer" = "us.m2web.talk2m.com"
```

→ The URL to reach the eWON becomes:

```
https://us.m2web.talk2m.com/t2mapi/get/<ewon name>/<path>?<query>&t2maccount=...
```

- Note -

> - *M2Web does not use HTTP redirections to handle such redirections: Not all client-side programs technologies support HTTP redirections.*
>
> - *Sessions cannot be used across servers. A program that uses sessions (aka stateful requests) must login again on the other M2Web server (us.m2web.talk2m.com in this example) before it uses the get service.*
>
> - *The m2webServer is a per-eWON setting. It may change at any time but is not likely to change often though since it is related to criterias such as the geographical location of the eWON and the account type (Free vs Pro).*

4.3.3. eWON Authentication

Retrieving information from eWONs requires to login in the eWON.

The **get** API accepts 2 additional parameters to create the eWON login:

```
?...&t2mdeviceusername=...&t2mdevicepassword=...
```

M2Web will create the appropriate *Authorization* HTTP header for the given *t2mdeviceusername* & *t2mdevicepassword* parameters.

5. eWON Services

eWON web services exist in many different flavors, such as

- Standard web services, eg *UpdateTagForm*

- Standard data download using export block descriptors

- Custom web pages using SSI tags. Those pages can be used to read and/or write data to the eWON

Those services help retrieve or update various kinds of information, including:

- eWON configuration, such as the list of tags.

- Instantaneous tag values.

- Tags history.

- Alarms definition and state. Acknowledgment of alarms.

- Remote execution of scripts.

Detailed information about those techniques can be found in the eWON Web Reference Guide available on developer.ewon.biz

5.1. Examples of eWON web services used through M2WebAPI

5.1.1. Read all instantaneous Tag values

Request :

```
https://m2web.talk2m.com/t2mapi/get/MyeWON/rcgi.bin/ParamForm?
AST_Param=$dtIV$ftT&t2maccount=MyAccounts&t2musername=MyName&t2mpassword=MyPassword&t
2mdeveloperid=61e9ddf4-6ade-493e-ad4e-
a79dde788140&t2mdeviceusername=MyeWONUser&t2mdevicepassword=MyeWONPassword
```

This request allows to retrieve the instantaneous values of all your Tags in the form of a CSV file. Actually, an Export Block Descriptor is used in the request. See 6.1.3

Response :

```
"TagId";"TagName";"Value";"AlStatus";"AlType";"Quality"
1;"tank";0;0;0;65472
2;"Flow";0;0;0;65472
3;"FtpTrigger";0;0;0;65472
```

5.1.2. Update (Write) Tag value

Request :

```
https://m2web.talk2m.com/t2mapi/get/MyeWON/rcgi.bin/UpdateTagForm?
TagName1=Tank&TagValue1=12&t2maccount=MyAccount&t2musername=MyName&t2mpassword=MyPass
word&t2mdeveloperid=61e9ddf4-6ade-493e-ad4e-
a79dde788140&t2mdeviceusername=MyeWONUser&t2mdevicepassword=MyeWONPassword
```

The request aims to update the Tag "Tank" with the value 12. Several Tags can be updated in a single call using parameters "TagName2, TagValue2, TagName3, TagValue3,...

Response :

```
<td>
Done.
</td>
```

5.1.3. Retrieve Export Block Descriptor

The export block descriptor feature of eWON allows retrieving plenty of different data such as the Tags List, the eWON Configuration, the list of Alarms, The Historical Data...

Export Block Descriptors are explained in the General Reference Guide available on http://wiki.ewon.biz/Support/07_Documentations/01_RG

An Export Block Descriptor helper allowing to generate easily an Export Block Descriptor Syntax is available on https://ewonsupport.biz/ebd/

Request :

```
https://m2web.talk2m.com/t2mapi/get/MyeWON/rcgi.bin/ParamForm?
AST_Param=$dtHL$ftT$et_0$st_m1&t2maccount=MyAccount&t2musername=MyName&t2mpassword=My
Password&t2mdeveloperid=61e9ddf4-6ade-493e-ad4e-
a79dde788140&t2mdeviceusername=MyeWONUser&t2mdevicepassword=MyeWONPassword
```

This request aims to retrieve the Historical data (the last minute) from eWON.

Response :

```
"TagId";"TimeInt";"TimeStr";"IsInitValue";"Value";"IQuality"
36;1422015320;"23/01/2015 12:15:20";0;0;3
35;1422015321;"23/01/2015 12:15:21";0;0;3
36;1422015321;"23/01/2015 12:15:21";0;0;3
35;1422015322;"23/01/2015 12:15:22";0;0;3
36;1422015322;"23/01/2015 12:15:22";0;0;3
35;1422015323;"23/01/2015 12:15:23";0;0;3
36;1422015323;"23/01/2015 12:15:23";0;0;3
```

```
35;1422015324;"23/01/2015 12:15:24";0;0;3
36;1422015324;"23/01/2015 12:15:24";0;0;3
...

36;1422015378;"23/01/2015 12:16:18";0;0;3
35;1422015379;"23/01/2015 12:16:19";0;0;3
36;1422015379;"23/01/2015 12:16:19";0;0;3
35;1422015380;"23/01/2015 12:16:20";0;0;3
```

5.1.4. Execute Basic Script on eWON.

Request :

```
https://m2web.talk2m.com/t2mapi/get/MyeWON/rcgi.bin/ExeScriptForm?Command1=SENDSMS
"0479565656,gsm,0","MessageToSend"&t2maccount=MyAccount&t2musername=MyName&t2mpasswor
d=MyPassword&t2mdeveloperid=61e9ddf4-6ade-493e-ad4e-
a79dde788140&t2mdeviceusername=MyeWONUser&t2mdevicepassword=MyeWONPassword
```

Several Commands can be sent in a single call using the parameters "Command2, Command3,..."

Response :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Success Message</title>
<link rel="styleSheet" type="text/css" href="../lay/_gz_ewon_css_1_1.css"/>

</head>
<body>
<table cellspacing="15" class="res_ok"><tr><td>
1 command(s) of 1 executed.
</td></tr></table>
</body>
</html>
```

Appendix I – Error codes

The table below lists all error codes and messages that can be returned by M2Web, as described in paragraph API Response Structure / Errors:

Code	HTTP Status	Message	Context (API)
400	Bad Request	Service Unavailable	get (with proxy)
400	Bad Request	Missing argument	all
403	Forbidden	Invalid method	N/A
403	Forbidden	Invalid session ID	All (stateful)
403	Forbidden	Invalid credentials	All (stateless)
403	Forbidden	Permission denied	get
410	Gone	Device not found	get/getEwons
429	Too Many Requests	Too many concurrent web connections	All (stateful)
500	Internal Server Error	Internal server error	all
500	Internal Server Error	Unexpected exception	all
502	Bad Gateway	Error while getting proxy to remote device (behind the eWON)	get (with proxy)
502	Bad Gateway	Error while proxying request	get
503	Service Unavailable	Error while sending device offline	sendOffline
503	Service Unavailable	Error while sending device offline	wakeUp

Revision

Revision History

Revision Level	Date	Description
1	23/09/2013	Initial version
2	13/11/2013	Clarified Developer ID
3	03/07/2014	Account Type property added in chapter 5.2.1
4	21/01/2015	APIs examples added
1.2	31/08/2016	Version adjustment to match SDK version
1.3	02/02/2018	Version adjustment to match SDK version

Document build number: 58

Note concerning the warranty and the rights of ownership:

The information contained in this document is subject to modification without notice. Check https://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 the manufacturer's responsibility could be called for direct, indirect, accidental or other damage occurring from any defect of the product of errors 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.

HMS Industrial Networks sa