PoolLab2 WSAPI Reference

Water-i.d. GmbH

January 3, 2024

Contents

1 Introduction

This document serves as reference for the PoolLab2's Webrequest-over-WiFi Synchronization API (WSAPI). It is intented for developers aiming to integrate the PoolLab2 into their custom Cloud webserver, to automatically receive measurement data from their PoolLab2 without using the `LabCom Cloud Service`.

2 The WiFi Interface

The PoolLab2 has a built-in WiFi 4 capable radio, which can be used to *update the firmware and device database* as well as *synchronize source and measurement data* with a webserver. Before any of the WiFi functions can be used with the device, a one-time setup via Bluetooth LE API is required. At minimum, the WiFi Access Point's SSID and password must be set.

2.1 User Interaction

The radio is disabled by default. The WiFi-interface must be enabled on-device, by the end-user. A key on the keypad-foil indicates which key to press to start the **WiFi-process**. The WiFi radio is disabled automatically after the process completes.

3 WiFi and Network requirements

3.1 Version and Security

The PoolLab2 requires a WiFi 4 Access Point, configured with WPA2 security and a password of max. 32 characters. Unsecured networks and empty-passwords are not supported.

3.2 (B)SSID

The access point is not required to broadcast it's SSID. However, the SSID must not be empty and must contain only ASCII characters.

3.3 IP / DHCP

The access point is required to automatically assign an IPv4-address to the PoolLab2 by providing DHCP. Static IP addresses are currently not supported.

3.4 NTP

As step of the **WiFi-process**, the firmware periodically tries to synchronize it's internal clock with a time-server. The access point may provide network time to the PoolLab2 via DHCP option 004 and 042. If not provided, the PoolLab2 will try to connect with *time.nist.gov*.

4 Server / Cryptography Requirements

A webserver capable of serving a valid **Cloud API** or **Update API Endpoint** must provide a secure websocket connection.

4.1 Socket

The server must provide a TCP/IPv4 socket connection on port 443.

4.2 SSL

The firmware requires an API Endpoint to provide TLS/SSL security. The server must provide encryption based on a valid SSL certificate that matches the URL under which it is requested. The certificate must be signed by a Trusted Root Certificate. Currently, the firmware provides a set of Trusted Root Certificates as extracted from the Mozilla CA certificate store, plus the `Let's Encrypt Root Certificate`.

4.3 Websocket

Useful data is wrapped in Websocket/HTTP Protocol, i.e. GET/POST-requests. The **Update API Endpoint** must support the *Range* HTTP request header.

5 Configuration

Before the PoolLab2's WiFi interface can be used, some configuration must be done by using the Bluetooth LE API. A separate document, `PoolLab2 BLE API Reference`, explains the bluetooth API in more detail. This section gives only an overview of what can be configured, see the aforementioned document for how to set the configuration.

5.1 WiFi Logins

A pair of a WiFi Access Point SSID and password. See `PoolLab2 BLE API Reference` [5.17 SET WIFI LOGINS Command].

5.2 Update API Path

The URL where a self-update endpoint is hosted. See `PoolLab2 BLE API Reference` [5.9 SET UPDATE API PATH Command]. The default points to the stable-release channel of the original manufacturer of the device [Water-i.d. GmbH].

5.3 Cloud API Path

The URL where a data-synchronization ("Cloud") endpoint is hosted. See `PoolLab2 BLE API Reference` [5.10 SET CLOUD API PATH Command]. The default points to the "LabCom Cloud" (https://cloud-synchronizer.labcom.cloud/api/v1).

5.4 Cloud API Logins

The username and password to login with the configured Cloud API endpoint. See `PoolLab2 BLE API Reference` [5.18 SET CLOUD LOGIN Command]. For the "LabCom Cloud", you can register an account for free at https://labcom.cloud/ and use these logins for the PoolLab2.

6 The WiFi-process

When the end-user starts the **WiFi-process**, the device firmware follows the below steps:

Step	**Comment**
Battery Check	*Aborts if the battery level is too low to reliably activate the radio interface*
Radio Power On	*Activates the built-in radio*
WiFi AP Scan	*Scan for the configured access point*
WiFi Connect	*Connect to the configured access point*
DHCP Handshake	*Request and negotiate IPv4 address with gateway*
NTP Update	*Synchronizes internal clock with the gateway's NTP server*
Cloud Synchronization	*Only if valid login data are configured*
Update Check	*Downloads available firmware and database version info*
Update Dialog	*On-screen confirmation dialog for end-user*
Database Update	*Self-update the chemistry database*
Firmware Update	*Self-update the firmware*

Table 1: WiFi-process Steps

7 API Endpoints

During the **WiFi-process**, the firmware utilizes different `API Endpoints`. The end-user can configure the URL of the endpoints, i.e. to synchronize measurement data with a custom data system. See section **Configuration** for more information on how to change the `API Endpoints` addresses.

7.1 Update API Endpoint

By changing the **Update API Endpoint** address, the customer can redirect the firmware's Update-Check and Self-Update to a custom server. However, no custom/modified firmware can be installed on the device, as the bootloader will reject unsigned binaries. The intention of allowing a custom **Update API Endpoint** is to enable customers to run locally-cached copies of the manufacturers original Endpoint.

The factory-default value is `[cloud-synchronizer.labcom.cloud/api/v1]`. The `https://` prefix is automatically prepended by the firmware and must not be included when changing to a new URL. This is the **base-url** for the **Update API Endpoints**.

7.1.1 /checkupdate Endpoint Request

Endpoint URL: `[base-url]`**/checkupdate**
Request Type: `POST`
Parameters: **oem_id**, **hwrev**, **fw_version** (`x-www-form-urlencoded`)

oem_id will be set to an integer value indicating the OEM version of the firmware. Note that each OEM version has a different firmware file, so the server is required to process this variable to offer the correct firmware download URL in response.

hwrev will be set to an integer value indicating the hardware revision of the device making the request. Again the server must process this variable to provide the matching firmware image file download URL.

fw_version will be set to the current firmware version of the device making the request. The server must process this variable to limit the database version offered as update to the device. For more information, please send a request to leon.hock@water-id.com .

7.1.2 /checkupdate Endpoint Response

If a server-side error occurs, any http-error code can be returned to abort the process device-side. If the request is malformed (missing any parameter, parameter of wrong type, etc), the server shall respond with http code `200 OK` and a `body` content set to [`S:ERR`] (without brackets). This will cause the upgrade process to be aborted, and the firmware will show the error-code for "update protocol error".

On success, the server shall respond with http code `200 OK` and a `body` content set to [`S:OK;`**A**`;`**B**`;`**X**`;`**Y**] (without brackets).
Where **A** and **B** are integers indicating (**A**) the `versioncode` of the available firmware update as well as (**B**) the `versioncode` of the available database update.

X must be a fully-qualified URL to (**X**) the firmware update file matching the requested `hardware revision` and `OEM ID` as well as the `versioncode` delivered with the response.

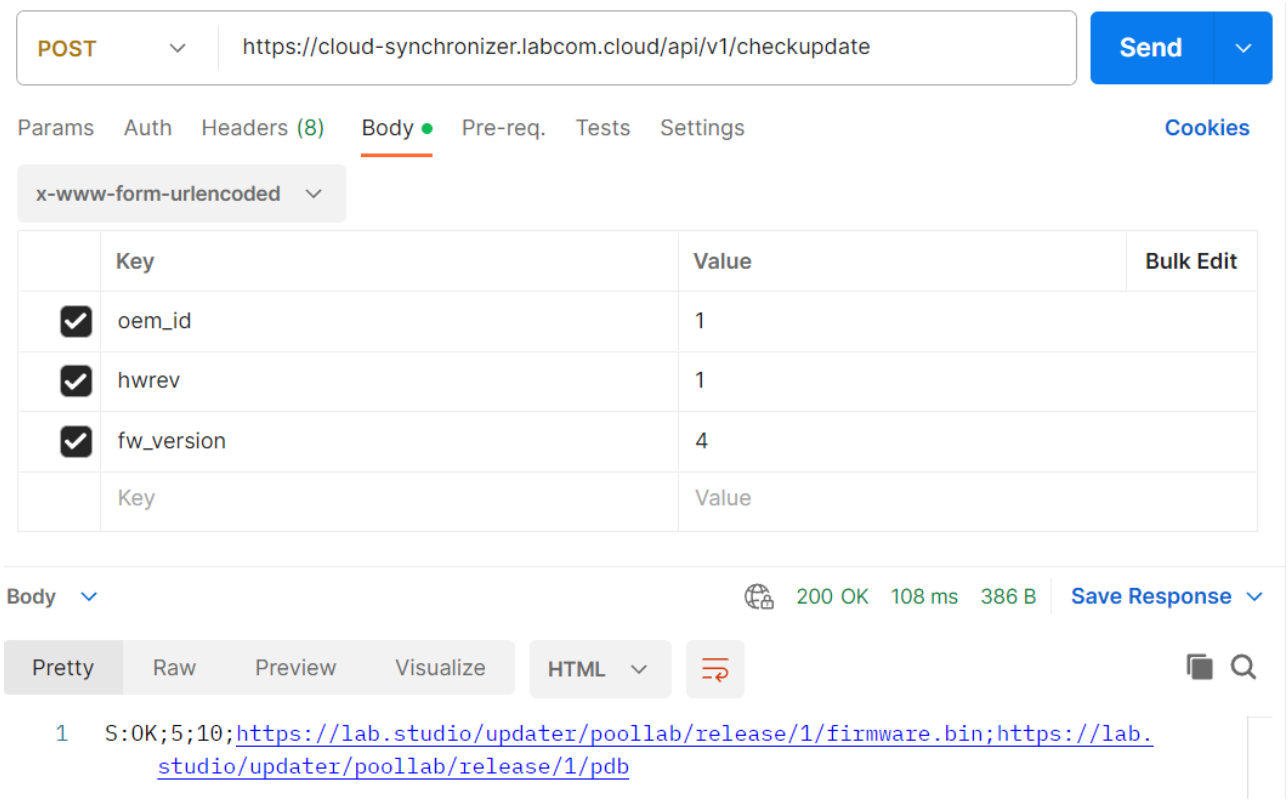
Y must be a fully-qualified URL to the `database update repository` matching the requested `hardware revision` and `OEM ID` as well as the database `versioncode` given in the response. This URL is used as **database-base-url**. The firmware appends several tokens to this path under which it expects to reach http-file-downloads of the update files.

Filepath/Filename	Comment
[database-base-url][/**pdb_parameter.txt**]	*Parameter file*
[database-base-url][/**tlist_0_tab.txt**]	*T-Menu 0*
[database-base-url][/**tlist_1_tab.txt**]	*T-Menu 1*
[database-base-url][/**tlist_2_tab.txt**]	*T-Menu 2*
[database-base-url][/**tlist_0_liq.txt**]	*T-Menu (liq) 0*
[database-base-url][/**tlist_1_liq.txt**]	*T-Menu (liq) 1*
[database-base-url][/**tlist_2_liq.txt**]	*T-Menu (liq) 2*
[database-base-url][/**tlist_1_glass.txt**]	*T-Menu (glass) 1*
[database-base-url][/**pdb_units.txt**]	*Units file*
[database-base-url][/**pdb_curves.txt**]	*Curves file*

Table 2: Database-Update Files

The structure and content of these files are not publicly documented. Developers wanting to provide locally-cached copies of the update-server are recommended to provide a carbon-copy of the directory and file structures found on the official update server (using the factory-default Endpoint URL).

7.1.3 /checkupdate Endpoint Postman Example



7.2 Cloud API Endpoint

The PoolLab2 can synchronize it's measurement data and *Sources* with a webserver by the push of a button. By default, this webserver is Water-i.d.'s free **LabCom Cloud** (https://labcom.cloud). By changing the URL of the **Cloud API Endpoint**, the end-user can configure his PoolLab2 to synchronize the measurement data with a custom webserver / custom cloud service.

The factory-default value is `[cloud-synchronizer.labcom.cloud/api/v1]`. The `https://` prefix is automatically prepended by the firmware and must not be included when changing to a new URL. This is the **base-url** for the **Cloud API Endpoints**.

7.2.1 /putmeasure Endpoint Request

This `Cloud API Endpoint` is used by the firmware to upload saved *Measurement Data* from the device. Each measurement record is uploaded within it's own request, therefore it is strongly recommended to use a server that supports `HTTP persistant connection`, also known as `HTTP keep-alive`, to bundle multiple requests within a single socket connection.

Endpoint URL: `[base-url]`**/putmeasure**
Request Type: `POST`
Parameters: **email**, **pass**, **serial**, **scenario_id**, **parameter_id**, **value**,
time_taken, **account_id**, **account_uuid** (`x-www-form-urlencoded`)

The **email** and **pass** parameters will be set to the cloud account login as configured on the device. A custom server may choose to ignore these values, however the firmware requires it's internal configuration to have non-empty values for the login email and passwords. Otherwise the firmware considers no logins to be configured, and does not attempt cloud data synchronization.

The **serial** parameter will be set to the text-representation of the serial-number of the device making the request.

The **scenario_id** and **parameter_id** parameters indicate which type of measurement was made. For more information, please contact leon.hock@water-id.com.

The **value** parameter will be the floating-point formatted representation of the measurement value. A value of `1000000.0` represents a range-overflow, or so called "Overrange" error. A value of `-1000000.0` represents a range-underflow, or so called "Underrange" error.

The **time_taken** parameter is the epoch-timestamp (UTC) when the measurement was taken.

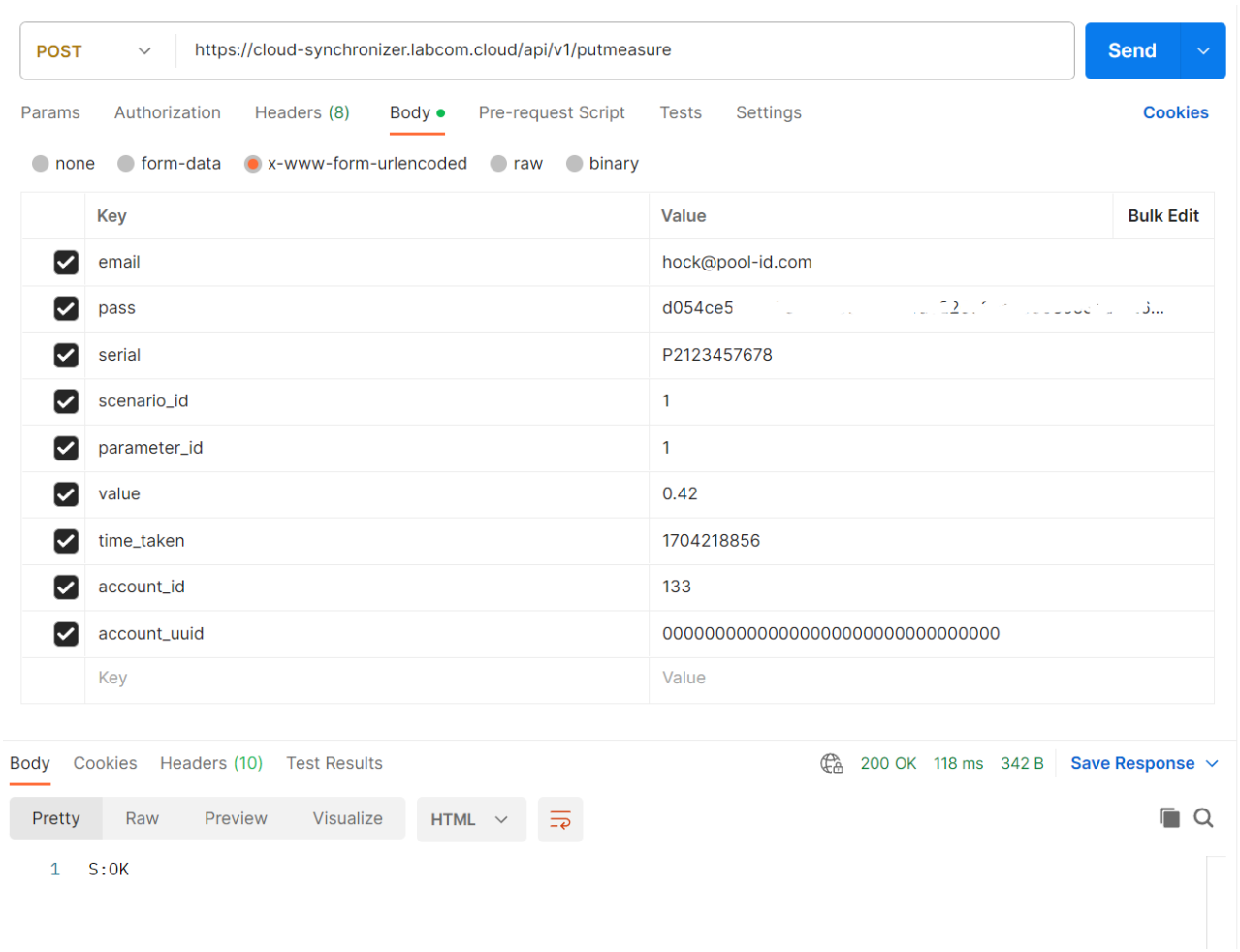
The **account_id** and **account_uuid** parameters are set to the *Source ID* and *Source UUID* values of the *Source* that was selected when this measurement was taken.

7.2.2 /putmeasure Endpoint Response

If a server-side error occurs, any http-error code can be returned to abort the process device-side. If the request is malformed (missing any parameter, parameter of wrong type, etc), the server shall respond with http code `200 OK` and a `body` content set to [`S:ERR`] (without brackets). This will cause the synchronization process to be aborted, and the firmware will show the error-code for "cloud protocol error".

On success, the server shall respond with http code `200 OK` and a `body` content set to [`S:OK`] (without brackets).

7.2.3 /putmeasure Endpoint Postman Example



7.2.4 /getsources Endpoint Request

This `Cloud API Endpoint` is used by the firmware to synchronize the list of *Sources* saved on the device. A *Source* is one of the on-device user-selectable items, used to organize measurement data by location or customer. The firmware will always synchronize measurement data first, and the *Sources* second, so that there are no measurement data with unreferenced *Source IDs*, in case one or more *Sources* is removed between synchronizations.

A *Source* consists of a numerical (32bit) `Source ID`, a 32-byte `Source UUID`, and a 23-Character (ASCII) `Source Name`. Only the `Source Name` is displayed on device.

Endpoint URL: `[base-url]/getsources`
Request Type: `POST`
Parameters: **email**, **pass**, **serial** (`x-www-form-urlencoded`)

The **email** and **pass** parameters will be set to the cloud account login as configured on the device. A custom server may choose to ignore these values, however the firmware requires it's internal configuration to have non-empty values for the login email and passwords. Otherwise the firmware considers no logins to be configured, and does not attempt cloud data synchronization.

The **serial** parameter will be set to the text-representation of the serial-number of the device making the request. The `LabCom Cloud` is using the serial number to allow to configure different sets of sources for different devices while using the same `Cloud Account` login data. A custom server may choose to ignore this value.

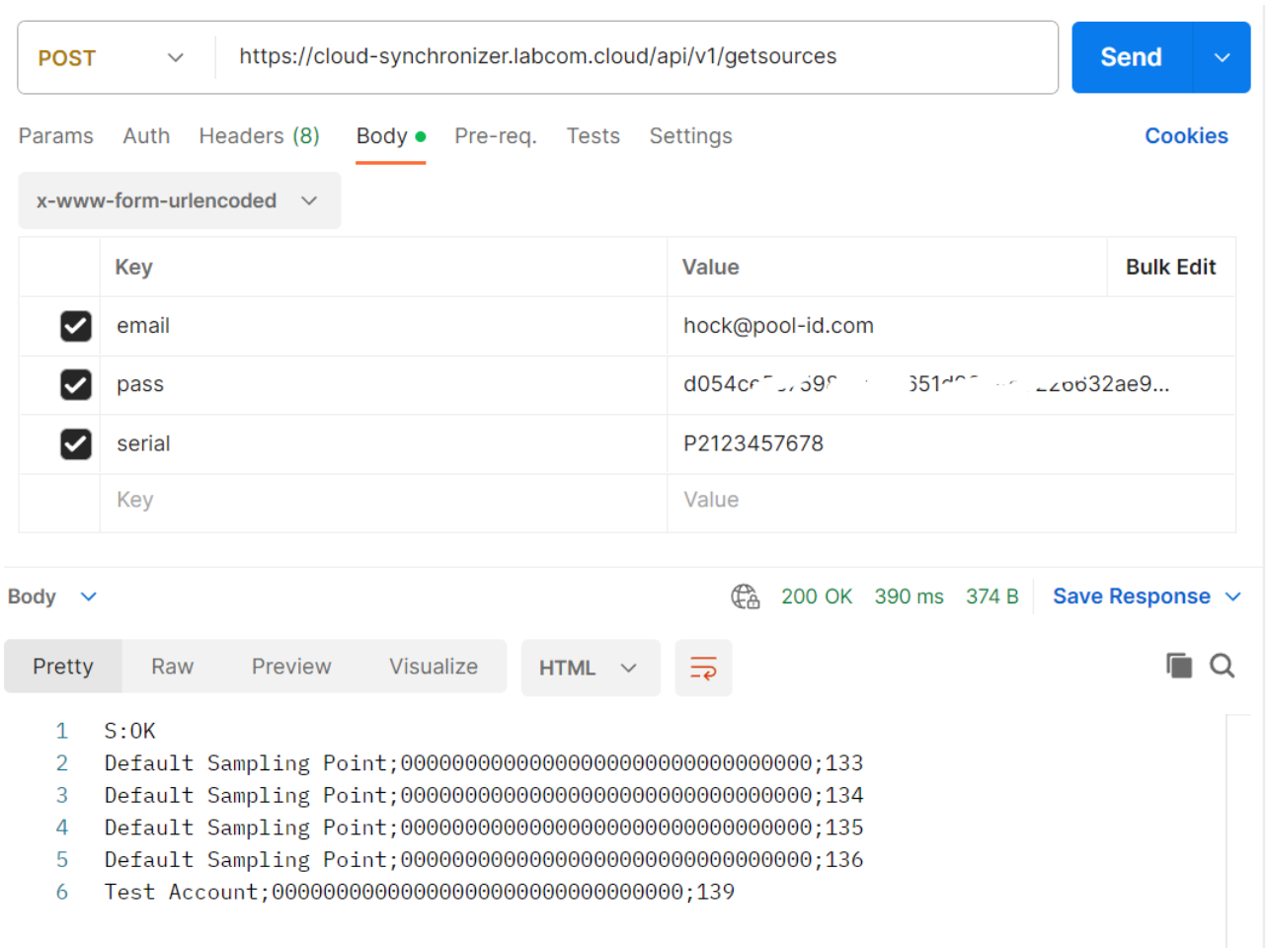
7.2.5 /getsources Endpoint Response

If a server-side error occurs, any http-error code can be returned to abort the process device-side. If the request is malformed (missing any parameter, parameter of wrong type, etc), the server shall respond with http code `200 OK` and a `body` content set to [`S:ERR`] (without brackets). This will cause the synchronization process to be aborted, and the firmware will show the error-code for "cloud protocol error".

On success, the server shall respond with http code `200 OK` and a `body` content set to the list of *Sources* to be saved on the device. The response shall begin with [`S:OK`] followed by a *newline-character* (**n**). Afterwards, the response shall be the list of all *Sources*. Each *Source* consists of 3 semicolon (**;**) separated values (*Source Name*;*Source UUID*;*Source ID*), followed by a *newline-character* (**n**). The last *Source* in the response shall **not** be followed by a *newline-character*.

Atleast one *Source* must be supplied, or the firmware will clear all current *Sources* and automatically add a *Default Source*.

7.2.6 /getsources Endpoint Postman Example



8 Document Revision

Version	Date	Author	Note
1	**03.12.2023**	**M.Opheiden**	*First Release*

Table 3: Document Revision History