





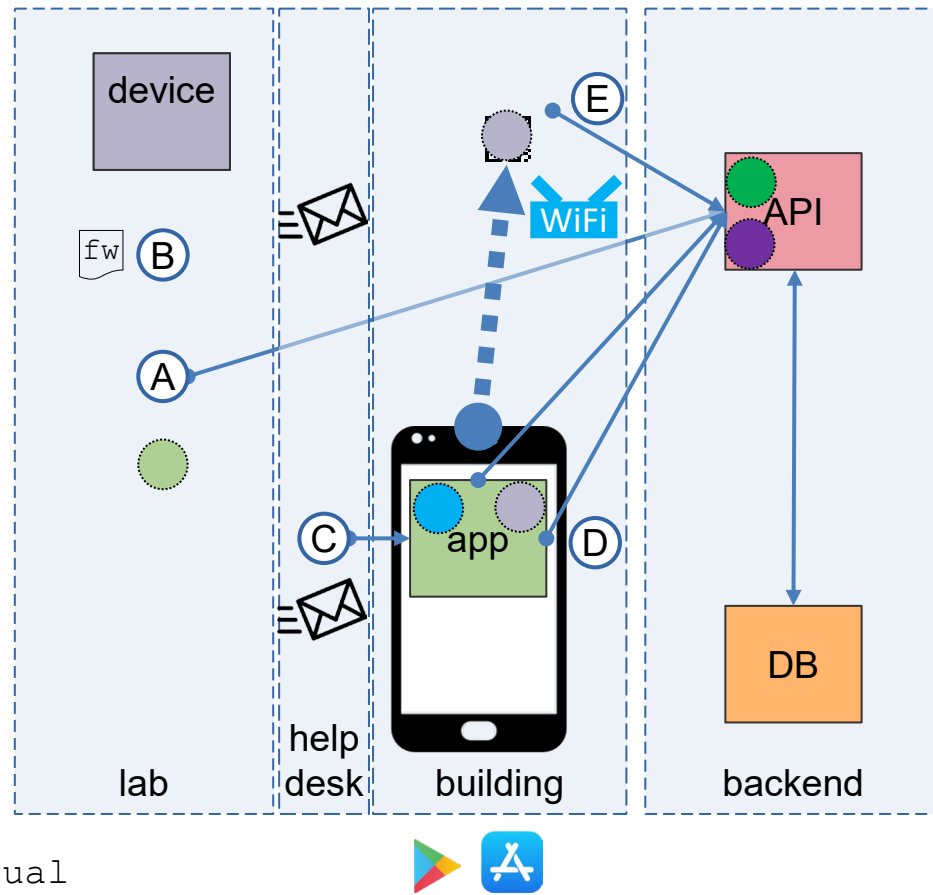


NeedForHeat Provisioning v2

- A. Account creation
- B. Device preparation & shipping
- C. App installation & account activation
- D. App device activation
- E. Backend device activation

token	activation	session
account		
device		

 Wi-Fi:SSID+pwd;
server:tst/prod
 installation manual









Entities and roles

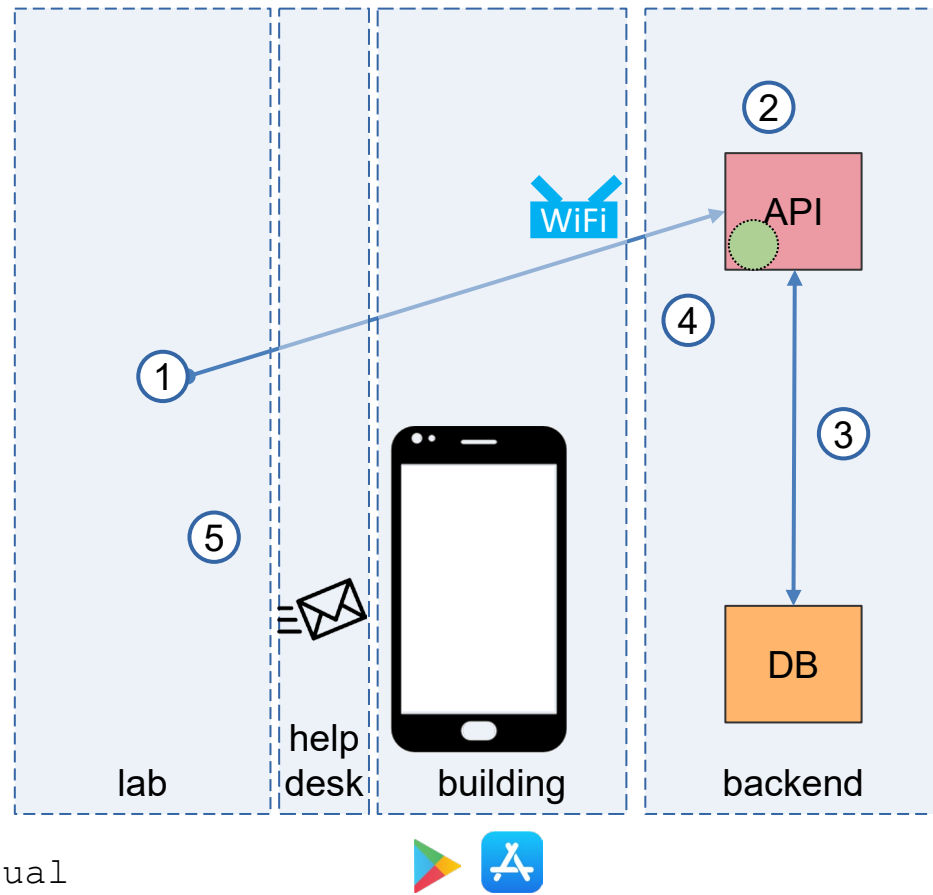
- **Resident**
 - a natural person, who participates in the research project as a subject, or who is a client of an energy advisor;
 - represented by an `account` in the API and database;
 - represented by a unique, random, secret, short-lived `account_activation_token`, used once to link an app to an account;
 - tokenized as a unique, random, secret, long-lived `account.session_token` identifying the `account` in calls from the app;
 - represented by a `pseudonym` in communication between helpdesk & researcher.
- **Device** (measurement device) :
 - represented by a `device` in the API and database;
 - represented by a unique, random, secret, long-lived `device.activation_token` that can be used once by the app and device to link the device to an account;
 - tokenized as a unique, random, secret, long-lived device `device.session_token` identifying the device in calls from the device;
- **App** (mobile app, e.g. WarmteWachter and NeedForHeat)
 - used by the subject to facilitate installing/activating/monitoring/ stopping devices on the resident's premises;
- **Backend: API + Database**
 - the server environment responsible for storing and managing accounts, devices and measurements;
- **Lab**
 - the place where devices preparation takes place;
- **Building**
 - the premises of a resident where device installation, activation and measurements take place;
- **Researcher**
 - A researcher who is and remains unaware of personally identifiable information that might enable him/her to trace back measurement data to a resident;
 - refers to resident only by `pseudonym` (a unique number, similar to an OpenID Connect pairwise subject identifier);
- **Helpdesk**
 - A contact person (or persons), who can be contacted by residents and who is (are) aware of (but does not disclose to third parties) the relation between a `pseudonym`, and personally identifiable information about the resident (e.g., name, e-mail address and/or street address).

(A) Account creation

1. Researcher calls POST on /account API endpoint; parameters
 - campaign: a measurement campaign name
2. API creates
 - account.activation_token, a unique, random token, to identify the account
3. API creates entity in account table in database, stores account.activation_token and retrieves parameter values for provisioning_url from the campaign table
4. API returns result of the POST on /account to researcher, with output
 - invitation_url.
 - at Windesheim, we use the general form
`https://<firebase_domain>/?link=https%3A%2F%2Faccount%2F<account_activation_token>&apn=<apn>&ibi=<ibi>&isi=<isi>&efr=1`
 with <parameter> replaced by the URL-encoded version of in the parameter value, e.g.
`https://energietransitiwindesheim.page.link/?link=https%3A%2F%2Faccount%2FQD3foH2AZPNabx5n1FMT-Ail3v5lm8r2RTH0oTfjvTg&apn=nl.windesheim.energietransitie.warmtewachter&ibi=nl.windesheim.energietransitie.warmtewachter&isi=1563201993&efr=1`
5. Researcher sends [(pseudonym, invitation_url)] to helpdesk





token	activation	session
account		
device		



-  Wi-Fi:SSID+pwd;
server:tst/prod
-  installation manual

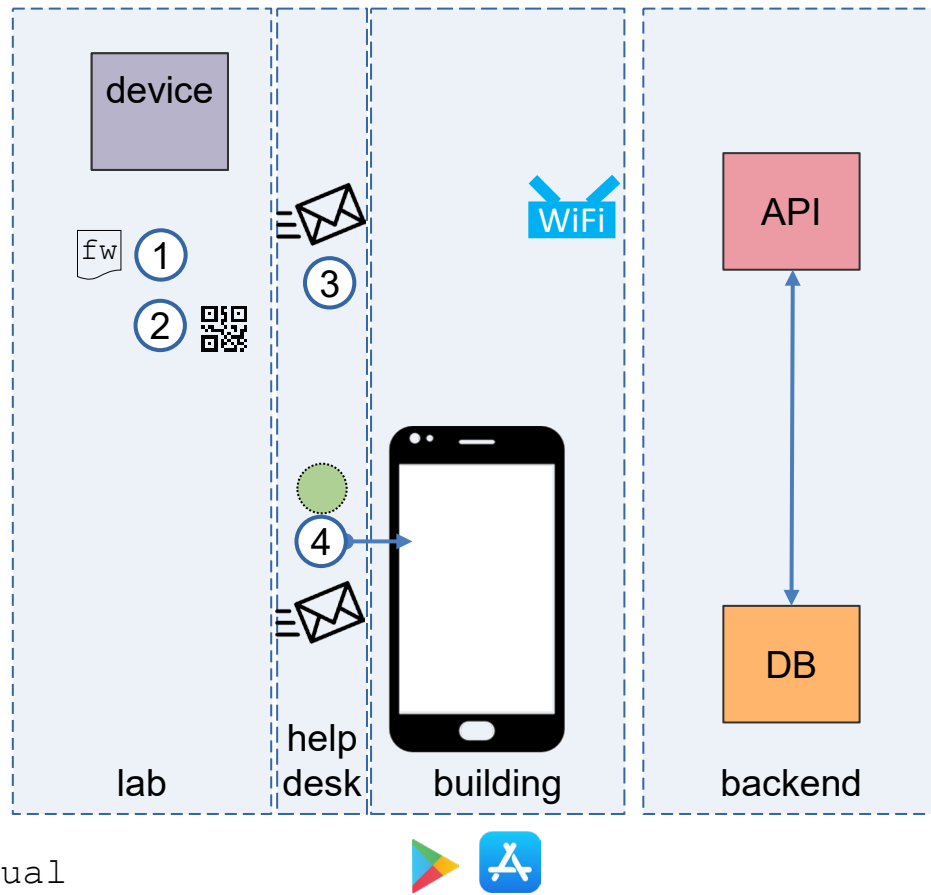


(B) Device preparation & shipping

1. Researcher uploads proper firmware to device
2. (only for devices without e-ink screen)
Researcher creates device-specific QR-code with offline QR-code generator, prints QR-code and attaches it to the (back of the enclosure of) the device; examples:
 - `{ "ver": "v1", "name": "9C0A-0D45DF", "pop": "810667973", "transport": "ble" }`
 - `{ "ver": "v1", "name": "9C0A-8E23A6", "pop": "516319575", "transport": "softap", "security": "1", "password": "516319575" }`
3. Devices are shipped (via helpdesk) to residents:
 - helpdesk defines sets of devices to be assembled per pseudonym, for researcher
 - researcher packages a set of devices per resident; labels sets with pseudonym
 - helpdesk relabels pseudonym with street address of the of the resident, ships
4. App invites are sent (by helpdesk) to residents:
 - helpdesk sends the provisioning_url to the proper resident by e-mail
 - helpdesk is responsible for associating the pseudonym with the proper resident (and his/her personally identifiable information)





token	activation	session
account		
device		



 Wi-Fi:SSID+pwd;
server:tst/prod
 installation manual

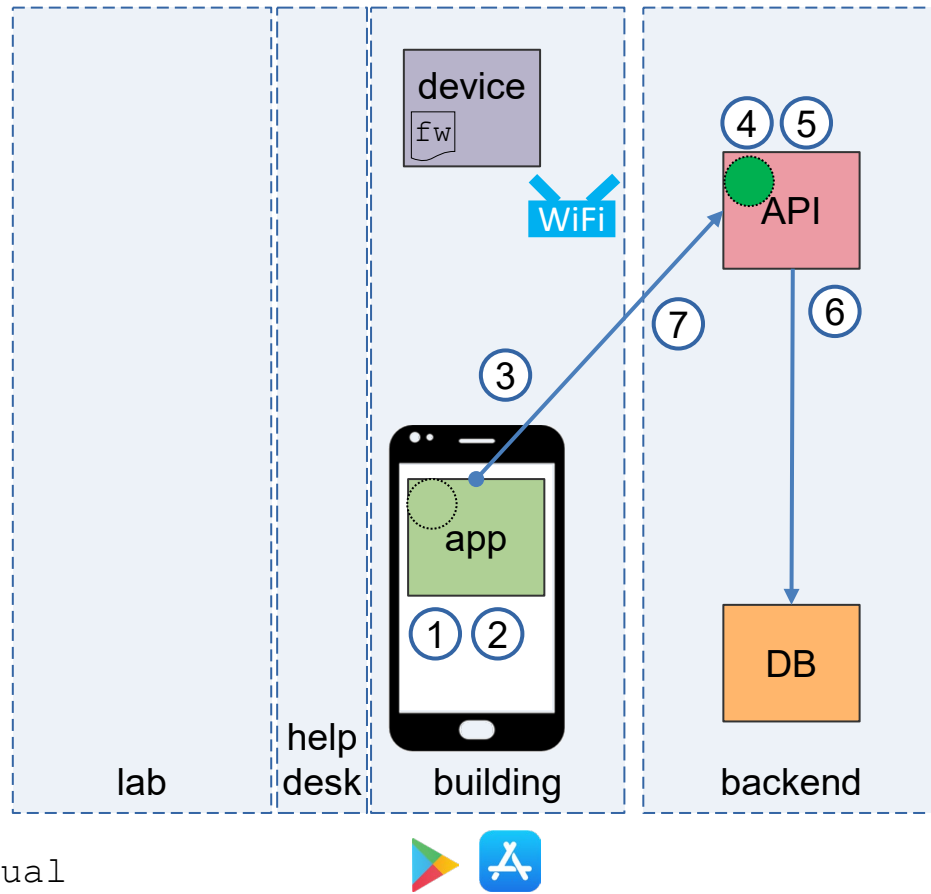


(C) App installation & account activation

1. Resident clicks Firebase Dynamic Link in e-mail on smartphone
2. Resident clicks 'install' and 'continue'; app installs & opens on smartphone
 - App reads `account.activation_token` from Firebase Dynamic Link
 - N.B.: this might involve two tokens, one for test server and one for production server
3. App calls [POST on /account/activate](#), for each token;
 - `account.activation_token` as bearer authentication
 - With JSON body:
 - `latitude, longitude`: course-grained location from smartphone (GPS; address dialog; rounding to hexagon)
 - `tz_name`: timezone database name of the smartphone timezone
 - (later, in v3?) `building and building-specific parameters` (`yr_built, type, floor_area, heat_loss_area, energy_label, energy_index`, to be discussed based on privacy concerns)
4. API verifies `account.activation_token`:
 - *account must not have been activated yet*,
 - token must not be expired
5. API generates a random, long-lived, secret `account.authorization_token`
 - identifies and authorizes the account in calls from the app
6. API updates the associated entry in the `account` table in the database
 - `account.activated_at` timestamp;
7. API returns `account.authorization_token` in result of POST to the app
 - Stored persistently; used as bearer authentication token in API calls from app





token	activation	session
account		
device		



-  Wi-Fi:SSID+pwd;
server:tst/prod
-  installation manual

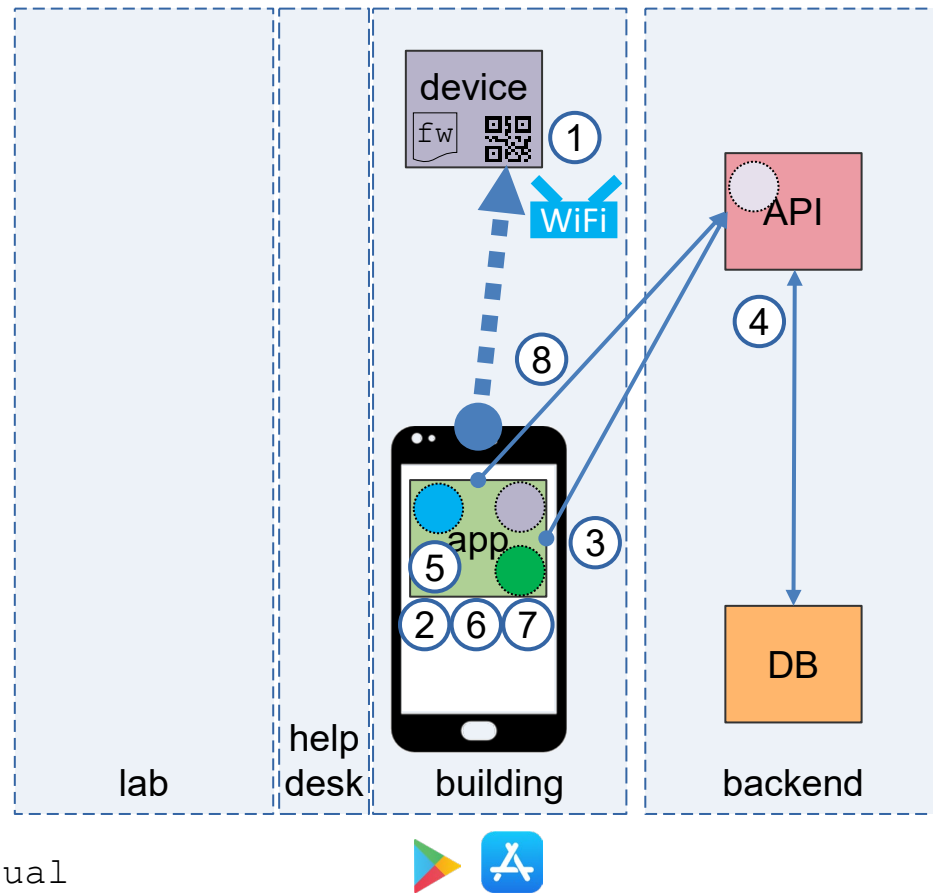


(D) App device activation

- (device with e-ink) Resident presses button to display provisioning QR-code
- App scans QR and reads values from the following keys in the QR payload:
 - name: the service name (BLE service or SSID) used for provisioning; in NeedForHeat we use `CONCAT(HASH(<device_type.name>;n);RIGHT(<MAC>;m))`, n & m: avoid collisions
 - pop: a secret, random value (generated dynamically on a device with e-ink screen)
 - transport: type of transport (softap or ble) used during provisioning
 - security: whether transport is secured (0 of 1) during provisioning;
 - password: wifi password (only when "transport": "softap", "security": "1")
- App calls POST on `/device` (and not `/account/device/activate`) using parameters
 - name: name value of the QR-code
 - building_id
 - activation_secret = pop value from the QR code.
- API returns the results of POST on `/device` to the app:
 - Server first retrieves `device_type.name`, by looking up `device_type` for which: `LEFT(name; n) = HASH(<device_type>;n)`
 - `Device_type.installation_manual_url`, for installation instructions for this `device_type`
 - If this device already exists and the account is not the original account, then an error is returned: prevent malicious residents to couple another resident's device.
- App retrieves `CONCAT(Device_type.installation_manual_url;<ISO language code>;'')` and shows (language-specific) installation instructions to resident
 - Resident installs and powers up device according to the installation manual
- Resident performs various actions & finally confirms installation & power up in app.
 - selects Wi-Fi SSID & enters password (only needed for first device in building);
 - (if app has both test and production session tokens) selects test/production server
- App send Wi-Fi SSID+password+`UseProductionServer` to device using Unified Provisioning
- App repetitively calls GET on `/device/{device_name}`
 - Until first measurement is uploaded (app signals this with green check)
 - Or until a timeout occurs (app signals this with a red cross)





token	activation	session
account		
device		

-  Wi-Fi:SSID+pwd;
server:tst/prod
-  installation manual



(E) Backend device activation

1. Device detects it is online, and not yet activated and calls [POST on device/activate](#); on the selected server (test/production):
 - `device.activation_secret` as bearer authentication
 - `name` as parameter
2. API generates a random, long-lived, secret `device.authorization_token`
3. API returns result of the [POST on device/activate](#)
 - `device.session_token`; stored persistently; used as `DeviceSessionTokenBearer` in future PI calls by the device
 - `info_url`: URL if not empty, then this should URL should be displayed as a QR-code on the e-ink screen after provisioning.
4. Device starts uploading measurements, via:
 - [POST on /upload](#)
 - heartbeat measurements are mandatory for devices; first heartbeat must be uploaded asap after `device/activate`;
 - units should be encoded in property names, using the '[PhysiQuant Unit naming convention](#)'..
5. (See also last point on previous slide)
After receiving the first upload (typically: the first heartbeat), the app signals to resident that the installation process for this device completed successfully.

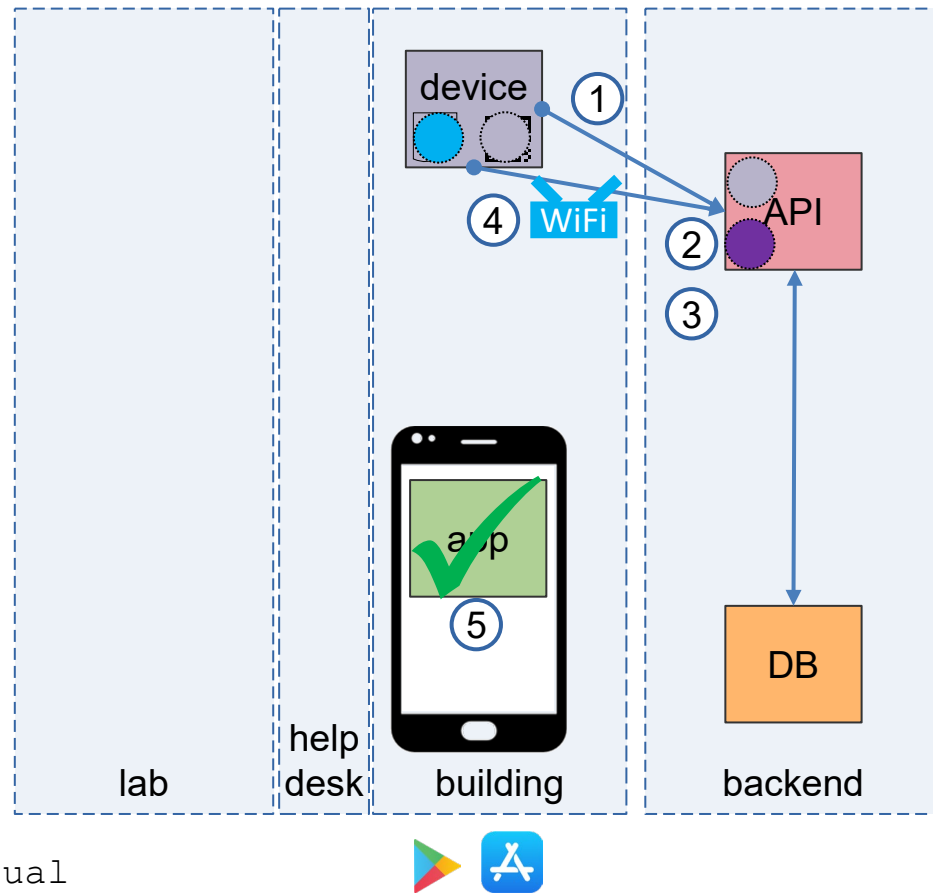
token	activation	session
account		
device		



Wi-Fi:SSID+pwd;
server:tst/prod



installation manual



Migration from API v1 to v2

- **API: change POST on /account endpoint**
 - remove location and tz_name parameters;
 - add campaign_name as parameter
 - retrieve components for firebase URL from campaign table;
 - use the proper account token key name in the provisioning_url returned, based on whether the server called is a test server or production server
- **API change post on /account/activate endpoint**
 - add location and tz_name parameters (currently: optional)
 - (later, v3.0) add optional building parameters (yr_built, type, floor_area, heat_loss_area, energy_label, energy_index, to be discussed based on privacy concerns)
- **API: change POST on /device endpoint**
 - allow any authorized account to POST on this endpoint
 - if this device already exists and the account is not the original account, then return an error: prevent malicious residents to couple another resident's device.
 - Include effect of /account/device/activate endpoint: API links the specific device to (the building of) the account; the link between device and account can be undone only manually (e.g. for device re-deploy later to building of another account);
 - return installation_manual_url in the result.
- **API: deprecate /account/device/activate endpoint**
- **API: deprecate /device_type/{device_name} endpoint**
- **API: change POST on /device/activate endpoint**
 - return URL incl. {device_name}; used as QR-code payload on devices with e-ink. TO DO: server expands URL to combine info from device.info_url and campaign.info_url
- **API: change GET on /device/{device_name} endpoint**
 - server should return more info in this call: for each property, the last time data was received and the value (this would enable longer term health monitoring by the app)
- **API: change POST on /device/measurements/variable-interval, /device/measurements/fixed-interval endpoints**
 - accept & store all valid properties in an upload (i.e. do not reject ALL properties if only one property is invalid) from a valid device (i.e. one installed by an app with a valid account)
- **MariaDB: add campaign table; with columns**
 - name, info_url, provisioning_url
- **MariaDB: remove unit from property table**
 - units should be encoded in property names using the ['PhysiQuant_Unit' naming convention](#).
- **MariaDB: remove device_type_property table and csv_create_update() function**
 - Allow all device types and all properties. This does not add to security
- **MariaDB: remove display_name from device_type table**
 - display_name is language dependent; should be included in content pointed to by installation_manual_url.
- **App: use all activation tokens present in the firebase URL**
 - A valid firebase URL should contain at least one account.activation_token, for production server and/or for production server. The app should call /account/activate on each token it finds; using LAB_DOMAIN_PROD and LAB_DOMAIN_TEST (these are compiled constants) to construct the endpoint URL.
- **App: send test/production to device**
 - If an app has both a session token for the test server of the lab and a session token for the production server (which is a special case needed only in some lab scenarios), then for each device the user should be given a choice: do you want to link this device to the test server, or the production server? The Espressif Unified Provisioning protocol is extensible and should be extensible to send this additional info (which is essentially one bit of info, i.e. a boolean: UseProductionServer?).
- **App: remember activated devices (for display & recoupling)**
 - When QR of known device (already provisioned earlier by this account) is scanned; do not call POST on /device; instead, just allow for recoupling: ask user whether to retry existing WiFi SSID & password, or whether to scan WiFi again & ask password.
- **App: do not use deprecated /account/device/activate endpoint**
- **App: do not use deprecated /device_type/{device_name} endpoint**
- **App: just show content of installation manual URL; don't use display_name**
- **Generic device firmware:**
 - change device.name: use CONCAT (HASH (<device_type.name>;n); RIGHT (<MAC>;m));
 - determine proper values for n & m lengths; suitable to avoid collisions
- **Generic device firmware**
 - Use URL returned by POST on /device/activate as a basis for the QR-code to display (if any)
- **Generic device firmware**
 - Include device_type.display_name in installation manual content (should be in server HTML)
- **Generic encode units in property name**
 - Recommended: encode units in property names, add as postfix after a double underscore.