REST access: API keys and access rights

External clients (kiosk terminals, mobile apps, integrations) talk to Webdesk through its REST interface (path /rest/... below the Webdesk context path). This page describes how such a client authenticates, how an API key is created and managed, and how **access rights** and **access right sets** define which REST calls it may make. It is intended for system administrators and consultants. The access rights for REST endpoints are independent of the action permissions of the authorization control[1] (see permission types[2]): those control who may open an action in the user interface, whereas the access rights described here control which REST URLs may be called with which HTTP method.

Two ways to authenticate

Aspect	Login with a user (session)	API key
Typical use	Integrations and apps that act on behalf of a Webdesk user	Devices and systems without a user login, e.g. kiosk terminals
Flow	POST /rest/auth/login with a JSON body containing username, password and optionally rememberLogin. Response 200 with session data (including authentication-state, session-timeout, login-token), or 401 if the login fails. Subsequent calls must send the session cookie.	Every call carries the HTTP header X-Auth-Token: <your-api-key>. No prior login is needed.
Authorization	Access rights of the user's **roles** (role tab "Access Rights")	Access rights of the **API key** (directly on the key and via assigned access right sets)

If the X-Auth-Token header is set, the call is handled as an API key call, otherwise as a session call. (Only in running mode dev or test is Basic Authentication via the Authorization header accepted as well; it is intended for tests.)

The /rest/auth/login call and a few other public endpoints (e.g. password reset, system time) are not checked by the REST authentication. For browser logins the normal way remains the Webdesk login page; /rest/auth/login is meant for external clients.

Session calls: authorization via roles

External clients that log in with a user need role-based access rights. All access rights and access right sets of all roles held by the logged-in user count. If neither a granting nor a denying right exists for the URL and HTTP method, the call is rejected with 403. A few endpoints intended for every logged-in user (e.g. under /rest/po/systeminfo/, /rest/po/users/, /rest/po/actionpermissions/) are exempt from the rights check; being logged in is sufficient for them. When the user interface itself calls REST endpoints (detected through the Referer), the access rights stored in the respective action apply instead (see below).

API key with user identification (impersonation)

An API key can additionally act on behalf of a Webdesk user, e.g. when a kiosk terminal fetches the data of a person after scanning an NFC card. For this, the client sends **one** of the two headers:

Header	Description
User-Identification-NfcCard	ID of the NFC card; the user is the person to whom the card is assigned.

User-Identification-Useruid	UID of the person.

Restrictions:

- The key must first be authorized for the called endpoint (same check as without the header).
- Both headers must not be sent at the same time and the value must not be empty (response 401).
- If no person can be found for the card or UID, the response is 401.
- Impersonation is only allowed for **kiosk keys**, i.e. keys assigned to a kiosk, and only for persons of **the same client** as the kiosk. For a key without a kiosk or a person of another client the response is 403.

The access rights of the key remain decisive even with impersonation; the rights of the identified person do not extend them.

Managing API keys

The screens are located in the admin menu under "System" (in the folder "Security"): **"API Keys"** (list) and **"Accessright Sets"** (access right sets). Depending on the menu layout of the system, the entry "Accessright Sets" may also appear directly under "System".

List "API Keys"

The list shows all keys with the columns "API Key" (the key value itself), "API Consumer", "Valid from" and "Valid till"; keys that have been invalidated remain visible. The icon in the first column opens a key, the button **"Generate New API Key"** creates a new one.

Creating a key

1. In the "API Keys" list, click **"Generate New API Key"**.
2. Enter the name of the client in the **"API Consumer"** field (description only, e.g. "Terminal Main Entrance").
3. In the section "Api Key Access Right Sets" assign one or more access right sets using "Add", and/or enter individual rights in the section "Api Key Specific Access Rights" (see below).
4. Save. "API Key saved successfully" is shown; the field "API Key" now contains the generated key value, which has to be entered in the client.

Warning: A new key without an access right set and without its own rights is valid, but may not call a single endpoint (response 403).

Warning: The key value is the secret with which the client identifies itself, and it is displayed in clear text in the list and in the screen. Access to these screens should be restricted accordingly; keys do not belong in documentation, tickets or URLs. For images (/image/...) Webdesk additionally accepts the parameter apiKey, but the header X-Auth-Token is preferable.

Editing, invalidating, deleting a key

- **"API Consumer"** and the access rights can be changed at any time. "Valid from" and "Valid till" are display-only in the screen. When saving, the sets and rights of the key are replaced by the content of the screen. Already saved right rows are read-only and can only be deleted; a changed right is implemented as a deleted row and a newly entered row. Changes to the rights apply to the following calls; the client does not need to log in again.
- **"Invalidate"** (confirmation: "Are you sure you want to invalidate this API-key?"): the key is terminated (Valid till = previous day) and immediately returns 401. The entry stays in the list; the button disappears. An invalidated key cannot be made valid again in the user interface.
- **"Delete"** (confirmation: "Are you sure you want to delete this API-key?", button "Delete API-Key"): the key is removed permanently.

If the (valid) key belongs to a kiosk, the confirmation points out that this kiosk will no longer work afterwards. Invalidating or deleting breaks the link between kiosk and key; a new key is generated in the kiosk screen (see below).

Keys of kiosks

For kiosks (menu "Cards and Kiosks" > "Kiosks") Webdesk manages the key itself:

- When a kiosk is created or registered, a key is generated automatically (API Consumer = kiosk name) and the **default access right set** of the kiosk type is assigned to it (see below).
- In the kiosk screen, the field "API key" with the button "Show API Key" displays the key. The button is only available to users who may view all clients (system administration).
- If the kiosk has no valid key, the button **"Generate API Key"** appears. It generates a new key with the default set and removes the old key.
- If a kiosk is deleted, its key is invalidated or deleted along with it.
- If an existing PTM kiosk is additionally registered as a TA kiosk (or vice versa), it becomes type PTM_TA and also receives a new key.

Access rights and access right sets

Structure of an access right

Property	Description	Possible value(s)
"Selector"	Path of the REST endpoint, without context path and without query parameters, starting with / rest/..., e.g. /rest/po/kiosks. Wildcards: * = any string (also across /), ? = exactly one arbitrary character. The selector must match the entire path.	Text, required
"Privilege"	Permitted kind of access. For the mapping to the HTTP method see below.	READ, CREATE, UPDATE, DELETE, FULL
"Deny"	If checked, the right forbids the access instead of allowing it.	Yes/No (default: No, the right allows)

HTTP method	required privilege
GET	READ
POST	CREATE
PUT, PATCH	UPDATE
DELETE	DELETE

FULL corresponds to READ, CREATE, UPDATE and DELETE together. Other HTTP methods (e.g. HEAD) cannot be mapped and result in an error response 401.

Where access rights are assigned

Place	Effect
API key (sections "Api Key Access Right Sets" and "Api Key Specific Access Rights")	For calls with X-Auth-Token. The sum of the key's own rights and the rights of all assigned sets applies.
Role (tab "Access Rights" in the role screen, sections "Role Access Right Sets" and "Role Specific Access Rights"; see Roles[3])	For session calls of all role holders. The tab is only visible to system administrators and not for derived client roles.
Action (file act-descr.xml)	For REST calls that the user interface makes from this action. Cannot be changed in the user interface: the action only displays them in the tab "Access Rights", changes made there are not saved.
Access right set	Reusable collection of rights that can be assigned to any number of keys and roles.

In act-descr.xml rights are declared as <accessrights><accessright selector="/rest/po/audit/label/*" privilege="GET"/></accessrights>. The attribute privilege takes one or more HTTP methods separated by commas, e.g. GET,POST. Such rights always allow; denying is not possible.

Maintaining access right sets

1. Open the admin menu "System" > **"Accessright Sets"** (list "Accessright Sets" with the column "Name").
2. Click **"New Access Right Set"** or open an existing set.
3. Enter a unique **"Name"** (required; the name may only exist once).
4. Using "Add", enter rows with "Selector", "Privilege" and, if needed, "Deny".
5. Save. Existing rows are read-only and can only be deleted; to change one, delete the row and enter it again. The set can then be assigned in the API key or role (selection field "Access Right Set"; a jump icon opens the set).

Warning: Changes to a set affect all keys and roles to which it is assigned.

Default sets for kiosks

For kiosk keys, Webdesk creates a set named Default <type> Kiosk Access Right Set on first demand (type = PTM, TA or PTM_TA) and assigns it to the new key. The sets contain (all granting):

Selector	Privilege	in set
/rest/po/kiosks*	FULL	PTM, TA
/rest/po/nfctags*	FULL	PTM, TA
/rest/ptm/*	FULL	PTM, TA
/image/*	READ	PTM, TA
/rest/ta/*	FULL	TA only

"Default PTM_TA Kiosk Access Right Set" combines the rights of both sets. A set that has been created once is found again by its name; later changes to the default configuration do not change an existing set. However, a set can be edited in "Accessright Sets", which changes the rights of all kiosks of that type.

How the authorization of a call is decided

For an API key call, the rights of the key and of all its sets are merged and checked against the path and HTTP method of the call:

1. The path of the call (without context path, e.g. /rest/po/kiosks/4711) is compared with the selectors. Method and required privilege result from the table above.
2. A selector **without wildcards** that equals the path exactly and contains a right for the required privilege decides immediately (allowed or denied).
3. Otherwise, among the matching selectors **with wildcards** the most specific one decides, i.e. the one with the most characters that are not wildcards. Selectors without a right for the required privilege are not considered. For selectors of equal specificity the order is not defined; such overlaps should be avoided.
4. Within the same selector, **"Deny"** always wins over "allow", regardless of the order.
5. FULL acts like READ, CREATE, UPDATE and DELETE. If a selector allows UPDATE or CREATE, READ is also allowed for that selector (unless READ is denied there).
6. If no right applies, the call is **not authorized** (403). There is no default access.

For session calls, the same rules are applied to the rights of the user's roles. Here too, no matching right means 403.

Example: a key for a terminal that may only read

An external terminal is to retrieve person data but change nothing. (The paths are examples; the Swagger UI shows the actual paths of an endpoint.)

1. "System" > "Accessright Sets" > "New Access Right Set": "Name" = Terminal read only.
2. Add a row: "Selector" = /rest/po/persons/*, "Privilege" = READ, "Deny" not checked. Save.
3. "System" > "API Keys" > "Generate New API Key": "API Consumer" = Terminal Main Entrance; add the set "Terminal read only" in the section "Api Key Access Right Sets"; save.
4. Enter the displayed key in the terminal as the header X-Auth-Token.

Result: GET /rest/po/persons/... is answered. POST, PUT or DELETE on the same path, and every call to another path, return 403. If access is to be withdrawn later, it is enough to change the set or to block the key with "Invalidate" (401).

Note: The terminal key in this example is not assigned to a kiosk and therefore cannot use user identification (User-Identification-...). Keys for kiosks are created through the kiosk administration.

Troubleshooting: 401 and 403

Response	Possible causes and what to check
401 (not authenticated)	• The header X-Auth-Token is missing or empty and there is no session. • The key is unknown (typo, deleted) or no longer valid: check "Valid from"/"Valid till" in "API Keys". • Impersonation: both User-Identification-... headers at the same time, an empty value, or no person found for the card or UID. • Unsupported HTTP method (e.g. HEAD) or an unexpected error; the server log then contains "Problem during authentication/authorization.". • Session call: session expired or never logged in (/rest/auth/login returns 401 for a failed login).
403 (not authorized)	• The key is valid, but no right matches path and method, or a "Deny" right applies. Check the sets and rights in the API key: is the selector

	correct (starts with /rest/, without context path)? Does the privilege match the method (GET = READ, POST = CREATE)? Does the path end with a slash that the selector does not cover? • Impersonation with a key that is not assigned to a kiosk, or with a person of another client. • Session call: the user's roles have no matching right (role screen, tab "Access Rights").

In addition, the service logs a line of the form "Check if api key ... has permission to make a GET call over /rest/..." at debug level for every check. It names the method and path of the call (logger at.workflow.webdesk.po.impl.PoAccessRightServiceImpl).

Browser clients: CORS

For calls from browsers originating from a different address, Webdesk sets the CORS headers. The allowed origins are defined in the system parameter allowedOriginsForCORS of the bean SecurityHeadersOptions (screen "Systemparameter"). A request with a matching Origin receives Access-Control-Allow-Origin with that value and Access-Control-Allow-Credentials: true. Entries may contain *, e.g. https://app.example.com or https://*.example.com. Allowed request headers include X-Auth-Token, User-Identification-NfcCard and User-Identification-Useruid. Native apps and terminals do not send an Origin header and are not affected.

Warning: In the delivered default the list is empty. In that case every Origin that is sent is returned unchanged, i.e. every origin is allowed. For systems reachable from the internet, the allowed origins should be entered.

Security notes

- **CSRF** is currently not implemented in Webdesk. The protection through the session-bound continuation id of forms does not apply to REST calls and login_process. Keep this in mind when assessing security topics and when adding new POST endpoints.
- Short-lived login tokens (LoginTokenUtils, e.g. for server-side report downloads with user switching) and temporary access via security-key reside in the cache of the respective node. In a cluster, calls that land on another node can therefore fail unless requests are pinned to one node.
- Session and cookie settings (e.g. SameSite, headers, timeouts) depend on the environment and must be verified on site.

Swagger UI and Postman collection

- **Swagger UI:** The menu entry **"Swagger Rest-API Explorer"** (admin menu "System") opens rest/swagger-ui/index.html in a new window, showing the REST endpoints of Webdesk with their paths, methods and parameters. Calls from the Swagger UI run with the session of the logged-in user. They are permitted for system administrators and for users who have access rights for the endpoint through any of their actions. The Swagger UI is useful for looking up paths for selectors.
- **Postman collection:** The repository contains postman/Webdesk - Public API.postman_collection.json. It contains the call "Login" (POST {{restURL}}/auth/login) and sample calls for time and attendance (absence reasons, own bookings, current state, performing a booking). The variable restURL is not preset in the collection and has to be set to the REST base address of the system (Webdesk address plus /rest). The collection uses the session login, not API keys.

1. /daisy/personalwolke-admin/5479-dsy.html?language=4
2. /daisy/personalwolke-admin/2650-dsy.html?language=4
3. /daisy/personalwolke-admin/3320-dsy.html?language=4