Humanet
powered by HOUR

HumanetAPI Integrator Guide

Standard endpoints, access and usage rules

2. 11. 2026

Contents

1 About HumanetAPI 4
2 Finding the server and the database hash 5
2.1 Look up the HumanetAPI address 5
2.2 Where to find the database hash 6
3 Authentication 7
3.1 API token for an integration 7
3.2 Short-lived token for a named user 7
3.3 Basic authentication 8
3.4 Checking the setup 8
3.5 Listing the databases of a login 8
4 Permissions 9
4.1 Default rights 9
4.2 Adding a right 9
4.3 What a technical user cannot do 9
4.4 Further controls in the database 10
5 Data conventions 11
5.1 Format 11
5.2 Dates 11
5.3 Parameters 11
5.4 Identifiers, validity and history 11
5.5 Language 11
5.6 Writes 11
6 Request limits and good practice 12
6.1 Request limit 12
6.2 Other limits 12
6.3 Good practice 12
7 Errors 13
8 Endpoint reference 14
8.1 Access and system (Base, Master) 14
8.2 Companies and workplaces (BaseErp) 15
8.3 Organisational structure and code lists (BaseOrs, PamTools) 15
8.4 Change log of the organisational structure (ChangeLog) 17
8.5 Persons and employment relations (Pam) 18
8.6 Attendance and absences (Attendance) 21
8.7 Attendance terminals 25
8.8 Invoicing (bookkeeping databases) 25
8.9 Bookkeeping, address book and dealers (bookkeeping databases) 26
9 Typical integrations 28
9.1 Employee master data for another system 28
9.2 Creating employees from a recruiting or HR system 28
9.3 Keeping the organisational structure in sync 28
9.4 Attendance and absences for payroll, BI or planning 28
9.5 Card and access systems 29
9.6 Invoices from an e-shop or another sales system (bookkeeping database) 29

10 Go-live checklist and support 30
10.1 Before going live 30
10.2 Support 30
A All endpoints at a glance 31
B Frequently used code lists 34

1 About HumanetAPI

HumanetAPI is the REST interface of Humanet, the HR, payroll, attendance and bookkeeping system of HOUR, spol. s r.o. It lets your application read data from a customer's Humanet database and write selected records back: employees and their employment relations, the organisational structure, attendance, absences, invoices and bookkeeping documents.

- Every endpoint lives under `https://<server>/HumanetApi/api/...`. The server depends on the database (chapter 2).
- Data is exchanged as JSON in UTF-8. Send `Accept: application/json` and, with a body, `Content-Type: application/json`.
- HTTP methods keep their usual meaning: `GET` reads, `POST` creates, `PUT` updates, `DELETE` deletes or deactivates.
- Each request carries the credentials in the `Authorization` header (chapter 3). The credentials pick the database, so one token always reaches exactly one customer database.

Interactive documentation (Swagger)

Every server publishes the same list of standard endpoints with request and response schemas and a "Try it out" console:

```
https://klient.humanet.sk/HumanetApi/swagger/ui/index
https://klient.humanet.sk/HumanetApi/swagger/docs/v1     (OpenAPI 2.0 JSON, for client
generators)
```

Use the Swagger of the server that hosts your database, because "Try it out" sends the request to that server. Swagger is the authority for field names and types. This guide explains what each endpoint is for, how the endpoints fit together and what to watch out for.

This guide covers the **standard endpoints**, which every customer can use: the 105 operations published in Swagger today. Some customers also have endpoints built for them; HOUR's customer centre or the customer's project manager hands their documentation directly to that customer.

2 Finding the server and the database hash

Humanet runs on several application servers. Each customer database lives on exactly one of them, and HOUR may move a database to another server. A request sent to the wrong server fails with HTTP 440 *Database not found*. Do not hard-code the server: ask the Humanet portal for it.

2.1 Look up the HumanetAPI address

```
GET https://api.humanet.sk/api/HWsApi/Get?hash=1-<database-hash>&format=json
```

The call needs no authentication. The `hash` parameter is the prefix `1-` followed by the database hash (2.2). Always send `format=json`; without it the request is rejected. Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "Status": "OK",
  "WebServiceApiUrl": "https://klient.humanet.sk/HumanetApi/api",
  "DbHash": "00000000-0000-0000-0000-000000000000"
}
```

Field	Meaning
`Status`	`OK`, or `ERROR` when the `hash` parameter is malformed (then also HTTP 400).
`WebServiceApiUrl`	Base address of the endpoints, ending in `/api`. Append the endpoint path without the `api/` prefix, e.g. `WebServiceApiUrl + "/Pam/Persons"`.
`DbHash`	The database hash without the `1-` prefix, as sent.

Typical values (bookkeeping databases can be on another server; use the value as it comes, because servers can be added):

- `https://klient.humanet.sk/HumanetApi/api`
- `https://klient3.humanet.sk/HumanetApi/api`
- `https://klient4.humanet.sk/HumanetApi/api`

> **The lookup does not confirm that the database exists.** Check the hash once with `GET api/Base/Token/Check` (chapter 3) when you set up the integration.

Recommended handling

- Look the address up when the integration starts (or once a day) and keep it in memory. Do not call the lookup before every request.
- When HumanetAPI answers 440, look the address up again and retry once. That is what happens after HOUR moves a database to another server.
- Keep the database hash in your configuration, not the server address.

On-premise installations. Customers who run Humanet in their own network have their own HumanetAPI address (for example `https://humanet.customer.example/HumanetApi`). The portal lookup does not apply to them; ask the customer's IT for the address.

2.2 Where to find the database hash

The database hash is the identifier of a customer database in Humanet, a GUID such as `00000000-0000-0000-0000-000000000000`. Handle it as internal information of the customer. There are four ways to get it:

1. **In Humanet**: ***Pomocník / Systém***, page *Systémové informácie*, field *HASH dátového prostredia*. If the field shows two values (`Master: … DB: …`), use the *Master* value. Do not confuse it with *Id dátového prostredia pre webovú službu*, which is a different identifier.
2. **In the Humanet portal** (`moj.humanet.sk`): the owner of the database opens the list of agendas, chooses *Agenda information* in the agenda's menu and copies the *Hash* field. The field is shown only to the agenda's owner.
3. **From HumanetAPI** with a Humanet login and password: `GET api/Master/UserDatabases` (chapter 3.5) returns every database the login can open, with its `dbHash`.
4. **From HOUR**: the customer centre gives the hash to the customer's administrator on request.

Calls with an API token generated in Humanet do not send the hash. You still need it for the server lookup.

3 Authentication

Every request carries credentials in the `Authorization` header. For integrations, use an **API token generated in Humanet**. The scheme name is not case-sensitive.

```
GET https://klient.humanet.sk/HumanetApi/api/Pam/Persons
Authorization: ApiToken 0123456789ABCDEF…
Accept: application/json
Culture: en-GB
```

3.1 API token for an integration

The customer's administrator generates the token in Humanet:

1. Open ***Pomocník / Systém***. The page *Systémové informácie* opens.
2. At the bottom of the page, in the list *Zoznam API tokenov*, add a new token. Enter a name (for example the name of the integration) and, optionally, *Platné do*, the last day of validity. Leave it empty for a token without expiry.
3. Humanet creates a technical user for the token, named `api01@humanet.sk`, `api02@humanet.sk` and so on, and puts it into the access group *API prístupová skupina* (chapter 4).
4. The token is shown **only once**. Copy it with *Skopírovať do schránky* and hand it to the integrator through a secure channel.

The list is visible to administrators and to users with the right *Správa systému → Užívatelia*.

Action in the list	Effect
Edit the token	Generates a new token. The previous one stops working.
Delete the token	Deactivates the token and its technical user.
Platné do passed	The token stops working at midnight after that day: 403 "API token is expired."

The token contains encrypted information about the database it was issued for and is valid only for that database, so you send nothing else to choose the database. Keep it like a password: in a secret store, never in a URL, a log or source code. Use one token per integration, so that you can revoke one without affecting the others.

3.2 Short-lived token for a named user

`POST api/Base/Token/Generate` signs in an ordinary Humanet user with a name and password and returns a token. With `ValidForSeconds` (60–86400) the token expires at `expiresAt` (UTC):

```
POST https://klient.humanet.sk/HumanetApi/api/Base/Token/Generate
Content-Type: application/json

{ "dbHash": "00000000-0000-0000-0000-000000000000",
  "UserName": "jan.novak@example.com",
  "Password": "…",
  "ValidForSeconds": 900 }

200 OK
{ "apiToken": "0123456789ABCDEF…", "expiresAt": "2026-11-02T08:15:00Z" }
```



The token then has that user's rights. Use this for applications in which people act as themselves; for system-to-system integrations use the token from 3.1. For a named user always send `ValidForSeconds`, check `expiresAt` in the response and stop using the token when it expires.

3.3 Basic authentication

Instead of a token you can send the user name and password with every request:

```
Authorization: Basic base64("{db-hash}:{user-name}:{password}")
```

Example for the database `00000000-0000-0000-0000-000000000000`, the user `jan.novak@example.com` and the password `Heslo123` (made-up values):

```
Authorization: Basic
MDAwMDAwMDAtMDAwMC0wMDAwLTAwMDAtMDAwMDAwMDAwMDAwOmphbi5ub3Zha0BleGFtcGxlLmNvbTpIZXNsbzEyMw==
```

The string is encoded in UTF-8. Prefer a token: a password in every request is harder to protect and to rotate.

3.4 Checking the setup

```
GET https://klient.humanet.sk/HumanetApi/api/Base/Token/Check
Authorization: ApiToken 0123456789ABCDEF…

200 OK
"OK"
```

A 403 means an invalid or expired token, 440 an unknown database (usually the wrong server, chapter 2), 402 a database whose access is disabled.

3.5 Listing the databases of a login

`GET api/Master/UserDatabases` uses a personal login without a database hash, `Authorization: Basic base64("{user-name}:{password}")`, and returns `[{"name", "type", "dbHash"}]` for every database the login can open. It is meant for applications that let a person pick a database. A technical API user cannot call it.

4 Permissions

HumanetAPI uses the same rights as the Humanet user interface. Each endpoint requires a right (listed in chapter 8 and appendix A). A token from 3.1 has the rights of the access group of its technical user. Humanet puts every new technical user into the access group *API prístupová skupina* (code `API`).

4.1 Default rights

When Humanet creates the group, it grants:

Database	Rights	Endpoints usable at once
HR and payroll	*Základné nastavenia* → *Číselníky, Organizácie*; *Personálne údaje* → *Osoby, Roly*; communication with Humanet attendance terminals	persons, roles, duties, addresses, relatives, leave balance, `PersonExtendedImport`, organisations and workplaces, code lists, terminals
Bookkeeping (ERP)	*Adresár partnerov*, *Organizačná štruktúra* → *Dealeri*, *Denník*, *Výstupy účtovníctva*, *Fakturácia a odbyt*	all bookkeeping, invoicing and address-book endpoints

Everything else, in particular the organisational code lists, contacts, payroll parameters and all attendance endpoints, needs a right the administrator adds to the group. Endpoints marked `DEFAULT` in chapter 8 work without it.

Humanet attendance terminals use their own access group, *APITERM prístupová skupina*. If no integration of the database needs the terminal communication right, the administrator can remove it from *API prístupová skupina*.

4.2 Adding a right

- The administrator adds the right to *API prístupová skupina* in the Humanet rights administration, by the name shown in chapter 8 (for example *Pomocné dáta PAM* → *Stredisk*á).
- A right granted for reading allows `GET`. `POST`, `PUT` and `DELETE` need the right for reading and writing.
- A right on a higher level of the tree covers the levels below it only when it is granted for the whole branch.
- Changes apply to the next request; nothing has to be regenerated.
- A missing right answers 403 with the right's name and code, for example `Stredisk á (MODULE.{TOOLS_PAM}.{DEPARTMENTS}): Nedostatočné prístupové práva!`.

> Grant only what the integration needs. As long as all technical users stay in *API prístupová skupina*, a right added for one integration is available to all of them. If integrations need different rights, the administrator can create another access group and move the technical user of an integration (`api01@humanet.sk`, …) into it, in the same way as for any other user.

4.3 What a technical user cannot do

A technical API user is not linked to any employee. Endpoints that work with "my" data (my absences, my team, my payslip) are part of the Humanet mobile application and are not in the standard list. Among the standard endpoints, `api/Attendance/Transactions` limits the result to the roles the user may see: give the group the right *Zobrazenie rolí v podmodule Roly*.

Some endpoints also apply the record filters of the access group (for example a restriction to some organisations). If the API returns fewer records than you expect, check the group's filters.

4.4 Further controls in the database

- **Transfer types.** Each endpoint is a data-transfer type in Humanet. The administrator can disable it; a write then answers 403 "Transfer is not allowed". Calls are recorded in the database's transfer log.
- **Module availability.** An endpoint of a module the database does not have answers 501 "External class … is not defined in Humanet." (for example invoicing in an HR database).

5 Data conventions

5.1 Format

- JSON in UTF-8. Field names are exactly as in Swagger, mostly camelCase (`personalNumber`); a few older contracts use PascalCase (`Status` , `ServerTime`). Match names case-sensitively when you parse.
- Empty values are sent as `null` . Enumerations are numbers.

5.2 Dates

- In JSON, dates are ISO 8601 without a time zone, for example `"2026-03-01T00:00:00"` . They are local dates in Slovakia; for date fields use only the date part.
- In query parameters send `yyyy-MM-dd` , for example `input.from=2026-10-01` . Most periods include both end days.
- `expiresAt` of a token is the only UTC value (it ends with `Z`).

5.3 Parameters

- Swagger shows many query parameters with the prefix `input.` (`input.from`). The prefix can be left out (`from`), but do not mix both forms in one request.
- A list parameter is repeated: `?input.companies=001&input.companies=002` .
- Always send every parameter that Swagger marks as required.

5.4 Identifiers, validity and history

- Records are linked by numeric ids (`id` , `personId` , `roleId` , …). They are stable and never reused. Codes and numbers (`roleNum` , `personalNumber` , `code`) are what people see and can change.
- HR data has history. A read returns the version valid today, or at the end of the current payroll period for payroll-related data. Writes take effect from the start of the current payroll period unless the endpoint takes a `validFrom` .
- Many lists include deactivated records. The reference says where an `isActive` flag or a validity tells them apart.

5.5 Language

Names of code-list values and messages from Humanet come in the language of the `Culture` header: `sk-SK` (Slovak, default) or `en-GB` (English). Without it, `Accept-Language` is used. Any other value means Slovak. Messages produced by the API layer itself are in English.

5.6 Writes

- Successful writes answer 201, PUT included; deactivations and deletions answer 200. Some writes return the saved record with its `id` , others an empty body (chapter 8).
- A `PUT` replaces the whole record. Read it with `GET` , change the fields and send it back complete.
- If a write fails, read the data again before you repeat it. A 502 on a write means the request may have been processed.
- `POST` is never idempotent: repeating it creates another record.

6 Request limits and good practice

6.1 Request limit

HumanetAPI limits the number of requests. Do not use additional tokens to increase throughput. Today the limit is **600 requests per 60 seconds**. HOUR can change it, so read the limit from the response headers instead of hard-coding it:

Header	Meaning
`X-Rate-Limit-Requests-Quota`	Requests allowed per window.
`X-Rate-Limit-Requests-Left`	Requests left in the current window. At 1, wait for the next window.
`X-Rate-Limit-Time-Window-Sec`	Length of the window in seconds.
`X-Rate-Limit-Time-Reset-Sec`	Seconds until the counter resets.

Above the limit the API answers `429` with `{"Code": 429, "Message": "Too Many Requests"}`. There is no `Retry-After` header: wait `X-Rate-Limit-Time-Reset-Sec` seconds and continue.

6.2 Other limits

- The size of a request body is limited. Send large imports in smaller batches.
- When the database is busy, a request can answer 503. Repeat it after a pause.
- Lists are not paged. Endpoints return the whole result in one response.

6.3 Good practice

- **Read lists, not records one by one.** One `GET api/Pam/PersonRoles` returns all roles; looping over employees with a filter costs hundreds of requests and the same data.
- Load code lists (org units, cost centres, positions, functions, code lists from `Base/Keys`) once per run and keep them in memory.
- Run full synchronisations once a day, outside working hours (for example between 19:00 and 6:00). During the month-end payroll closing, avoid heavy calls.
- Send requests one after another or with low parallelism (2–4). Many parallel requests do not finish sooner.
- For attendance, never loop over employees with `Attendance/Real`, `Plan` or `Real/Absences`: these recalculate attendance (8.6).
- Retry 429, 503 and network errors with a growing pause (for example 5 s, 30 s, 2 min). Do not retry 400, 401, 403 or 404 without changing the request.
- Set a client timeout of a few minutes for large lists.

7 Errors

Errors come back with a JSON body of this shape; read the body, not only the status line:

```
{ "Code": 403, "Message": "API token is not valid.", "Details": null }
```

When Humanet itself fails, `Message` holds an incident number and `Details` the description. Quote the incident number when you contact HOUR.

Status	Typical message	What to do
400	Invalid input, business rule violated (for example "The date is not in editable range")	Fix the request. Do not retry unchanged.
401	No `Authorization` header; "User verification failed"; "Password expired"	Check the credentials. For a token from Humanet, have the administrator regenerate it.
402	"Database access is disabled."	The customer's access to the database is disabled. Contact the customer.
403	"API token is not valid." / "API token is expired." / "…: Nedostatočné prístupové práva!" / "Transfer is not allowed" / "The database is currently being updated. Please try again later."	Token: generate a new one. Right: add it to the group (chapter 4). Update: retry in a few minutes.
404	Unknown path or record	Check the path (it is case-insensitive) and the id.
429	"Too Many Requests"	Wait for the next window (chapter 6).
440	"Database not found."	The database is not on this server. Repeat the lookup from chapter 2 and retry once.
500	Incident number in `Message`	An error in Humanet. Retry once later; if it persists, report the incident number to HOUR.
501	"External class … is not defined in Humanet."	The database does not have this module, or the endpoint is switched off for it.
502	"Connection to Humanet was lost while the request was processed."	For a write, check whether it was saved before you repeat it.
503	"Humanet is not available for the moment. Try again later."	Retry with a growing pause.

8 Endpoint reference

The endpoints are grouped by area. Each entry gives the path, what each HTTP method does, the parameters, the right the API access group needs (code and name in the rights tree) and what to watch out for. DEFAULT marks endpoints that work with the default rights of the group, WRITES marks reads that also change data. Field lists name the important fields; Swagger has the complete schemas.

8.1 Access and system (Base, Master)

Token handling, database discovery, code lists and server time. Chapter 3 explains authentication in detail.

---

api/Base/Token/Generate

POST Signs a Humanet user in and returns an API token: body `{"dbHash": "…", "UserName": "…", "Password": "…", "ValidForSeconds": 900}`, response `{"apiToken": "…", "expiresAt": "…Z"}`.

Right: Humanet user name and password in the body

- `ValidForSeconds` (60–86400) is optional. With it the token expires at `expiresAt` (UTC); without it the token does not expire and `expiresAt` is left out. Always send it for named users.

---

api/Base/Token/Check

GET Checks the credentials in the `Authorization` header and the connection to the database. Returns `"OK"`, otherwise the error from chapter 7.

Right: Any valid credentials

- Use it when you configure the integration and for occasional health checks, not before every request.

---

api/Master/UserDatabases

GET Lists the databases a login can open: `name`, `type`, `dbHash`.

Right: Portal / Humanet login (Basic user:password)

- Authenticates with `Authorization: Basic base64(user:password)` of a personal login, without a database hash. A system token does not work here.

---

api/Base/Keys

GET Returns a code list (číselník): every value with `code`, `weight`, `shortName` and `value` (long name), or one value when `valueCode` is given.

Parameters: `keyCode` (required, e.g. `S003008`, without the `input.` prefix), `valueCode` (optional). An unknown key or value returns an empty list.

Right: No right needed (valid token only)

- Names come in the language of the `Culture` header. Frequently used code lists are listed in appendix B.

---

api/Base/Sys/Time

GET Returns the server time: `{"Status": 0, "ServerTime": "…"}`. Used by attendance terminals to set their clock.

Parameters: `input.ver=1` (required). Another version returns `Status` 2.

Right: No right needed (valid token only)

- `ServerTime` is the local time of the server (Central European Time) without an offset.

`api/Base/IO/EDE/Transfer`

`POST` Uploads a data package (base64 content with source, target and title) to the Humanet data-exchange queue, where Humanet imports it later.

Right: Administrator (all-modules right)

- Used only in data-exchange scenarios agreed with HOUR, which also defines the package content.

8.2 Companies and workplaces (BaseErp)

A Humanet database can hold several organisations (companies, employers). Most HR data refers to them by `companyId`. These endpoints are read-only.

`api/BaseErp/Companies` `DEFAULT`

`GET` Lists the active organisations registered in the database: `id`, `code`, `name`, `shortcut`, `type`.

Right: `MODULE.{TOOLS_ERP}.{ORGANIZATIONS}` – *Základné nastavenia → Organizácie*

- An organisation that has several organisation types appears once per type. Type `001` is an organisation whose HR and payroll run in Humanet.

`api/BaseErp/Companies/Contacts` also **`api/BaseErp/Companies/Contacts/Emails`** also **`api/BaseErp/Companies/Contacts/Phones`** also **`api/BaseErp/Companies/Contacts/Others`** `DEFAULT`

`GET` Contacts of the type-001 organisations: `contactType` (key S001005), `value`, `companyId`. `/Contacts` returns all of them; `/Emails`, `/Phones` and `/Others` return subsets.

Right: `MODULE.{TOOLS_ERP}.{ORGANIZATIONS}` – *Základné nastavenia → Organizácie*

- `/Phones` = types 001–003, `/Others` = fax and web (004, 006), `/Emails` = the payslip e-mail (007). The general company e-mail (005) is only in `/Contacts`.

`api/BaseErp/Companies/Logos` also **`api/BaseErp/Companies/Stamps`** `DEFAULT`

`GET` Lists the logos and stamps stored for the organisations (`id`, `name`, `pictureType`, `companyId`). The image itself is not returned.

Right: `MODULE.{TOOLS_ERP}.{ORGANIZATIONS}` – *Základné nastavenia → Organizácie*

`api/BaseErp/Companies/Workplaces` `DEFAULT`

`GET` Places of work (`id`, `companyId`, `code`, `shortcut`, `name`). Roles refer to them by `workPlaceId`.

Parameters: `companyId` (optional): omit it for all workplaces, or pass an organisation id. `0` returns an empty list.

Right: `MODULE.{TOOLS_ERP}.{ORGANIZATIONS}` – *Základné nastavenia → Organizácie*

8.3 Organisational structure and code lists (BaseOrs, PamTools)

Roles point to these code lists by numeric id (`orgUnitId`, `departmentId`, `positionId`, `functionId`, `jobId`, `turnsImageId`). Load each list once per run and translate ids to codes and names in memory.

Humanet distinguishes the **organisational unit** (útvar, the place in the org tree) from the **cost centre** (stredisko). A role has both.

The lists include deactivated items. Only `JobPositions` and `Functions` return an `isActive` flag.

---

api/BaseOrs/OrgUnits

GET Lists the organisational units with `validFrom` / `validTo`.

POST Creates an organisational unit.

PUT Updates the unit with the given `id`. Every field is replaced, so send the complete record.

DELETE Deactivates the unit (`?id=`). The record stays and is still listed.

Right: `MODULE.{TOOLS_PAM}.{UNITS}` – *Pomocné dáta PAM → Útvary*

- Writes are valid from the start of the current payroll period.
- POST and PUT answer 201 with an empty body. Read the list again to get the new id.

---

api/BaseOrs/Departments

GET Lists the cost centres (`id` , `code` , `name` , …).

POST Creates a cost centre.

PUT Updates the cost centre with the given `id` (complete record).

DELETE Deactivates the cost centre (`?id=`).

Right: `MODULE.{TOOLS_PAM}.{DEPARTMENTS}` – *Pomocné dáta PAM → Strediská*

- Writes are valid from the start of the current payroll period.
- POST and PUT answer 201 with an empty body.

---

api/BaseOrs/JobPositions

GET Lists the job positions with `isActive`.

POST Creates a job position.

PUT Updates the job position with the given `id` (complete record).

DELETE Deactivates the job position (`?id=`).

Right: `MODULE.{TOOLS_PAM}.{POSITIONS}` – *Pomocné dáta PAM → Pracovné miesta*

- Writes are valid from the start of the current payroll period.
- POST and PUT answer 201 with an empty body. `validFrom` / `validTo` are part of the schema but are not filled.

---

api/BaseOrs/PayPlaces

GET Lists the pay places (`id` , `code` , `name` , `shortName` , `companyId`).

Right: `MODULE.{TOOLS_PAM}.{PAY_PLACES}` – *Pomocné dáta PAM → Výplatné miesta*

---

api/PamTools/Functions

GET Lists the functions with `isActive` and the validity of the current version.

POST Creates a function (`code` , `shortName` , `name` required). Returns 201 with the record and its `id`.

PUT Updates the function with the given `id`. Returns 200, or 404 for an unknown id.

DELETE Deactivates the function (`?id=`) and returns it with `isActive=false`.

Right: `MODULE.{TOOLS_PAM}.{FUNCTIONS}` – *Pomocné dáta PAM → Funkcie*

- Writes are valid from the start of the current payroll period.

api/PamTools/Orders

GET Lists the orders (zákazky): `id`, `code`, `shortName`, `name`, `dateFrom`, `dateTo`. A role's `jobId` points here.

Right: `MODULE.{TOOLS_PAM}.{ORDERS}` – *Pomocné dáta PAM → Zákazky*

- Ended orders are included; use `dateTo` to tell them apart.

api/PamTools/Turns

GET Lists the rotas (turnusy): length, weekly duty in hours, duty percentage.

Right: `MODULE.{TOOLS_PAM}.{TURNS}` – *Pomocné dáta PAM → Turnusy*

api/PamTools/TurnsImages

GET Lists the active rota patterns (obrazy turnusov) with their rota (`turnsId`) and calendar. A role's main duty refers to them by `turnsImageId`.

Right: `MODULE.{TOOLS_PAM}.{TURNS_IMAGES}` – *Pomocné dáta PAM → Obrazy turnusov*

8.4 Change log of the organisational structure (ChangeLog)

For incremental synchronisation of the code lists. Each endpoint returns the history versions of the records that were created or superseded in a date range. A row has `Operat` (`I` = version created, `D` = version superseded or corrected), `ValidFrom`, `ValidTo` and `Record` in the same shape as the matching list endpoint. An update therefore appears as a `D` row for the old version and an `I` row for the new one.

Parameters: `input.refDateFrom` and `input.refDateTo` (both required, `yyyy-MM-dd`, whole days).

> A change of an attribute other than the name or the validity, made within an existing validity interval, may not show up, and neither does a deactivation. Reload the full list periodically (for example weekly) to stay in sync.

api/ChangeLog/BaseOrs/OrgUnits

GET Changes of organisational units.

Right: `MODULE.{TOOLS_PAM}.{UNITS}` – *Pomocné dáta PAM → Útvary*

api/ChangeLog/BaseOrs/Departments

GET Changes of cost centres.

Right: `MODULE.{ORG_MAN}` – *Organizačné riadenie*

- Needs a different right than `api/BaseOrs/Departments`.

api/ChangeLog/BaseOrs/JobPositions

GET Changes of job positions.

Right: `MODULE.{TOOLS_PAM}.{POSITIONS}` – *Pomocné dáta PAM → Pracovné miesta*

api/ChangeLog/BaseOrs/PayPlaces

GET Changes of pay places.

Right: `MODULE.{TOOLS_PAM}.{PAY_PLACES}` – *Pomocné dáta PAM → Výplatné miesta*

8.5 Persons and employment relations (Pam)

Humanet keeps a **person** (one human being, with a personal number) separately from a **role** (one employment relation of that person in one organisation). A person can have several roles at once or one after another. Each role has a **main duty** (hlavný úväzok) that places it in the organisational structure and can have secondary duties.

```
Persons.id                = PersonRoles.personId
PersonRoles.id            = PersonRoleDutyMains.roleId = PersonRolePositions.roleId
PersonRoleDutyMains.orgUnitId / departmentId / positionId / functionId / jobId /
turnsImageId
                          → OrgUnits / Departments / JobPositions / Functions / Orders /
TurnsImages
```

> Join records by these numeric ids only. A role number (`roleNum`) and a personal number (`personalNumber`) are often equal, but not always.

Several responses contain personal data that most integrations do not need: the birth number (`personalIdentificationNumber`), birth date, IBAN and BIC. Do not store or pass on fields you do not need.

`api/Pam/Persons` DEFAULT

GET Lists persons: `id`, `personalNumber`, `firstName`, `surname`, `gender`, `maritStatus`, `nationality`, `birthDate`, `personalIdentificationNumber`, `note`.

POST Creates a person. All fields except `note` are required. Returns 201 with the person and its `id`.

PUT Updates the person with the given `id`. Every field is replaced, including empty ones, so send the complete record as read by GET.

Parameters: GET: `persNum` and `id`, both optional exact filters. Without them, all persons that are not deactivated (former employees included).

Right: `MODULE.{PERSON_DATA}.{PERSONS}` – *Personálne údaje → Osoby*

- Code values: `gender` key S003008 (`001` male, `002` female), `maritStatus` S003011, `nationality` S003033 (see `api/Base/Keys`).

`api/Pam/PersonRoles` DEFAULT

GET Lists roles: `id`, `roleNum`, `personId`, `companyId`, `roleType`, `roleSpec`, `validFrom` (start of employment), `validTo`, `endingReason`, `isMainRole`, `isActive`, `workPlaceId`, `iban`, `bic`, ….

POST Creates a role. Required: `personId`, `companyId`, `roleSpec` (key S003026), `validFrom`, `wageLawType`. An empty `roleNum` is generated when the database allows it. Returns 201 with the role and its `id`.

PUT Updates the role with the given `id` from the start of the current payroll period. Send the complete record: an empty `validTo` clears the end date and an empty `iban` switches the payment to cash.

Parameters: GET: `input.personId` (optional, the roles of one person) and `input.includeInactive` (optional, default `false`).

Right: `MODULE.{PERSON_DATA}.{ROLES}` – *Personálne údaje → Roly*

- `isActive` is an administrative flag that HR clears when it deactivates a role, typically some time after the employee has left. **Detect leavers by `validTo`**, not by `isActive`. To see ended relations that have already been deactivated, pass `input.includeInactive=true`.
- Do not filter on `isMainRole`; many databases do not use it.

- `isActive` in the request body is ignored. `roleSpec` can be changed only for a role that starts in the current month or later.

---

api/Pam/PersonRoleDutyMains DEFAULT

GET Lists the main duty of each role: `roleId`, `orgUnitId`, `departmentId`, `positionId`, `functionId`, `jobId`, `turnsImageId`, `workDifficultyLevel`. Values are ids; translate them with the code lists in 8.3.

Parameters: `roleId` (optional): one role; omit it for all roles.

Right: `MODULE.{PERSON_DATA}.{ROLES}` – *Personálne údaje → Roly*

- Duties of inactive roles are included too. Keep only rows whose `roleId` is in your list of roles.
- When the duty has no cost centre of its own, `departmentId` is the cost centre of its organisational unit.

---

api/Pam/PersonRolePositions DEFAULT

GET Lists all duties, main and secondary (`roleDutyId`, `roleId`, `isMain`), with the main and additional agenda fields (`agendaMain` … `agendaMain5`, `agendaAdditional1` … `5`).

Parameters: `roleId` (optional).

Right: `MODULE.{PERSON_DATA}.{ROLES}` – *Personálne údaje → Roly*

- `agendaMain` and `agendaMain2` come as one text "code, name"; split it on the first comma.

---

api/Pam/PersonExtendedImport DEFAULT

POST Creates or updates one employee in one call: the person, one role, the role's main duty with its organisational assignment, external identifier, primary address, titles, e-mail, citizenship and out-of-records status. Assignments are given by **code** (org unit, cost centre, job position, function, rota pattern, order) and workplace by name.

Right: `MODULE.{PERSON_DATA}.{ROLES}` – *Personálne údaje → Roly*

- **Write scope.** The call changes person, role, duty, address, title and e-mail data. Grant its right for reading and writing only to an integration that may change all of these.
- The person is found by `personalNumber`, then by birth number, then by `roleNum`; the role by `roleNum`. At least one of them must be given.
- An optional `validFrom` applies to everything in the request. Without it a new role starts at its `roleFrom` and changes are written from the start of the current payroll period.
- Code lists are never extended: an unknown code leaves the field unchanged. `roleSpec` and `wageLawType` are matched by their **name** (long or short), and an unknown value makes the call fail.
- Titles in the request replace the stored titles. An empty `iban` switches the payment to cash.
- **The call answers 201 even when parts of it were rejected.** Check the `Log` array of the response: `Level` 3 = warning, 4 = error, 5 = critical.

---

api/Pam/PersonAddress DEFAULT

GET Lists the addresses of a person (`?personId=`, required).

POST Creates an address. `addressType` key S003001 (`001` permanent, `002` temporary, `003` contact, `004` other); the post code must exist and the municipality must belong to it. Returns 201 with the `id`.

PUT Updates the address with the given `id` (complete record).

DELETE Deletes the address (`?id=`).

Right: `MODULE.{PERSON_DATA}.{PERSONS}` – *Personálne údaje → Osoby*

- Saving a new permanent address turns the previous permanent address into a contact address. Addresses have no history.

`api/Pam/PersonAddressMatch` DEFAULT

GET Checks whether one of two municipality / street pairs matches the permanent or temporary address of the person who owns a role. Returns `{"matchFound": true|false}`.

Parameters: `input.roleNum` (required), `input.municipality1`, `input.street1`, `input.municipality2`, `input.street2`. Municipalities are compared case-insensitively.

Right: `MODULE.{PERSON_DATA}.{PERSONS}` – *Personálne údaje → Osoby*

- A positive result can mean only a partial match of the address. Do not use it to verify identity or for access decisions; send both municipality and street.
- An unknown role number returns `matchFound=false`, not an error.

`api/Pam/PersonConnection` also `api/Pam/PersonConnections`

GET Lists the contacts of persons: `personId`, `itemIdx`, `typeCode` (key S001005), `item` (the value), `weight`, `specification`, `note`.

POST Adds a contact to a person. Returns 201 with the record and its `itemIdx`.

PUT Updates the contact identified by `personId` + `itemIdx`.

DELETE Deletes a contact (`?personId=&itemIdx=`).

Parameters: GET: `input.personId` (optional, omit for all persons) and `input.typeCode` (optional).

Right: `MODULE.{PERSON_DATA}.{PERSONS}.{IDENTIF}.{CONTACTS}` – *Personálne údaje → Osoby → Identifikácia → Kontakty*

- Contact types: `001` mobile, `002` phone, `003` home phone, `004` fax, `005` e-mail, `006` web, `007` e-mail for payslips. Always filter by `typeCode`; when there are several, the highest `weight` is the preferred one.
- The singular and plural paths are the same endpoint; Swagger lists the singular one.

`api/Pam/PersonRelatives` DEFAULT

GET Lists the family members of a person (`?personId=` , required).

POST Adds a family member. `relationType`: `001` spouse, `002` parent – child, `003` guardian – child. Returns 201 with the `id`.

PUT Updates the family member with the given `id` (names, birth date, gender, nationality, birth number, note). The relation type cannot be changed.

Right: `MODULE.{PERSON_DATA}.{PERSONS}` – *Personálne údaje → Osoby*

- `relationType` is not filled in the responses.

`api/Pam/VacationBalance` DEFAULT

GET For every active role: personal and role number, names, titles, org unit and cost centre code, role validity, and `vacationBalance`, the expected leave balance at the end of the year (this year's, last year's and additional leave together).

Parameters: `persNum` (optional): one person.

Right: `MODULE.{PERSON_DATA}.{ROLES}` – *Personálne údaje → Roly*

- Without `persNum` the balance is calculated for every role, which takes long in large databases. Call it at most once a day, outside working hours.



api/Pam/Payroll/SalaryParams

GET Payroll parameters of the person's main role, valid today: bank account and IBAN, monthly and hourly wage, average hourly earnings, deductions, wage execution flag, end of employment and probation, weekly working time.

Parameters: `persNum` (optional): one person; omit it for all active persons.

Right: `MODULE.{SALARY}.{ROLES}.{SALARY_ITEMS}.{PAM_ITEMS}` – *Spracovanie miezd → Roly → Platy a mzdy → Zložky mzdy*

- Slovak payroll only. The response holds salary and bank data: grant this right only to integrations that really need it.
- `isActive` in this response means that the person may use the partner attendance app, not that the role is active. An unknown `persNum` returns an empty list.

8.6 Attendance and absences (Attendance)

Attendance data is kept per role (`roleId` / `roleNum`). None of the attendance rights is in the default API group: the administrator has to add the rights listed below.

Unless stated otherwise, parameters are `input.from` / `input.to` as `yyyy-MM-dd` , inclusive.

> Filter only by role numbers you have read from `api/Pam/PersonRoles` , and check that every returned record belongs to the role you asked for.

api/Attendance/Transactions

GET Raw attendance transactions (arrivals, departures, passes) ordered by time: `id` , `roleId` , `roleNum` , `companyCode` , `cardNumber` , `transactionDate` , `effectiveDate` (the attendance day it counts for, e.g. a night shift), `direction` (0 in, 1 out, 2 pass), `transactionTypeId` / `Code` , `terminalCode` .

Parameters: `input.from` , `input.to` (required, at most 31 days), `input.roleNum` , `input.companyCode` (optional, exact).

Right: `PAGE.NALATTENDANCE.ROLES` – *Dochádzka → Roly* and `OPERATION.NALATTENDANCE.ROLES.CAN_VIEW_ALL` – *Zobrazenie rolí v podmodule Roly*

- The role is the holder of the card on the day of the transaction. Transactions of unassigned cards are returned only without a role or company filter.
- A system token needs both rights; without *Zobrazenie rolí v podmodule Roly* the call returns 403. The result is limited to the roles the access group may see.

api/Attendance/Real WRITES

GET Actual attendance calendar of one role: per day `Date` , `IsEditable` and `RealItems` (work shifts with times and duration; interruptions and extra pays are not included).

Parameters: `input.roleId` (internal role id), `input.from` , `input.to` .

Right: `PAGE.NALATTENDANCE.ROLES` – *Dochádzka → Roly*

> **This GET also writes.** Before answering, Humanet recalculates the role's attendance for the period (approved absences, breaks, durations, automatic extra pays) and saves the result, like an attendance processing run. Call it for one role and a short period at a time, never in a loop over all employees, and not while payroll is being closed. For raw clock-in data use `api/Attendance/Transactions` .

api/Attendance/Plan WRITES

GET Planned attendance of one role: per day the planned shift (`ShiftId` , texts, `Duration` , `TimeFrom` , `TimeTo`), `IsEditable` and `IsIndivPlan` .

POST Sets an individual planned shift for one role and day: `{RoleId, WorkShiftId, Date, Begin "hh:mm:ss", End}` ; an end before the start means the next day. An existing shift of that day is replaced. Returns the plan of that day.

DELETE Removes individual plan records of a role (`input.roleId` , `input.dateFrom` , `input.dateTo` , at most one month); the days fall back to the rota.

Parameters: GET: `input.roleId` , `input.from` , `input.to` .

Right: `PAGE.NALATTENDANCE.PLAN_CALENDAR` – *Dochádzka → Plánovací kalendár*

> **GET and POST also recalculate.** Before answering, Humanet recalculates the role's attendance for the period (approved absences, breaks, durations, automatic extra pays) and saves the result, like an attendance processing run. Call it for one role and a short period at a time, never in a loop over all employees, and not while payroll is being closed. For raw clock-in data use `api/Attendance/Transactions` .

- Only days in the editable period can be changed: after the last approved attendance day and within the role's rota pattern. Otherwise 400 "The date is not in editable range".

---

api/Attendance/Plan/Absences

GET Planned absences of roles: `roleNum` , `shiftId` , `from` , `to` , `isModified` .

Parameters: `input.from` , `input.to` (required), `input.roleId` or `input.roleNum` (optional).

Right: `PAGE.NALATTENDANCE.ROLES` – *Dochádzka → Roly*

- The period filters by the **start** of the absence: an absence that started before `from` is not returned even if it lasts into the period. Ask for a period that starts earlier and filter yourself.

---

api/Attendance/Real/Absences WRITES

GET Absences in the actual attendance: per day the items whose shift belongs to a shift group (`roleNum` , `shiftId` , `absenceType` , `timeFrom` , `timeTo`). Days without absences are left out.

Parameters: `input.from` , `input.to` (required), `input.roleNum` , `input.absencesShiftGroupCode` (code of the shift group whose shifts count as absences; default `SZ_NeprAPI` , which the customer defines in the attendance settings).

Right: `PAGE.NALATTENDANCE.ABSENCES` – *Dochádzka → Nemoci*

> **This GET also writes, for every role it returns.** Before answering, Humanet recalculates the role's attendance for the period (approved absences, breaks, durations, automatic extra pays) and saves the result, like an attendance processing run. Call it for one role and a short period at a time, never in a loop over all employees, and not while payroll is being closed. For raw clock-in data use `api/Attendance/Transactions` .

- The shift group must exist and contain shifts, otherwise the call fails.
- Without `roleNum` the call processes every role of the database, which can take very long.

---

api/Attendance/Illnesses

GET Sick leaves and other incapacities for work of roles in a period: `roleId` , `roleNum` , `persNum` , `personName` , `absenceTypeCode` , `absenceType` , `from` , `to` (null while it lasts).

Parameters: `input.from` , `input.to` (required), `input.roleNum` , `input.companyCode` (optional).

Right: `PAGE.NALATTENDANCE.ABSENCES` – *Dochádzka → Nemoci*

api/Attendance/Aba/AbsenceRequests

GET — Absence requests from the absence approval module in all states: `roleNum`, `persNum`, `personName`, `absenceType` / `absenceTypeCode`, `timeType` (1 whole day, 2 half day, 4 time interval), `from`, `to`, `statusCode`, `statusDescription`.

Parameters: `input.from`, `input.to` (required; requests that overlap the period), `input.roleNum`, `input.companyCode` (optional).

Right: `PAGE.NALATTENDANCE_ABA` – *Schvaľovanie neprítomností*

- Status codes: 1 waiting for approval, 2 cancelled, 3 approved, 4 rejected, 5 cancellation requested, 6 cancellation approved, 7 cancellation rejected, 9 approved, can still be rejected.

---

api/Attendance/AllAbsencesExt

GET — One merged overview of absences for calendars and planning tools: absence requests (waiting, approved, cancellation pending), sick leaves and out-of-records states, with name, colour, place, substitute and status.

Parameters: `input.from`, `input.to` (required), `input.persNr`: role numbers separated by `|` (at most 50), or empty for all.

Right: `PAGE.NALATTENDANCE_ABA.ABSENCE_OVERVIEW` – *Schvaľovanie neprítomností → Prehľad neprítomností*

- Roles are taken as they are valid today, so employees who have already left are not included, not even for past periods.

---

api/Attendance/PersonCards

GET — Persons with their attendance cards, for access and card systems: `id` (= `Persons.id`), `persNum`, `firstName`, `surname`, `email` (the e-mail contact with the highest weight) and `cards[]` (`cardNumber`, validity, today's assignment, `roleId`, `roleNum`, `companyCode`, `orgUnitCode`, `workPlaceCode`).

Parameters: Optional filters `input.companies`, `input.orgUnits`, `input.workPlaces` (codes). Repeat the parameter for several values: `?input.companies=001&input.companies=002`. Values of one filter are combined with OR, different filters with AND.

Right: `PAGE.NALATTENDANCE.EVIDENCES.ATTEND_CARDS` – *Dochádzka → Dochádzkové evidencie → Dochádzkové karty*

- The response also contains the e-mail of each person, including persons without a card. Consider this before granting the right; for contact data use `api/Pam/PersonConnection` with its own right.

---

api/Attendance/Cards

GET — All attendance cards (`cardNumber`, validity, note) with the assignment valid today (`roleId`, `roleNum`, `personName`, `assignedFrom`, `assignedTo`).

Right: `PAGE.NALATTENDANCE.EVIDENCES.ATTEND_CARDS` – *Dochádzka → Dochádzkové evidencie → Dochádzkové karty*

---

api/Attendance/RoleSettings

GET — Attendance settings of roles: data source, rota pattern, work schedule, time correction, break, extra-pay group, monthly account, wage transfer, shift group, self-approval, summary items, project groups.

Parameters: `input.roleId` or `input.roleNum` (optional).

Right: `PAGE.NALATTENDANCE.EVIDENCES` – *Dochádzka → Dochádzkové evidencie*

---

api/Attendance/RolePayrollData

GET Monthly values delivered by an external payroll system for roles: leave claim and remainder, doctor's visit hours (employee, family member, disabled person) claimed and drawn.

POST Stores these values for one role and month (`roleNum` , `year` , `month`). An existing record of that month is replaced completely; an omitted value becomes empty. Returns 201 with an empty body.

Parameters: GET: `year` (required), `month` , `roleNum` (optional).

Right: `PAGE.NALATTENDANCE` – *Dochádzka*

- For customers whose payroll is processed outside Humanet. GET returns active roles only.

---

api/Attendance/WorkHours

GET Worked hours per day for each role of a person: `PersonNum` , `RoleNum` , `WorkHoursDict` {date: hours}. The value is the summary item configured for the partner attendance app; days with zero are left out.

Parameters: `input.persNum` (personal number), `input.dateFrom` , `input.dateTo` .

Right: `PAGE.NALATTENDANCE` – *Dochádzka*

- Needs the partner app settings in the database; without them the call fails. Long periods are slow.

---

api/Attendance/WorkShifts

GET All defined work shifts, active and inactive: `Id` , `Code` , `Shortcut` , `Name` , `Begin` , `End` , `Active` . `ShiftId` in the calendars points here.

Right: `PAGE.NALATTENDANCE.EVIDENCES.WORK_SHIFTS` – *Dochádzka → Dochádzkové evidencie → Pracovné zmeny*

---

api/Attendance/Turns

GET Rotas of the attendance module: `id` , `code` , `name` , daily and weekly duty.

Right: `PAGE.NALATTENDANCE.EVIDENCES.TURNS` – *Dochádzka → Dochádzkové evidencie → Turnusy*

---

api/Attendance/TurnsImages

GET Active rota patterns of the attendance module with their rota, calendar, start offset and holiday rules.

Right: `PAGE.NALATTENDANCE.EVIDENCES.TURNS_IMAGES` – *Dochádzka → Dochádzkové evidencie → Obrazy turnusov*

---

api/Attendance/TransactionTypes

GET Transaction types (`id` , `code` , `shortcut` , `name` , `transactionIcon`) offered by the attendance emulator (web and mobile clock-in).

Right: `PAGE.NALATTENDANCE.EVIDENCES.TRANSACTION_TYPES` – *Dochádzka → Dochádzkové evidencie → Typy transakcií*

- Only the types of the terminal set as the attendance emulator are returned, so the list can be empty and need not contain every type found in `Transactions` .

8.7 Attendance terminals

These endpoints implement the protocol of the Humanet attendance terminal (CCR Premium Plus LTE). They are listed for completeness: only Humanet attendance terminals use them, and their manufacturers get the protocol description from HOUR. Integrations do not call them.

`api/Attendance/Terminal/Register`

POST Registers a new terminal in the database.

Right: Terminal protocol (provided by HOUR)

`api/Attendance/Terminal/Pair`

POST Completes the connection of a registered terminal after the administrator has approved it in Humanet.

Right: Terminal protocol (provided by HOUR)

`api/Attendance/Terminal/SysData` DEFAULT

POST Synchronises the terminal's settings and data with Humanet.

Right: `OPERATION.NALATTENDANCE.TERMINALS.ENABLE_CCR_PREMIUM_PLUS` – *Umožní komunikáciu s terminálom Humanet prostredníctvom Humanet API*

`api/Attendance/Transaction` DEFAULT

POST Uploads card transactions read on the terminal.

Right: `OPERATION.NALATTENDANCE.TERMINALS.ENABLE_CCR_PREMIUM_PLUS` – *Umožní komunikáciu s terminálom Humanet prostredníctvom Humanet API*

`api/Attendance/Card` DEFAULT

POST Registers a card read on the terminal as a new attendance card.

Right: `OPERATION.NALATTENDANCE.TERMINALS.ENABLE_CCR_PREMIUM_PLUS` – *Umožní komunikáciu s terminálom Humanet prostredníctvom Humanet API*

`api/Attendance/Terminal/Report` DEFAULT

POST Presence overview shown on the terminal display.

Right: `OPERATION.NALATTENDANCE.TERMINALS.ENABLE_CCR_PREMIUM_PLUS` – *Umožní komunikáciu s terminálom Humanet prostredníctvom Humanet API*

8.8 Invoicing (bookkeeping databases)

Available only in Humanet bookkeeping (ERP) databases; an HR database answers 501. The default API group of a bookkeeping database already has the rights.

A business document (obchodný doklad) has a `voucherType` from key S206008, for example `F` issued invoice.

`api/Invoicing/Vouchers` DEFAULT

GET Lists active business documents of one type: `id`, `voucherType`, `serie`, `number`, `amount`, `currency`, `voucherDate`, `partnerName`, states, attachments, contacts.

POST Creates a business document with its items. Returns 201 with `{id, serie, number, url, messages}`. The document gets the state "Importované webovou službou" and, where the type is set up so, is posted and moves stock.

Parameters: GET: `voucherType` (default `F`, case-sensitive), `from`, `to` (issue date, optional).

Right: `MODULE.{INVOICING}` – *Fakturácia a odbyt*

- The list is not paged. Always pass `from` and `to` and read long periods in monthly windows.
- POST is not idempotent: repeating it creates another document. A duplicate number answers 400. `dealerId` comes from `api/OrganizationalStructure/Dealers`.

`api/Invoicing/Vouchers/{id}` DEFAULT

GET Detail of one business document.

Right: `MODULE.{INVOICING}` – *Fakturácia a odbyt*

`api/Invoicing/Vouchers/{id}/Print` DEFAULT

GET The document as PDF (`Content-Type: application/octet-stream`, file name in `Content-Disposition`).

Parameters: `templateName` (default: the first template of the type, see `PrintTemplates`), `language` (`sk-SK`, `cs-CZ`, `de-DE`, `en-GB`, `fr-FR`; default `sk-SK`), `printAttachments`, `useItemDescriptions` (print item descriptions instead of names).

Right: `MODULE.{INVOICING}` – *Fakturácia a odbyt*

`api/Invoicing/Vouchers/PrintTemplates/{voucherType}` DEFAULT

GET Names of the print templates available for a document type.

Right: `MODULE.{INVOICING}` – *Fakturácia a odbyt*

`api/Invoicing/Vouchers/{id}/States` DEFAULT

GET State history of a document, newest first: `code` (key S206011), `timeStamp`, `description`.

POST Adds a state, for example `04001` sent by e-mail: `{code, description}`. The time is set by the server. Returns 201.

Right: `MODULE.{INVOICING}` – *Fakturácia a odbyt*

8.9 Bookkeeping, address book and dealers (bookkeeping databases)

Available only in bookkeeping (ERP) databases. The default API group of such a database has the rights.

`api/Accounting/Documents` DEFAULT

POST Creates an accounting document in the journal. `voucherType` key S201001 (`001` cash, `002` receivable, `003` liability, `004` bank, `005` internal); `voucherDate` is the issue date, `effectiveDate` the booking date (default: `voucherDate`). Series and number are generated when empty. Returns 201 with `{id, serie, number, url, messages}`.

Right: `MODULE.{BOOK_KEEPING_DIARY}` – *Denník*

- Every record needs an account on at least one side, and accounts must exist in the chart of accounts. The database must contain exactly one organisation.

`api/Accounting/Reports/assetbook/{year}` also **`api/Accounting/Reports/liabilitybook/{year}`**

DEFAULT

GET Book of receivables (issued invoices) or liabilities (received invoices): `voucherNumber`, `effectiveDate`, `dueDate`, `partner`, `ico`, `amount`, `vatAmount`, `amountToPay`, `paymentDate`.

Parameters: `year` in the path; `month` (1–12, optional). Without a month, the whole fiscal year that contains 1 January of `year`.

Right: `MODULE.{OUTPUT_BK}.{OBK_BOOKS}` – *Výstupy účtovníctva → Knihy*

`api/Accounting/Reports/mainbook/{year}` DEFAULT

GET General ledger by account: opening debit and credit, turnover in the period and balance.

Parameters: `year` in the path; `month` (1–12, optional).

Right: `MODULE.{OUTPUT_BK}.{OBK_FACTOR_LEDGER}` – *Výstupy účtovníctva → Hlavná kniha*

`api/AddressBook/Partners` DEFAULT

GET Active partners of the address book: `id`, `code`, `ico`, `dic`, `icdph`, names, `addresses[]`, `dealer`.

POST Creates a partner (`name` or `shortName` required). Returns 201 with `{id, code, url}`.

Parameters: GET: `ico`, `code` (optional, exact).

Right: `MODULE.{ADDRESS_BOOK}` – *Adresár partnerov*

- POST does not check duplicates: look the partner up by `ico` first. Send municipality and post code with every address.

`api/OrganizationalStructure/Dealers` DEFAULT

GET Active dealers (sales representatives): `id`, `code`, `shortName`, `name`.

Right: `MODULE.{ORS}.{DEALERS}` – *Organizačná štruktúra → Dealeri*

9 Typical integrations

The combinations below cover most integrations. Each lists the calls and the rights the API access group needs beyond the defaults.

9.1 Employee master data for another system

Access control, canteen, e-learning, intranet, BI or a supplier portal need the list of current employees with their organisational assignment.

1. Read the code lists: `BaseOrs/OrgUnits`, `BaseOrs/Departments`, `BaseOrs/JobPositions`, `PamTools/Functions`; keep maps id → code / name.
2. Read `Pam/Persons`, `Pam/PersonRoles` and `Pam/PersonRoleDutyMains`, each once and without filters.
3. Take the roles whose `validTo` is empty or in the future; each is one employment relation. Join person by `personId` and main duty by `roleId`, then translate the ids with the maps.
4. For e-mail and phone read `Pam/PersonConnection?input.typeCode=005` (e-mail) or `001` (mobile).

Extra rights: *Pomocné dáta PAM → Útvary, Strediská, Pracovné miesta, Funkcie*; for contacts *Personálne údaje → Osoby → Identifikácia → Kontakty*; for cards *Dochádzka → Dochádzkové evidencie → Dochádzkové karty*.

Seven to nine calls deliver the whole company. Run them once a day; compare with your previous run to find joiners, changes and leavers.

9.2 Creating employees from a recruiting or HR system

- For one call per new or changed employee, use `POST Pam/PersonExtendedImport`. Assignments are passed as codes, so create missing code-list items first (`POST BaseOrs/…`, `POST PamTools/Functions`). Always check the `Log` array of the answer.
- For finer control use `POST Pam/Persons`, then `POST Pam/PersonRoles` with the returned `personId`, then `POST Pam/PersonAddress` and `POST Pam/PersonConnection`.
- Agree with the customer's HR department which system is the master for which field. Humanet payroll depends on the role data: a wrong `validFrom` or an empty `iban` has immediate consequences in payroll.

Rights: *Personálne údaje → Osoby, Roly* for reading and writing (default); for contacts and code lists the rights from 9.1 for reading and writing.

9.3 Keeping the organisational structure in sync

Read the full lists once, then every night ask `ChangeLog/BaseOrs/…` for the previous day. Once a week reload the full lists, because deactivations and some attribute changes are not in the change log (8.4).

9.4 Attendance and absences for payroll, BI or planning

- Clock-ins: `Attendance/Transactions` in windows of at most 31 days. Translate `transactionTypeCode` with your own mapping and `cardNumber` / `roleId` with 9.1.
- Absences: `Attendance/Illnesses` (sick leave), `Attendance/Aba/AbsenceRequests` (requested and approved leave with status) or the merged `Attendance/AllAbsencesExt` for calendars.
- Shift definitions: `Attendance/WorkShifts`.

- Do not build exports on `Attendance/Real`, `Attendance/Plan` or `Attendance/Real/Absences`. They recalculate attendance on every call and are meant for single-role use (8.6).

Rights: *Dochádzka → Roly* and *Zobrazenie rolí v podmodule Roly* (transactions), *Dochádzka → Nemoci*, *Schvaľovanie neprítomností* and its *Prehľad neprítomností*, *Dochádzka → Dochádzkové evidencie → Pracovné zmeny*.

9.5 Card and access systems

`Attendance/PersonCards` returns everybody with their cards, filterable by organisation, org unit and workplace codes. A card is valid for a role when its assignment (`assignedFrom`, `assignedTo`) includes today. Right: *Dochádzka → Dochádzkové evidencie → Dochádzkové karty*.

9.6 Invoices from an e-shop or another sales system (bookkeeping database)

1. Find the customer in the address book: `GET AddressBook/Partners?ico=…`. Create it with `POST AddressBook/Partners` only when it does not exist.
2. Create the invoice with `POST Invoicing/Vouchers`; keep the returned `id`, series and number.
3. Get the PDF with `GET Invoicing/Vouchers/{id}/Print` and, after sending it, record the state with `POST Invoicing/Vouchers/{id}/States` (`04001` sent by e-mail).

All rights are in the default group of a bookkeeping database.

10 Go-live checklist and support

10.1 Before going live

- The database hash is in the configuration; the server is looked up through the portal (chapter 2) and a 440 triggers one new lookup.
- The token is in a secret store, not in code or logs. Its validity (*Platné do*) is in your calendar, together with whoever regenerates it.
- The API access group has exactly the rights the integration needs, for reading or for reading and writing. Every endpoint you use answered 200 in a test.
- The client reads the `X-Rate-Limit-*` headers, waits on 429 and retries 503 and network errors with a pause.
- Writes check the response: the `id` of the new record, and the `Log` of `PersonExtendedImport`.
- Personal data you do not need (birth number, birth date, IBAN) is neither stored nor passed on.
- Your client ignores response fields it does not know. New fields can be added to responses.

10.2 Support

Questions and problems go to the HOUR customer centre (Zákaznícke centrum HOUR) through the customer's usual support channel. Documentation of endpoints built for a specific customer is provided by the customer centre or the customer's project manager.

When you report a problem, include:

- the database name or hash and the server you called;
- the endpoint, method and parameters, and the time of the request with its time zone;
- the HTTP status and the response body, including the incident number from `Message`.

Never send the token, a password or a Basic string, not even to HOUR. If a token leaks, have the administrator regenerate it. If a password or a Basic string leaks, change the password and have the administrator check and revoke the tokens of that user; a new token alone is not enough.

A All endpoints at a glance

All 105 standard operations, in the order of chapter 8.

Method	Endpoint	Right
8.1 Access and system (Base, Master)		
POST	api/Base/Token/Generate	Humanet user name and password in the body
GET	api/Base/Token/Check	Any valid credentials
GET	api/Master/UserDatabases	Portal / Humanet login (Basic user:password)
GET	api/Base/Keys	No right needed (valid token only)
GET	api/Base/Sys/Time	No right needed (valid token only)
POST	api/Base/IO/EDE/Transfer	Administrator (all-modules right)
8.2 Companies and workplaces (BaseErp)		
GET	api/BaseErp/Companies	MODULE.{TOOLS_ERP}.{ORGANIZATIONS} DEFAULT
GET	api/BaseErp/Companies/Contacts	MODULE.{TOOLS_ERP}.{ORGANIZATIONS} DEFAULT
GET	api/BaseErp/Companies/Contacts/Emails	MODULE.{TOOLS_ERP}.{ORGANIZATIONS} DEFAULT
GET	api/BaseErp/Companies/Contacts/Phones	MODULE.{TOOLS_ERP}.{ORGANIZATIONS} DEFAULT
GET	api/BaseErp/Companies/Contacts/Others	MODULE.{TOOLS_ERP}.{ORGANIZATIONS} DEFAULT
GET	api/BaseErp/Companies/Logos	MODULE.{TOOLS_ERP}.{ORGANIZATIONS} DEFAULT
GET	api/BaseErp/Companies/Stamps	MODULE.{TOOLS_ERP}.{ORGANIZATIONS} DEFAULT
GET	api/BaseErp/Companies/Workplaces	MODULE.{TOOLS_ERP}.{ORGANIZATIONS} DEFAULT
8.3 Organisational structure and code lists (BaseOrs, PamTools)		
GET	api/BaseOrs/OrgUnits	MODULE.{TOOLS_PAM}.{UNITS}
POST	api/BaseOrs/OrgUnits	MODULE.{TOOLS_PAM}.{UNITS}
PUT	api/BaseOrs/OrgUnits	MODULE.{TOOLS_PAM}.{UNITS}
DELETE	api/BaseOrs/OrgUnits	MODULE.{TOOLS_PAM}.{UNITS}
GET	api/BaseOrs/Departments	MODULE.{TOOLS_PAM}.{DEPARTMENTS}
POST	api/BaseOrs/Departments	MODULE.{TOOLS_PAM}.{DEPARTMENTS}
PUT	api/BaseOrs/Departments	MODULE.{TOOLS_PAM}.{DEPARTMENTS}
DELETE	api/BaseOrs/Departments	MODULE.{TOOLS_PAM}.{DEPARTMENTS}
GET	api/BaseOrs/JobPositions	MODULE.{TOOLS_PAM}.{POSITIONS}
POST	api/BaseOrs/JobPositions	MODULE.{TOOLS_PAM}.{POSITIONS}
PUT	api/BaseOrs/JobPositions	MODULE.{TOOLS_PAM}.{POSITIONS}
DELETE	api/BaseOrs/JobPositions	MODULE.{TOOLS_PAM}.{POSITIONS}
GET	api/BaseOrs/PayPlaces	MODULE.{TOOLS_PAM}.{PAY_PLACES}
GET	api/PamTools/Functions	MODULE.{TOOLS_PAM}.{FUNCTIONS}
POST	api/PamTools/Functions	MODULE.{TOOLS_PAM}.{FUNCTIONS}
PUT	api/PamTools/Functions	MODULE.{TOOLS_PAM}.{FUNCTIONS}
DELETE	api/PamTools/Functions	MODULE.{TOOLS_PAM}.{FUNCTIONS}
GET	api/PamTools/Orders	MODULE.{TOOLS_PAM}.{ORDERS}
GET	api/PamTools/Turns	MODULE.{TOOLS_PAM}.{TURNS}
GET	api/PamTools/TurnsImages	MODULE.{TOOLS_PAM}.{TURNS_IMAGES}

Method	Endpoint	Right
8.4 Change log of the organisational structure (ChangeLog)		
GET	api/ChangeLog/BaseOrs/OrgUnits	MODULE.{TOOLS_PAM}.{UNITS}
GET	api/ChangeLog/BaseOrs/Departments	MODULE.{ORG_MAN}
GET	api/ChangeLog/BaseOrs/JobPositions	MODULE.{TOOLS_PAM}.{POSITIONS}
GET	api/ChangeLog/BaseOrs/PayPlaces	MODULE.{TOOLS_PAM}.{PAY_PLACES}
8.5 Persons and employment relations (Pam)		
GET	api/Pam/Persons	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
POST	api/Pam/Persons	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
PUT	api/Pam/Persons	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
GET	api/Pam/PersonRoles	MODULE.{PERSON_DATA}.{ROLES} DEFAULT
POST	api/Pam/PersonRoles	MODULE.{PERSON_DATA}.{ROLES} DEFAULT
PUT	api/Pam/PersonRoles	MODULE.{PERSON_DATA}.{ROLES} DEFAULT
GET	api/Pam/PersonRoleDutyMains	MODULE.{PERSON_DATA}.{ROLES} DEFAULT
GET	api/Pam/PersonRolePositions	MODULE.{PERSON_DATA}.{ROLES} DEFAULT
POST	api/Pam/PersonExtendedImport	MODULE.{PERSON_DATA}.{ROLES} DEFAULT
GET	api/Pam/PersonAddress	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
POST	api/Pam/PersonAddress	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
PUT	api/Pam/PersonAddress	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
DELETE	api/Pam/PersonAddress	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
GET	api/Pam/PersonAddressMatch	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
GET	api/Pam/PersonConnection	MODULE.{PERSON_DATA}.{PERSONS}.{IDENTIF}.{CONTACTS}
POST	api/Pam/PersonConnection	MODULE.{PERSON_DATA}.{PERSONS}.{IDENTIF}.{CONTACTS}
PUT	api/Pam/PersonConnection	MODULE.{PERSON_DATA}.{PERSONS}.{IDENTIF}.{CONTACTS}
DELETE	api/Pam/PersonConnection	MODULE.{PERSON_DATA}.{PERSONS}.{IDENTIF}.{CONTACTS}
GET	api/Pam/PersonConnections	MODULE.{PERSON_DATA}.{PERSONS}.{IDENTIF}.{CONTACTS}
POST	api/Pam/PersonConnections	MODULE.{PERSON_DATA}.{PERSONS}.{IDENTIF}.{CONTACTS}
PUT	api/Pam/PersonConnections	MODULE.{PERSON_DATA}.{PERSONS}.{IDENTIF}.{CONTACTS}
DELETE	api/Pam/PersonConnections	MODULE.{PERSON_DATA}.{PERSONS}.{IDENTIF}.{CONTACTS}
GET	api/Pam/PersonRelatives	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
POST	api/Pam/PersonRelatives	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
PUT	api/Pam/PersonRelatives	MODULE.{PERSON_DATA}.{PERSONS} DEFAULT
GET	api/Pam/VacationBalance	MODULE.{PERSON_DATA}.{ROLES} DEFAULT
GET	api/Pam/Payroll/SalaryParams	MODULE.{SALARY}.{ROLES}.{SALARY_ITEMS}.{PAM_ITEMS}
8.6 Attendance and absences (Attendance)		
GET	api/Attendance/Transactions	PAGE.NALATTENDANCE.ROLES + OPERATION.NALATTENDANCE.ROLES.CAN_VIEW_ALL
GET	api/Attendance/Real	PAGE.NALATTENDANCE.ROLES
GET	api/Attendance/Plan	PAGE.NALATTENDANCE.PLAN_CALENDAR
POST	api/Attendance/Plan	PAGE.NALATTENDANCE.PLAN_CALENDAR
DELETE	api/Attendance/Plan	PAGE.NALATTENDANCE.PLAN_CALENDAR
GET	api/Attendance/Plan/Absences	PAGE.NALATTENDANCE.ROLES

Method	Endpoint	Right
GET	api/Attendance/Real/Absences	PAGE.NALATTENDANCE.ABSENCES
GET	api/Attendance/Illnesses	PAGE.NALATTENDANCE.ABSENCES
GET	api/Attendance/Aba/AbsenceRequests	PAGE.NALATTENDANCE_ABA
GET	api/Attendance/AllAbsencesExt	PAGE.NALATTENDANCE_ABA.ABSENCE_OVERVIEW
GET	api/Attendance/PersonCards	PAGE.NALATTENDANCE.EVIDENCES.ATTEND_CARDS
GET	api/Attendance/Cards	PAGE.NALATTENDANCE.EVIDENCES.ATTEND_CARDS
GET	api/Attendance/RoleSettings	PAGE.NALATTENDANCE.EVIDENCES
GET	api/Attendance/RolePayrollData	PAGE.NALATTENDANCE
POST	api/Attendance/RolePayrollData	PAGE.NALATTENDANCE
GET	api/Attendance/WorkHours	PAGE.NALATTENDANCE
GET	api/Attendance/WorkShifts	PAGE.NALATTENDANCE.EVIDENCES.WORK_SHIFTS
GET	api/Attendance/Turns	PAGE.NALATTENDANCE.EVIDENCES.TURNS
GET	api/Attendance/TurnsImages	PAGE.NALATTENDANCE.EVIDENCES.TURNS_IMAGES
GET	api/Attendance/TransactionTypes	PAGE.NALATTENDANCE.EVIDENCES.TRANSACTION_TYPES
8.7 Attendance terminals		
POST	api/Attendance/Terminal/Register	Terminal protocol (provided by HOUR)
POST	api/Attendance/Terminal/Pair	Terminal protocol (provided by HOUR)
POST	api/Attendance/Terminal/SysData	OPERATION.NALATTENDANCE.TERMINALS.ENABLE_CCR_PREMIUM_PLUS DEFAULT
POST	api/Attendance/Transaction	OPERATION.NALATTENDANCE.TERMINALS.ENABLE_CCR_PREMIUM_PLUS DEFAULT
POST	api/Attendance/Card	OPERATION.NALATTENDANCE.TERMINALS.ENABLE_CCR_PREMIUM_PLUS DEFAULT
POST	api/Attendance/Terminal/Report	OPERATION.NALATTENDANCE.TERMINALS.ENABLE_CCR_PREMIUM_PLUS DEFAULT
8.8 Invoicing (bookkeeping databases)		
GET	api/Invoicing/Vouchers	MODULE.{INVOICING} DEFAULT
POST	api/Invoicing/Vouchers	MODULE.{INVOICING} DEFAULT
GET	api/Invoicing/Vouchers/{id}	MODULE.{INVOICING} DEFAULT
GET	api/Invoicing/Vouchers/{id}/Print	MODULE.{INVOICING} DEFAULT
GET	api/Invoicing/Vouchers/PrintTemplates/{voucherType}	MODULE.{INVOICING} DEFAULT
GET	api/Invoicing/Vouchers/{id}/States	MODULE.{INVOICING} DEFAULT
POST	api/Invoicing/Vouchers/{id}/States	MODULE.{INVOICING} DEFAULT
8.9 Bookkeeping, address book and dealers (bookkeeping databases)		
POST	api/Accounting/Documents	MODULE.{BOOK_KEEPING_DIARY} DEFAULT
GET	api/Accounting/Reports/assetbook/{year}	MODULE.{OUTPUT_BK}.{OBK_BOOKS} DEFAULT
GET	api/Accounting/Reports/liabilitybook/{year}	MODULE.{OUTPUT_BK}.{OBK_BOOKS} DEFAULT
GET	api/Accounting/Reports/mainbook/{year}	MODULE.{OUTPUT_BK}.{OBK_FACTOR_LEDGER} DEFAULT
GET	api/AddressBook/Partners	MODULE.{ADDRESS_BOOK} DEFAULT
POST	api/AddressBook/Partners	MODULE.{ADDRESS_BOOK} DEFAULT
GET	api/OrganizationalStructure/Dealers	MODULE.{ORS}.{DEALERS} DEFAULT

B Frequently used code lists

Values of a code list come from `GET api/Base/Keys?keyCode=…`; they can differ between legislations and customers, so load them rather than copying the examples.

Key	Meaning	Values	Used in
`S003008`	Gender	001 male, 002 female	`Persons.gender`
`S003011`	Marital status	load with `api/Base/Keys`	`Persons.maritStatus`
`S003033`	Nationality	load with `api/Base/Keys`	`Persons.nationality`
`S003001`	Address type	001 permanent, 002 temporary, 003 contact, 004 other	`PersonAddress.addressType`
`S001005`	Contact type	001 mobile, 002 phone, 003 home phone, 004 fax, 005 e-mail, 006 web, 007 payslip e-mail	`PersonConnection.typeCode`, `Companies/Contacts`
`S003026`	Role specification	load with `api/Base/Keys`	`PersonRoles.roleSpec`
`S003063`	Role type	load with `api/Base/Keys`	`PersonRoles.roleType`
`S003015`	Wage-law type	load with `api/Base/Keys`	`PersonRoles.wageLawType`
`S003017`	Reason of start	load with `api/Base/Keys`	`PersonRoles`
`S003054`	Reason of end	load with `api/Base/Keys`	`PersonRoles.endingReason`
`S003018`	Way of ending	load with `api/Base/Keys`	`PersonRoles`
`S003053`	Contract duration	load with `api/Base/Keys`	`PersonRoles`
`S003071`	Probation period	load with `api/Base/Keys`	`PersonRoles`
`S001002`	Organisation type	001 organisation run in Humanet	`BaseErp/Companies.type`
`S201001`	Accounting document type	001 cash, 002 receivable, 003 liability, 004 bank, 005 internal	`Accounting/Documents.voucherType`
`S206008`	Business document type	F issued invoice, …	`Invoicing/Vouchers.voucherType`
`S206011`	Business document state	01001 issued, 02001 printed, 03001 transferred to bookkeeping, 04001 sent by e-mail, …	`Invoicing/Vouchers/{id}/States.code`

Version 1.0 · 2. 11. 2026 · HOUR, spol. s r.o. · Describes the standard endpoints published in the production Swagger on the date of the version.