Asking the system instead of exporting by hand

Two endpoints that answer different questions, the dimension syntax that addresses a cube, and the four things to get right before any of it — authentication, paging, rate limits and the fact that analytics tables are stale until someone runs them.

Pierre Robentz Cassion

Lesson 5 of 8

Routine Data and DHIS2 — Getting the data out

What this lesson covers

- The monthly export is the problem
- Two endpoints, two questions
- Addressing the cube
- The response shape
- The four things to get right first
- Pull the metadata too
- What comes next

The monthly export is the problem

- Somebody opens the interface, picks the org units, picks the periods, picks the data elements, clicks export, and emails a spreadsheet.

Two endpoints, two questions

Endpoint	Returns	Use it for
`/api/dataValueSets`	Raw stored values, exactly as entered	Reproducing the register; anything where you must see what was typed
`/api/analytics`	Aggregated, calculated values	Indicators, totals up the hierarchy, anything the system computes

Two endpoints, two questions

- `dataValueSets` gives you the truth as stored — one row per data element, org unit, period and category option...
- `analytics` gives you the answer the system would show on a dashboard — aggregated up the hierarchy, indicators...
- Pull both when the two are supposed to agree — Where they do not, the gap is an aggregation rule you have not accounted...

Addressing the cube — Example

```
GET /api/analytics.json
  ?dimension=dx:PENTA3_UID;BCG_UID
  &dimension=ou:LEVEL-4;DISTRICT_UID
  &dimension=pe:202401;202402;202403
  &displayProperty=NAME
```

Addressing the cube

- `dx` — data dimension: data elements, indicators, data element operands.
- `ou` — org units. `LEVEL-4` means every unit at level 4; putting a district UID beside it scopes to that subtree.
- `pe` — periods, either explicit (`202401`) or relative (`LAST_12_MONTHS`).
- Prefer explicit periods over relative ones in a script — `LAST_12_MONTHS` returns a different window every month, which...

Addressing the cube — In Python

```python
import requests

BASE = "https://play.dhis2.org/40/api"

def analytics(dx, ou, pe, session):
    response = session.get(
        f"{BASE}/analytics.json",
        params={
            "dimension": [f"dx:{';'.join(dx)}", f"ou:{ou}", f"pe:{';'.join(pe)}"],
            "displayProperty": "NAME",
            "skipMeta": "false",
        },
        timeout=120,
    )
    response.raise_for_status()
    return response.json()
```

Addressing the cube — In R

```r
library(httr2)

analytics <- function(dx, ou, pe) {
  request("https://play.dhis2.org/40/api/analytics.json") |>
    req_url_query(
      dimension = paste0("dx:", paste(dx, collapse = ";")),
      dimension = paste0("ou:", ou),
      dimension = paste0("pe:", paste(pe, collapse = ";")),
      displayProperty = "NAME",
      .multi = "explode"
    ) |>
    req_perform() |>
    resp_body_json()
}
```

The response shape — In Python

```
def to_frame(payload):
    import pandas as pd

    columns = [h["name"] for h in payload["headers"]]
    frame = pd.DataFrame(payload["rows"], columns=columns)

    names = {uid: item["name"]
             for uid, item in payload["metaData"]["items"].items()}
    for column in ("dx", "ou", "pe"):
        if column in frame:
            frame[f"{column}_name"] = frame[column].map(names)
    return frame
```

The response shape — In R

```r
to_frame <- function(payload) {
  cols <- vapply(payload$headers, \(h) h$name, character(1))
  rows <- do.call(rbind, lapply(payload$rows, \(r) unlist(r)))
  as.data.frame(rows, stringsAsFactors = FALSE) |> setNames(cols)
}
```

The response shape

- Keep the UIDs as well as the names — Names change; UIDs do not

The four things to get right first

- Authentication — Use a personal access token where the instance supports it, basic auth otherwise, and never put either...

The four things to get right first — In Python

```
import os

session = requests.Session()
session.headers["Authorization"] = f"ApiToken {os.environ['DHIS2_TOKEN']}"
```

The four things to get right first — In R

```
req_headers(request(url), Authorization = paste("ApiToken", Sys.getenv("DHIS2_TOKEN")))
```

The four things to get right first

- Paging — Metadata endpoints page by default at 50

The four things to get right first — In Python

```python
def paged(url, session, page_size=1000):
    page = 1
    while True:
        payload = session.get(url, params={"page": page, "pageSize": page_size},
                              timeout=120).json()
        yield payload
        pager = payload.get("pager", {})
        if page >= pager.get("pageCount", 1):
            return
        page += 1
```

The four things to get right first — In R

```
# Same idea: loop until page == pageCount, and never assume one request is all of it.
```

The four things to get right first

- Rate and size — Ask for one district and one year at a time, not the country and five years
- Analytics tables are stale — This is the one that surprises people

The four things to get right first — In Python

```
info = session.get(f"{BASE}/system/info", timeout=60).json()
print(info.get("lastAnalyticsTableSuccess"))
```

The four things to get right first — In R

```
resp <- request(paste0(base, "/system/info")) |> req_perform() |> resp_body_json()
resp$lastAnalyticsTableSuccess
```

The four things to get right first

- Record that timestamp with every extract — It is the difference between "the numbers changed" and "the numbers changed...

Pull the metadata too — Example

```
/api/dataElements.json?fields=id,name,aggregationType,categoryCombo[id,name]
/api/organisationUnits.json?fields=id,name,level,parent[id]&paging=false
/api/dataSets.json?fields=id,name,periodType,organisationUnits~size
/api/indicators.json?fields=id,name,numerator,denominator,indicatorType[name]
```

What comes next

- You can ask the system a question.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/routine-data-and-dhis2/dhis2-05-the-web-api