An extract you can point at

One script, one dated folder, a manifest that records what was asked and when, and the raw response kept beside the tidy table — so a figure published in March can still be defended in September.

Pierre Robentz Cassion

Lesson 6 of 8

Routine Data and DHIS2 — Getting the data out

What this lesson covers

- The question this lesson answers
- The shape
- The manifest
- The script
- Validate at the boundary
- Diff against the last extract
- What this replaces
- What comes next

The question this lesson answers

- Six months after you publish a district coverage figure, somebody reruns the query and gets something else.

The shape — Example

```
extracts/
  2026-07-28/
    manifest.json          what was requested, by whom, when
    raw/
      analytics.json       the response, untouched
      dataValueSets.json
      dataElements.json
      organisationUnits.json
      dataSets.json
      indicators.json
    tidy/
      coverage_by_facility_period.csv
```

The shape

- One dated folder per extraction — never overwritten
- The raw response is kept untouched — If your parsing turns out to be wrong, the data is still there
- The manifest is the point — Everything else is recoverable from it

The manifest — In Python (cont.)

```
import json
import datetime as dt
from pathlib import Path

def write_manifest(folder, request, system_info):
    manifest = {
        "extracted_at": dt.datetime.now(dt.timezone.utc).isoformat(),
        "extracted_by": "me-officer@example.org",
        "instance": request["base_url"],
        "dhis2_version": system_info.get("version"),
        "analytics_last_run": system_info.get("lastAnalyticsTableSuccess"),
        "request": {
            "endpoint": request["endpoint"],
            "dx": request["dx"],
            "ou": request["ou"],
            "pe": request["pe"],
```

The manifest — In Python (cont.)

```
    },
    "rows_returned": request["rows"],
    "script_version": "extract.py 1.3",
}
Path(folder, "manifest.json").write_text(json.dumps(manifest, indent=2))
return manifest
```

The manifest — In R

```r
manifest <- list(
  extracted_at        = format(Sys.time(), "%Y-%m-%dT%H:%M:%SZ", tz = "UTC"),
  instance            = base_url,
  analytics_last_run  = system_info$lastAnalyticsTableSuccess,
  request             = list(endpoint = "analytics", dx = dx, ou = ou, pe = pe),
  rows_returned       = nrow(tidy),
  script_version      = "extract.R 1.3"
)

jsonlite::write_json(manifest, file.path(folder, "manifest.json"), auto_unbox = TRUE)
```

The script — In Python (cont.)

```
def extract(base_url, dx, ou, pe, token, out_root="extracts"):
    session = requests.Session()
    session.headers["Authorization"] = f"ApiToken {token}"

    folder = Path(out_root, dt.date.today().isoformat())
    (folder / "raw").mkdir(parents=True, exist_ok=True)
    (folder / "tidy").mkdir(exist_ok=True)

    info = session.get(f"{base_url}/system/info", timeout=60).json()

    payload = analytics(dx, ou, pe, session)
    (folder / "raw" / "analytics.json").write_text(json.dumps(payload))

    for name, path in METADATA_ENDPOINTS.items():
        meta = session.get(f"{base_url}/{path}", timeout=120).json()
        (folder / "raw" / f"{name}.json").write_text(json.dumps(meta))
```

The script — In Python (cont.)

```
tidy = to_frame(payload)
tidy.to_csv(folder / "tidy" / "coverage_by_facility_period.csv", index=False)

write_manifest(folder, {"base_url": base_url, "endpoint": "analytics",
                        "dx": dx, "ou": ou, "pe": pe, "rows": len(tidy)}, info)
return folder
```

The script — In R

```r
extract <- function(base_url, dx, ou, pe, token) {
  folder <- file.path("extracts", Sys.Date())
  dir.create(file.path(folder, "raw"), recursive = TRUE, showWarnings = FALSE)
  dir.create(file.path(folder, "tidy"), showWarnings = FALSE)
  # ... same shape: system info, analytics, metadata, tidy, manifest
  folder
}
```

The script

- The window is computed, then passed in — The caller decides which twelve months; the script records them

The script — In Python

```
def last_full_months(n, today=None):
    today = today or dt.date.today()
    first_of_this = today.replace(day=1)
    months = []
    for i in range(n, 0, -1):
        m = first_of_this - dt.timedelta(days=1)
        for _ in range(i - 1):
            m = m.replace(day=1) - dt.timedelta(days=1)
        months.append(m.strftime("%Y%m"))
    return months
```

The script — In R

```r
last_full_months <- function(n) {
  ends <- seq(Sys.Date(), by = "-1 month", length.out = n + 1)[-1]
  rev(format(ends, "%Y%m"))
}
```

Validate at the boundary — In Python

```python
def check(tidy, expected_org_units, expected_periods):
    problems = []
    if tidy["ou"].nunique() != expected_org_units:
        problems.append(f"{tidy['ou'].nunique()} org units, expected {expected_org_units}")
    if tidy["pe"].nunique() != expected_periods:
        problems.append(f"{tidy['pe'].nunique()} periods, expected {expected_periods}")
    if tidy["value"].isna().any():
        problems.append("null values in the response")
    return problems
```

Validate at the boundary — In R

```r
check <- function(tidy, expected_ou, expected_pe) {
  c(if (n_distinct(tidy$ou) != expected_ou) "unexpected org unit count",
    if (n_distinct(tidy$pe) != expected_pe) "unexpected period count")
}
```

Diff against the last extract — In Python

```
def compare(old_folder, new_folder, keys=("ou", "pe", "dx")):
    old = pd.read_csv(Path(old_folder, "tidy", "coverage_by_facility_period.csv"))
    new = pd.read_csv(Path(new_folder, "tidy", "coverage_by_facility_period.csv"))

    merged = old.merge(new, on=list(keys), how="outer",
                       suffixes=("_old", "_new"), indicator=True)
    changed = merged[merged["value_old"] != merged["value_new"]]
    return {
        "in_both": int((merged["_merge"] == "both").sum()),
        "new_only": int((merged["_merge"] == "right_only").sum()),
        "gone": int((merged["_merge"] == "left_only").sum()),
        "values_changed": len(changed),
    }
```

Diff against the last extract — In R

```r
compare <- function(old, new) {
  full_join(old, new, by = c("ou", "pe", "dx"), suffix = c("_old", "_new")) |>
    summarise(changed = sum(value_old != value_new, na.rm = TRUE),
              gone = sum(is.na(value_new)), added = sum(is.na(value_old)))
}
```

What this replaces — Example

```
Before:  a spreadsheet in an email, monthly, method unknown
After:   extracts/2026-07-28/, one command, manifest, raw responses,
         a diff against last month, and a figure you can defend in September
```

What comes next

- You have a routine figure you can point at.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/routine-data-and-dhis2/

dhis2-06-a-reproducible-extract