



1-5
June
2026

Metadata Harmonization of GNSS Data Repositories

Rui Fernandes¹, Fernando Geraldes¹, David Mencin², Alex Hamilton², Henry Berglund²,
Markus Bradke³, Nils Brinckmann³, Brandon Owen⁴, Ryan Ruddick⁴

¹ UBI, Portugal

² EarthScope, U.S.A.

³ GFZ, Germany

⁴ Geosciences Australia, Australia

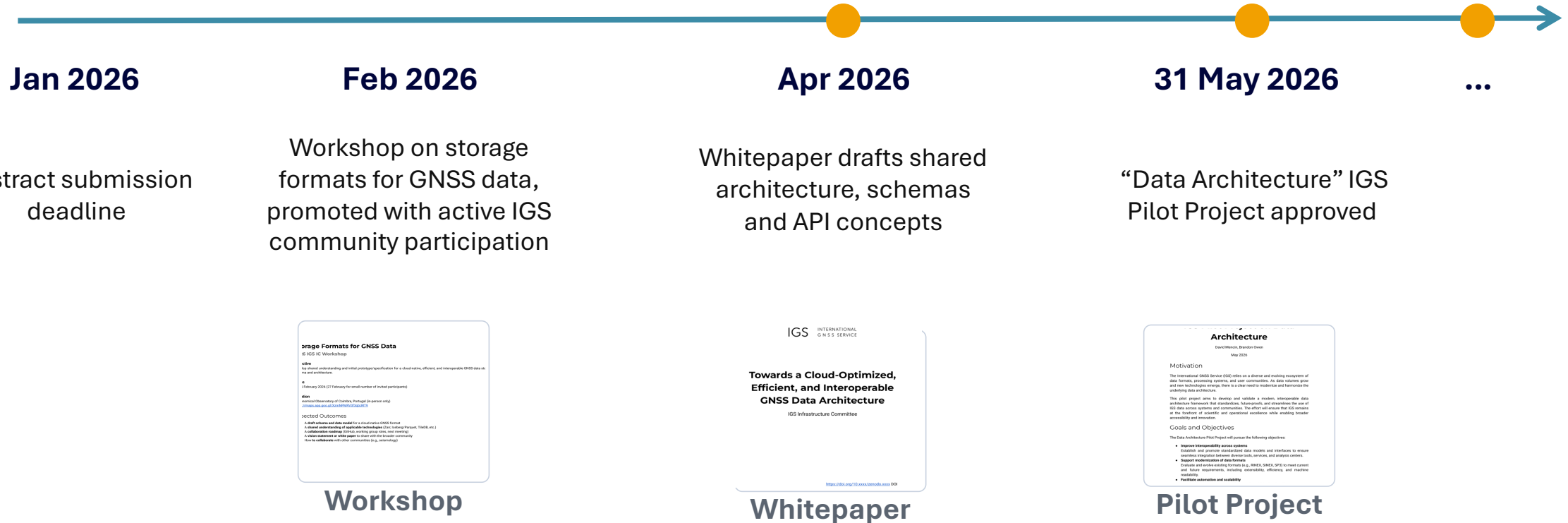
ORGANIZING INSTITUTIONS



SANTIAGO – CHILE

www.igs-workshop-2026.com

From independent implementations to community common effort





1-5
June
2026

Part A: New GNSS Data Formats

ORGANIZING INSTITUTIONS



SANTIAGO - CHILE

www.igs-workshop-2026.com

From RINEX files to interoperable GNSS data services

- **Why change?**

- RINEX remains essential, but file-based archives do not scale for modern discovery, sub-setting and federation.

- **What is changing?**

- Structured archives using array, columnar or table formats; APIs generate user-facing products on demand.

- **Why metadata?**

- Federation requires common, queryable information on what each repository contains.

- **What is being built?**

- Standard data models, availability schemas and APIs for repository discovery, comparison and federated access.

RINEX is the standard, but file archives are now a bottleneck

Main Limitations

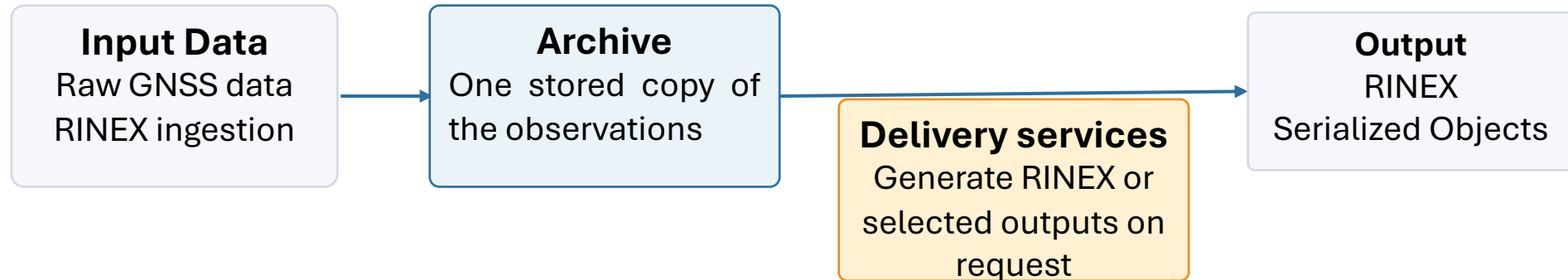
- Access is organized around files, not queryable time windows.
- Subsetting by time, satellite, signal or observable requires reading full files.
- The same measurements are often repeated in separate files at different sampling rates.
- Metadata corrections require regenerating or republishing files.
- Millions of small files are inefficient in cloud/object storage.

R07	114852540.567	7	21455412.578	7	46.000	114852574.574	7	21455413.038	7	45.500	89329846.134	7	21455423.595	7	45.750	89329853.1
R08	183737282.959	8	19372183.138	8	49.250	183737271.966	8	19372183.894	8	48.750	89584546.992	8	19372192.019	8	50.250	89584637.0
R09	114688189.597	7	21477378.118	7	46.750	114688128.000	7	21477378.055	7	46.500	89281954.812	7	21477386.002	7	47.000	89281954.0
R10	128584175.391	6	22821266.841	6	41.250	128584178.390	6	22821266.407	6	40.800						
R16	124390035.773	6	23286066.002	6	40.750	124390051.765	6	23286067.125	6	40.500	96747884.695	6	23286077.003	6	38.250	96747892.6
R17	128139925.225	6	23946088.810	6	36.750	128139928.228	5	23946088.008	5	35.750	99664442.971	6	23946080.312	6	37.250	99664436.9
R22	129505308.583	5	24260747.378	5	34.250	129505314.584	5	24260746.896	5	33.750	100726331.782	5	24260745.206	5	32.750	100726319.8
R23	113691190.811	7	21253309.362	7	45.750	113691176.818	7	21253309.163	7	44.750						
R24	118749182.282	8	18718649.048	8	49.750	118749166.383	8	18718649.241	8	49.250	86138298.398	7	28718657.538	7	46.750	86138290.3

- Sometimes I only want certain constellations
- Sometimes I only want certain observables (e.g. phase+range vs SNR)
- Sometimes I want 30 hours of data so I have full satellite arcs on either end of the UTC day

Target model: data-centered archive instead of a file-centered archive

Keep one stored representation of the observations and generate the required files when needed



RINEX remains essential for interoperability, while the storage structure can be better suited to large, sparse GNSS data.

EarthScope operational experience

Example of structured GNSS observation storage and API-based access.

Operational Pattern

- GNSS observations stored in a structured multidimensional data store.
- Station metadata managed separately.
- RINEX 2, 3, and 4 generated on request.
- Access services support selective extraction.
- SDK (Client tools) simplify access for users and applications.
- Architecture optimized for cloud object storage.

```
from earthscope_sdk import EarthScopeClient
```

```
es = EarthScopeClient() ← transparent auth handling
```

```
table = es.data.gnss_observations( ← no URL building  
    start_datetime=datetime(2025, 7, 1), ← Python types  
    end_datetime=datetime(2025, 7, 30),  
    station_name="AC60",
```

EarthScope SDK: user requests what is needed

```
table = await es.data.gnss_observations(  
    start_datetime=dt.datetime(2025, 8, 1),  
    end_datetime=dt.datetime(2025, 8, 30),  
    station_name="AC60",  
    system="G", ← just GPS  
    obs_code=["1C", "2W"], ← just signals 1C & 2W  
    field=["phase", "range"], ← just phase & range
```

Server-side subsetting by time, system, code and field

Technology landscape: storage and access layers

Several implementations are being explored; interoperability depends on common schemas (data and metadata) and interfaces.

Storage candidates

TileDB

array storage engine for sparse or dense multidimensional data

Parquet

columnar file format for efficient storage and analytical access

Iceberg

table metadata layer for managing large file-based datasets in object storage

Access / interchange

Arrow

in-memory columnar format for efficient data exchange between storage, APIs, and processing tools

EarthScope is using TileDB operationally, but the community is still evaluating alternative storage approaches.



 **1-5**
June
2026

Part B:

Metadata Harmonization of GNSS Data Repositories

ORGANIZING INSTITUTIONS



SANTIAGO – CHILE

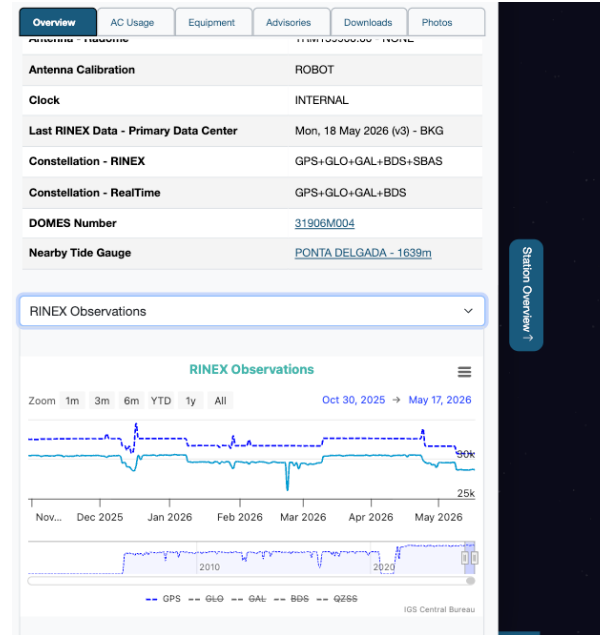
www.igs-workshop-2026.com

"What data do you exactly have?"

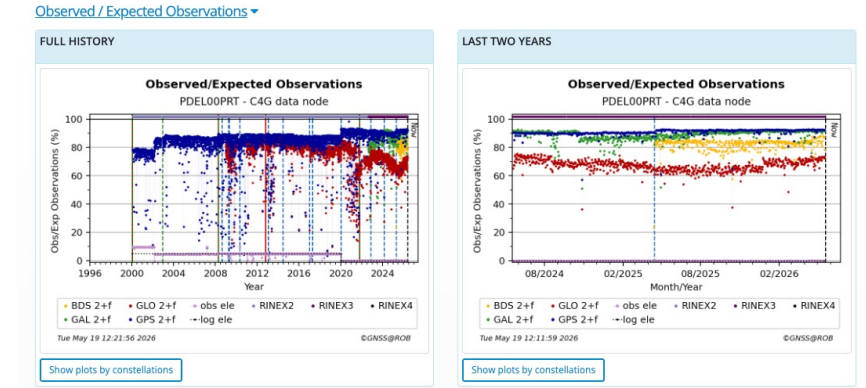
A service must know what each repository contains.

Metadata Attributes

- Station and stream identifiers
- Time windows and sampling
- Constellations, satellites and signals
- Epoch completeness and observation counts
- Integrity information for synchronization



IGS



EPOS



EarthScope

Current metadata representations are not directly comparable

EarthScope and EPOS stores similar information, but through different storage paths and conventions.

EarthScope

Data

- Observation data in array-oriented structures.

Metadata

- Metadata separated from observations.
- Availability derived from array contents and operational indexes.

EPOS

Data

- GNSS data distributed as RINEX files.

Metadata

- Availability and quality information from relational metadata.
- Availability derived from files and QC/database records.

Current metadata representations are not directly comparable

EarthScope

- station_edid
- session_edid
- observation window
- epoch_count / expected_epoch_count
- satellite masks
- signal masks
- system mask
- ...

EPOS

- station_id
- reference_date
- Constellation
- Nsat
- epo_expt / epo_have / epo_usbl
- xint_slp
- obstype
- frequency_band
- obs_have / obs_expt / obs_sats
- pha_slps / cod_mpth
- elevation_cutoff
- ...

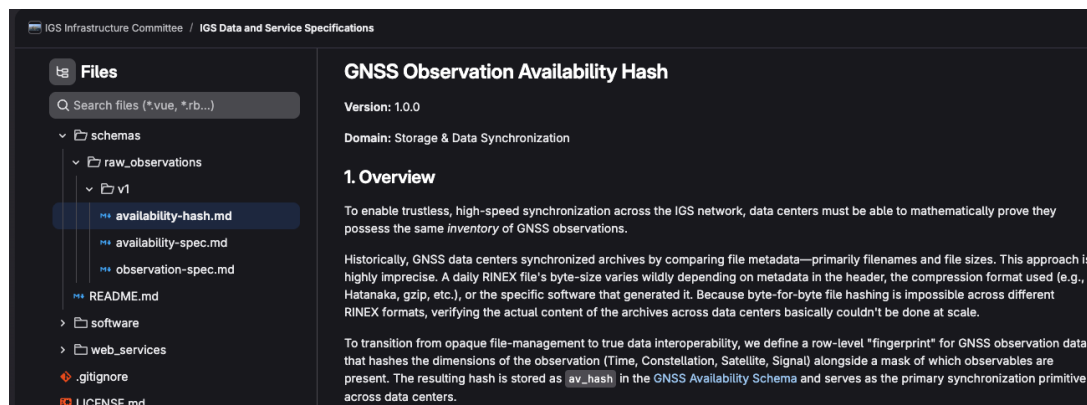
A common schema **must define the necessary attributes first;**
each repository then maps its current implementation to those attributes.

Metadata Schema – IGS community

Minimum information needed to describe holdings in a machine-queryable way.

<https://git.gfz-potsdam.de/igs/igs-data-specifications>

- The current discussion is centered on which metadata attributes are fundamental and must be defined consistently across repositories.
- Many of these attributes are represented as masks within each time bin, for example the presence of constellations, satellites, signals, and observables.



IGS Infrastructure Committee / IGS Data and Service Specifications

Files

Search files (*.vue, *.rb...)

- ✓ schemas
 - ✓ raw_observations
 - ✓ v1
 - availability-hash.md
 - availability-spec.md
 - observation-spec.md
 - README.md
 - software
 - web_services
 - .gitignore
 - LICENSE.md
 - README.md

GNSS Observation Availability Hash

Version: 1.0.0

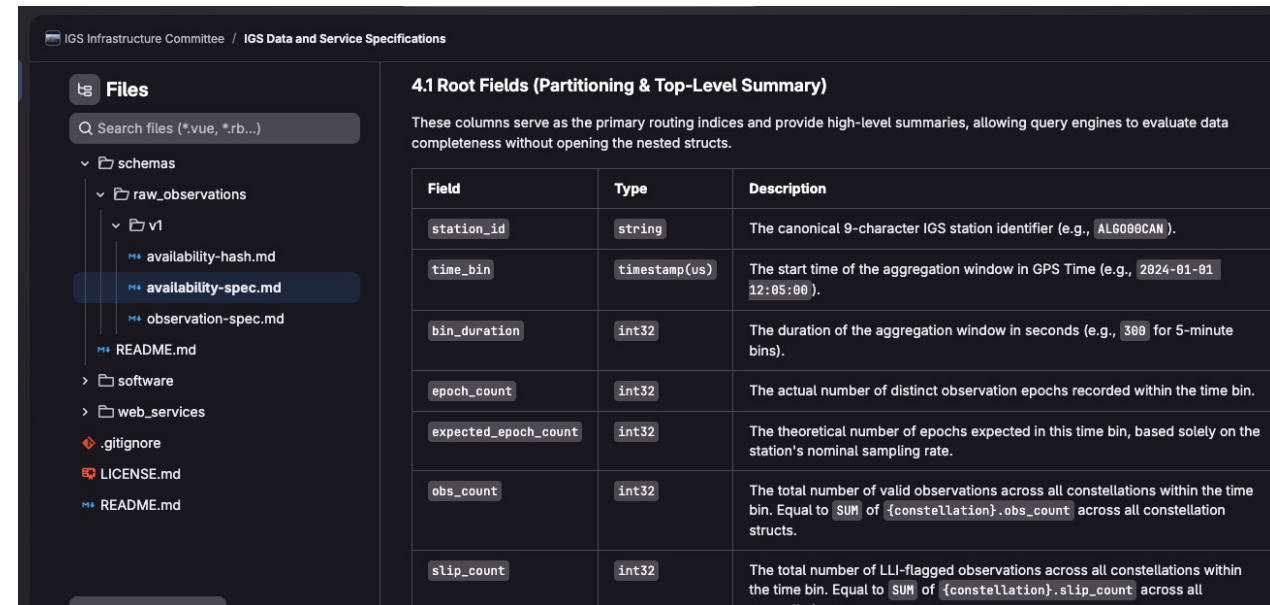
Domain: Storage & Data Synchronization

1. Overview

To enable trustless, high-speed synchronization across the IGS network, data centers must be able to mathematically prove they possess the same *inventory* of GNSS observations.

Historically, GNSS data centers synchronized archives by comparing file metadata—primarily filenames and file sizes. This approach is highly imprecise. A daily RINEX file's byte-size varies wildly depending on metadata in the header, the compression format used (e.g., Hatanaka, gzip, etc.), or the specific software that generated it. Because byte-for-byte file hashing is impossible across different RINEX formats, verifying the actual content of the archives across data centers basically couldn't be done at scale.

To transition from opaque file-management to true data interoperability, we define a row-level "fingerprint" for GNSS observation data that hashes the dimensions of the observation (Time, Constellation, Satellite, Signal) alongside a mask of which observables are present. The resulting hash is stored as `av_hash` in the GNSS Availability Schema and serves as the primary synchronization primitive across data centers.



IGS Infrastructure Committee / IGS Data and Service Specifications

Files

Search files (*.vue, *.rb...)

4.1 Root Fields (Partitioning & Top-Level Summary)

These columns serve as the primary routing indices and provide high-level summaries, allowing query engines to evaluate data completeness without opening the nested structs.

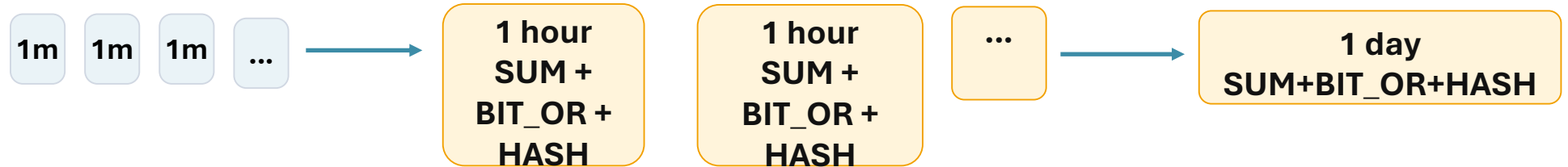
Field	Type	Description
<code>station_id</code>	string	The canonical 9-character IGS station identifier (e.g., <code>ALG000CAN</code>).
<code>time_bin</code>	timestamp(us)	The start time of the aggregation window in GPS Time (e.g., <code>2024-01-01 12:05:00</code>).
<code>bin_duration</code>	int32	The duration of the aggregation window in seconds (e.g., <code>300</code> for 5-minute bins).
<code>epoch_count</code>	int32	The actual number of distinct observation epochs recorded within the time bin.
<code>expected_epoch_count</code>	int32	The theoretical number of epochs expected in this time bin, based solely on the station's nominal sampling rate.
<code>obs_count</code>	int32	The total number of valid observations across all constellations within the time bin. Equal to <code>SUM of {constellation}.obs_count</code> across all constellation structs.
<code>slip_count</code>	int32	The total number of LLI-flagged observations across all constellations within the time bin. Equal to <code>SUM of {constellation}.slip_count</code> across all constellation structs.

- A key field under discussion is `av_hash`, which allows fast comparison of observation content between repositories for the same time bin.

Time-Bin & Aggregation (Operationalization)

Based on existing tools made available by EarthScope and tested at EPOS repositories – developed API by UBI

Current
EarthScope



Evaluation



Counts aggregate by SUM:

- epochs
- observations
- LLI / raw flags

Presences aggregate by BIT_OR:

- constellations
- satellites
- signals

Integrity HASH enables:

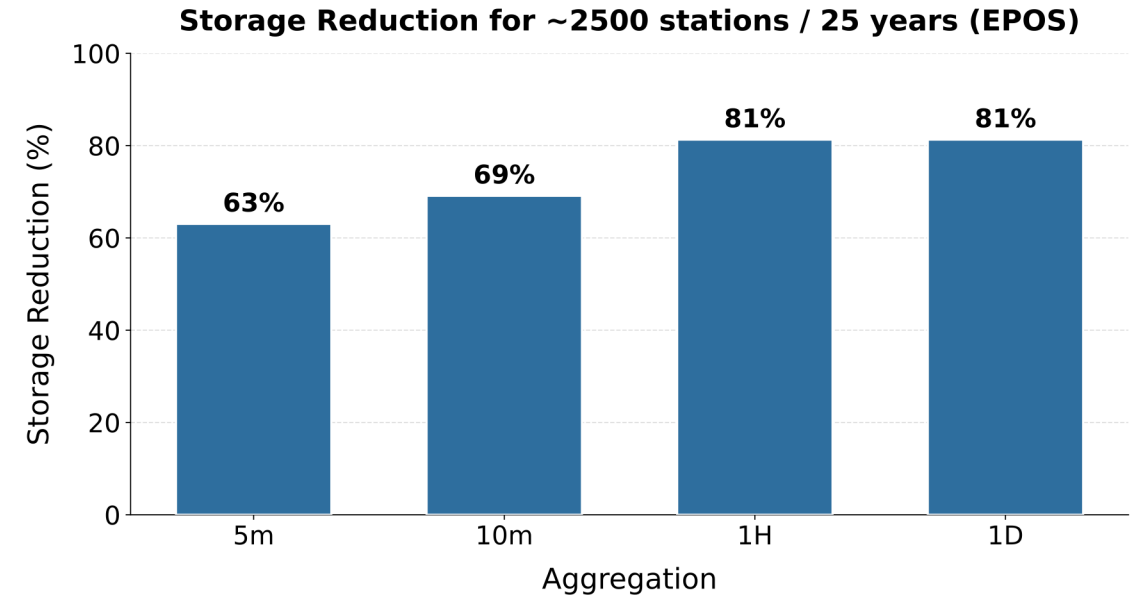
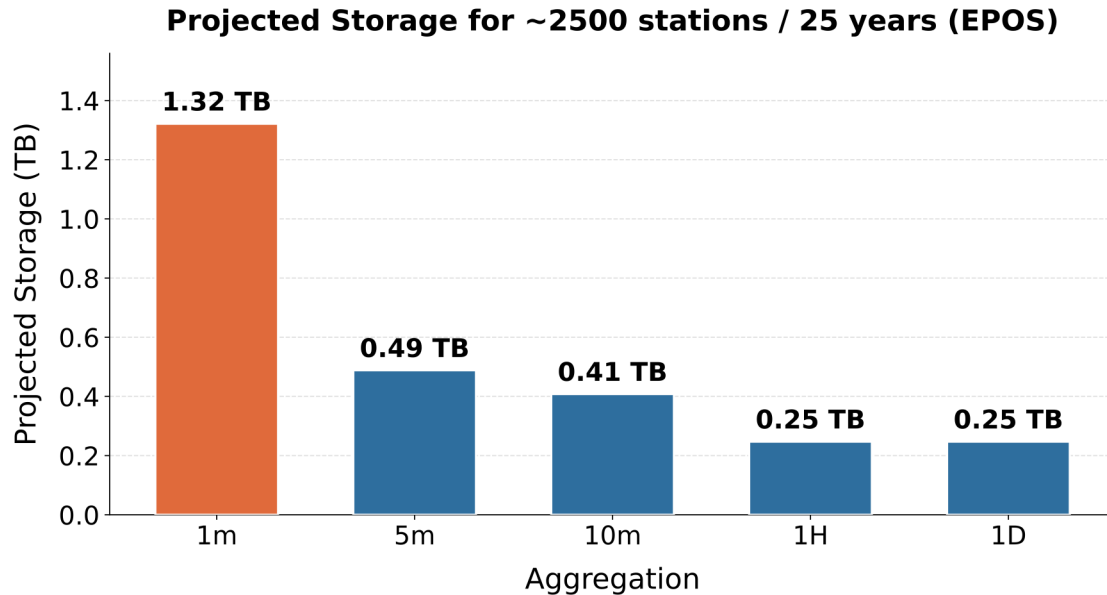
- repository comparison

Finer bins increase storage requirements but preserve more detailed availability information

Time-Bin & Aggregation (Storage)

Required Storage of Parquet Files – extrapolation based on the tested UBI repository

UNIT: YEAR -> MONTH -> SITE_DAYOFMONTH



1min time-bin occupies more space than the other time-bins

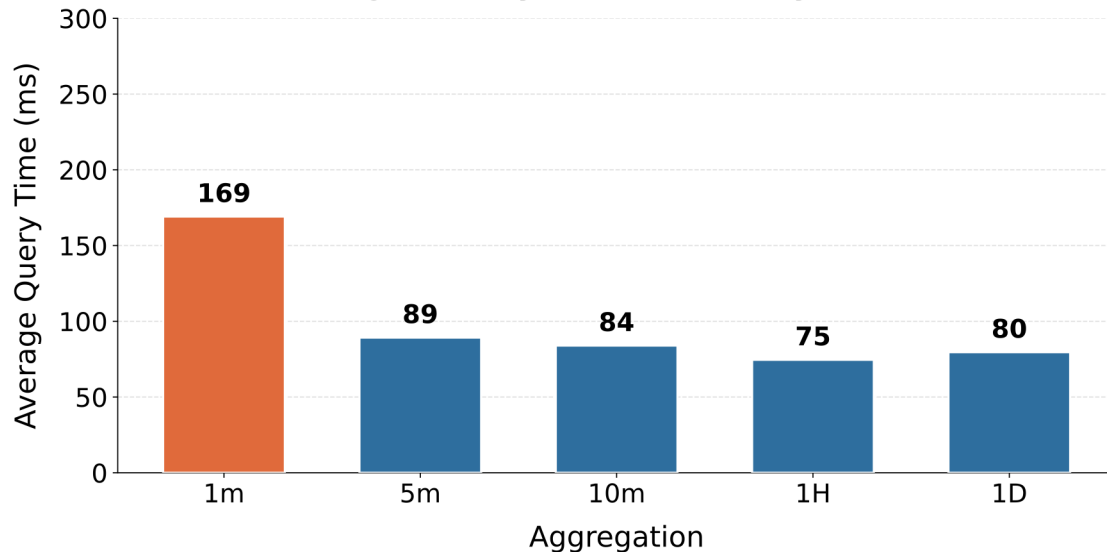
Time-Bin & Aggregation (Access)

Access Time to the Parquet Files – based on tested UBI repository

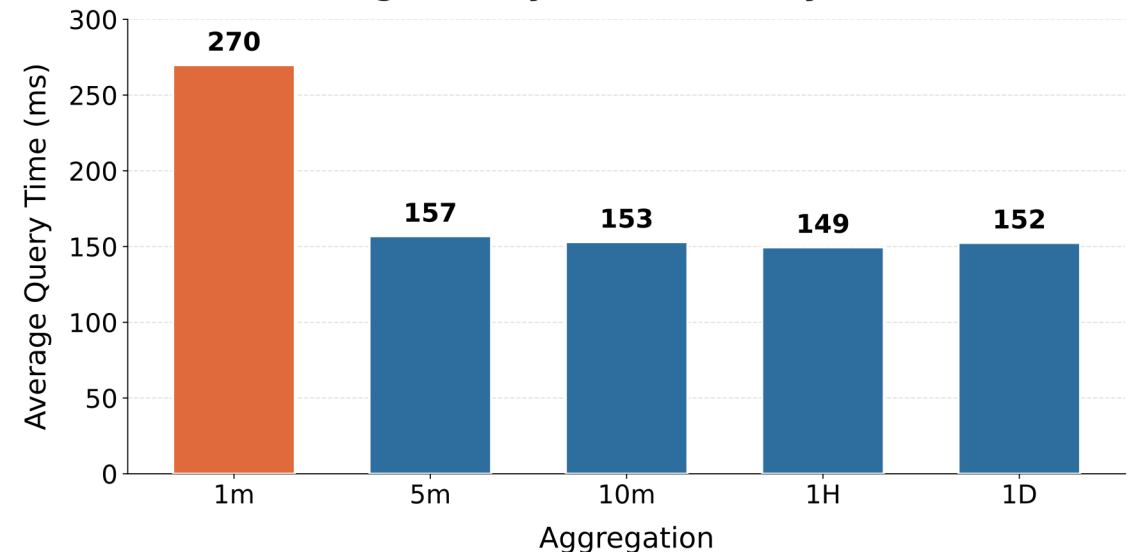
UNIT: YEAR -> MONTH -> SITE_DAYOFMONTH

Comparing access to one Parquet file or two Parquet files

Average Query Time — 1-Day Window



Average Query Time — 2-Day Window



1min time-bin requires more time than the other time-bins

Conclusions

- GNSS archives are moving from file-centric storage toward structured, queryable data systems.
- RINEX remains essential for exchange and legacy compatibility; the archive and API layer can evolve.
- IGS is an essential body to coordinate the efforts being carried out by many organizations.
- The critical federation layer is not only the observation format; it is standardized discovery and availability metadata.



Join us ...

email:
gnss.formats@segal.ubi.pt



There is still (some) room ...