

True-Ups Using the SIF Infrastructure

Executive Summary

Event-driven synchronization gets most SIF ecosystems “nearly” correct—until real-world operations introduce drift: missed publishes, edge cases, manual database edits, and other out-of-band changes. A **True-Up** is a **periodic reconciliation** that forces a consumer data store (often an **Operational Data Store / ODS**) back into alignment with the **source of truth** (typically a Student Information System / SIS) by retrieving the full record set for a domain, ingesting what is returned (add/update), and removing what is *not* returned (delete-by-absence). **This is explicitly not a “first-of-year initialization.”**

True-Ups are best understood as **data hygiene**: they don’t replace events; they close the last gap—moving fidelity from “~99%” toward “as close to 100% as practical.” A pragmatic cadence is **once or twice per week** for manageable datasets, because True-Ups are designed to catch edge cases rather than serve as the primary sync mechanism. The approach has been possible since **SIF 2** via request/response patterns, but **SIF 3** makes it more feasible due to improved performance characteristics.

Abstract

A True-Up is a bulk reconciliation pattern implemented with standard SIF request/response operations to ensure a consumer’s datastore accurately mirrors a provider’s current dataset, including the difficult case of determining what should no longer exist. This paper defines True-Ups, explains where they fit within SIF Infrastructure, provides an implementation blueprint (including optional hashing/flagging optimization), and documents operational pitfalls—especially performance impacts and the scalability limits of delete-by-absence for very large datasets

Glossary (topic-specific)

True-Up: A periodic reconciliation operation that retrieves the full dataset from a provider, ingests adds/updates, and removes consumer-side records not present in the provider response (delete-by-absence). Not a first-of-year init.

SIF Infrastructure: The integration mechanisms supporting eventing and request/response data exchange across SIF implementations; True-Ups use standard request/response (not a new protocol).

Provider (Source of Truth): The system authoritative for a domain (commonly the SIS) whose data should be mirrored downstream.

Consumer: The receiving system maintaining a synchronized datastore (commonly an ODS) that applies events and periodically True-Ups to correct drift.

ODS (Operational Data Store): A consumer-side datastore designed to maintain an operationally useful copy of source system data for reporting, analytics, or downstream interoperability.

Event-driven sync: Add/update/delete events published by the provider and consumed downstream; typically handles the majority of changes but can miss edge cases.

Data drift: Divergence between provider and consumer state caused by missed events, out-of-band edits, or edge-case behavior.

Delete-by-absence: Deletion inference method where records existing in the consumer but not returned in the provider's full extract are treated as deleted and removed.

Referential integrity (cross-object consistency): Consistency of references across objects (e.g., enrollments referencing existing students); True-Ups must consider processing order and/or eventual consistency strategies.

What a True-Up Is (and what it is not)

Definition

A True-Up is a **corrective synchronization pass** executed during normal operations (mid-year, mid-term, continuous operation) to reconcile a consumer datastore with the provider's authoritative dataset.

Not “first-of-year initialization”

A True-Up is **not** the annual “load everything at the start of the year” initialization. It exists because—even after an initial load—events can fail to represent reality perfectly across time.

Why True-Ups are necessary even with eventing

In an ideal world, events would be sufficient. In practice, gaps happen: direct database edits, edge cases where an update occurs without a publish, and other imperfections that prevent the consumer from matching the provider's true state.

Why It Matters (decision-maker framing)

The governance problem: trust in downstream data

If the ODS (or other consumer) is used for reporting, accountability, data products, or downstream exchange, drifting even slightly from the source can undermine confidence and create operational risk. True-Ups provide a systematic mechanism to restore trust by periodically proving alignment with the provider's truth.

What True-Ups deliver

- **Higher fidelity** than event-only pipelines by catching missed changes.
- **A verified picture of current reality**, including what should no longer exist in the consumer (the “negative space” problem).
- **Operational resilience**: a recovery pathway when assumptions about “perfect publishing” fail.

When you need True-Ups

You need True-Ups when providers are “less than perfect”—whether due to inherent system limitations or untracked manual processes. Practically, most real environments qualify, making True-Ups a strong baseline hygiene practice.

How often

Because True-Ups are intended to catch edge cases (not replace events), a typical cadence is **once or twice a week** for most datasets.

Where True-Ups Fit in the SIF Infrastructure

True-Ups are not a special SIF feature or a new concept: they are implemented using standard **request/response** retrieval patterns that have existed since **SIF 2**. What changes in modern implementations is feasibility: **SIF 3**'s improved performance characteristics make bulk retrieval and reconciliation more realistic at scale.

Key positioning:

- **Events** remain the primary mechanism for near-real-time change propagation.
- **True-Ups** are the periodic reconciliation backstop that confirms and corrects full-state alignment.

True-Up Workflow (high-level)

A True-Up can be described as a three-phase reconcile loop:

1. **Retrieve the full dataset**
 - Example: retrieve all *StudentPersonal* records from the SIS/provider.
2. **Ingest everything returned**
 - Create new records if absent in the ODS; update existing records if changed.
3. **Remove anything not returned**
 - Any ODS record not present in the provider's full response is treated as deleted (or otherwise no longer valid) and removed (or inactivated) in the consumer datastore.

This last step—ensuring not only what *is* present is in sync, but also what is *not* present—is the distinguishing value of True-Ups versus many “refresh” patterns.

Implementation Blueprint (developer guidance)

Scope selection

Start with domains where drift is high-impact and volumes are manageable:

- Students (StudentPersonal)
- Staff

- Enrollments (with sequencing considerations)
- Program participation objects, etc.

Efficient change detection (optional but recommended)

A practical optimization is to avoid field-by-field comparisons:

- **Pre-pass flagging:** mark all existing ODS records in-scope as “not yet seen in this True-Up.”
- **Hashing:** store a hash of canonicalized record content; compute hash for incoming records and compare to determine whether an update is necessary.
- **Seen clearing:** as each provider record is processed, clear the “not yet seen” marker and update the stored hash (if used).
- **Deletion sweep:** after processing all incoming records, any remaining “not yet seen” records are candidates for delete-by-absence.

Delete semantics

Delete-by-absence is straightforward for moderate datasets. It becomes problematic for very large datasets (e.g., attendance at extreme scale), where re-fetching everything just to infer deletions is not operationally realistic.

Practical mitigations (implementation/architecture choices):

- Prefer **soft-delete** or **deletion metadata** patterns where feasible (e.g., keeping deletion markers or complementary deletion logs) so deletes can be communicated without full re-extracts.
- Use True-Ups for large datasets only with **careful scoping** (partitioning, partial windows, or reduced cadence).

Referential integrity and processing order

True-Ups at the “document/object level” do not automatically solve cross-object integrity issues. If child objects reference parent objects, processing order matters:

- True-Up **Students before Enrollments** to avoid transient orphans and support consistent ingestion.
- If the source system itself violates referential integrity (e.g., enrollments referencing missing students), True-Ups can expose the problem but cannot fully correct provider-side integrity defects.
- Where ordering cannot be guaranteed, adopt an **eventual consistency** strategy (deferred validation/retry) rather than hard-failing ingestion.

Operational Considerations

Costs and resource impact

True-Ups are “heavyweight” because both sides pay:

- **Provider load:** bulk extraction and outbound transfer of large record sets.
- **Consumer load:** reconciliation compute, hashing/compare, and bulk writes/deletes.

Scheduling and orchestration

- Use **off-peak** windows.
- Stagger object domains if multiple True-Ups are required.
- If doing partial True-Ups (by school, district, year, etc.), maintain a clear ledger of what has been reconciled and when.

Monitoring and accountability

True-Ups should produce measurable outputs:

- count of adds/updates/deletes applied
- timing metrics (duration, throughput)
- anomaly detection (spikes may indicate eventing or provider issues)

Pitfalls and Limitations

1. Time and resource cost

True-Ups can stress both provider and consumer systems due to bulk processing demands.

2. Delete-by-absence does not scale infinitely

For extremely large datasets, inferring deletions by full extraction is not workable at high frequency; additional architectural patterns are required.

3. Operational complexity for partial True-Ups

If you partition scope, you must track which scopes were reconciled to avoid blind spots.

4. True-Ups can mask deeper interoperability defects

If True-Ups routinely correct large numbers of issues, the event pipeline or provider behavior likely needs improvement. Over-reliance can reduce pressure to fix systemic publishing problems.

Conclusion

True-Ups are a practical, standards-aligned reconciliation pattern that leverages SIF's existing request/response capabilities to keep consumers (like ODS systems) aligned with the provider's source of truth—including the critical requirement to remove records that no longer exist. They should be treated as **periodic data hygiene**, not a replacement for event-driven sync, and generally executed **once or twice weekly** to capture edge cases without imposing undue load. When implemented with disciplined scoping, optional hashing/flagging optimizations, correct object processing order, and realistic deletion strategies for large datasets, True-Ups materially increase data fidelity and confidence across interoperable ecosystems.