Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Recovering Failed Health-Interface Messages: A Synthetic No-PHI Workflow for Terminology Error Isolation and Warehouse Closure

Krishna Sai Pokala

ok50250@umbc.edu

Independent Researcher https://orcid.org/0009-0001-3012-4018

Method Article

Keywords: Health Information Technology, health informatics, healthcare interoperability, health-interface recovery, observation results, ORU-style messages, terminology validation, dead-letter queue, failed message recovery, idempotency, EHR data quality, synthetic data, no-PHI, warehouse verification, software reliability

Posted Date: June 16th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10027435/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Recovering Failed Health-Interface Messages:

A Synthetic No-PHI Workflow for Terminology Error Isolation and Warehouse Closure

Krishna Sai Pokala
Independent Researcher, Health Information Technology, United States
ok50250@umbc.edu
ORCID: 0009-0001-3012-4018

Preprint version: June 2026

Repository: https://github.com/KrishnaSaiPokala/pokala-healthops
Project site: https://krishnasaipokala.github.io/pokala-healthops/

Preprint note. This manuscript has not been certified by peer review. The project is a synthetic, no-PHI Health Information Technology engineering case study. It is not a clinical decision system, not a production electronic health record integration, not a certified Health IT module, not an HL7/FHIR conformance claim, and not a HIPAA certification claim.

Abstract

Background: Health-interface pipelines may fail when observation-result messages contain invalid, missing, local, or unmapped terminology codes. If these failures are handled as ordinary exceptions, records may be silently lost, retried without control, duplicated, or allowed to reach downstream analytic storage without adequate verification. **Objective:** To present a reproducible, synthetic, no-PHI workflow for isolating terminology-related health-interface failures, recovering remediated records, and verifying warehouse closure. **Methods:** The project models a synthetic observation-result pipeline with terminology validation, dead-letter isolation, mapping remediation, controlled record recovery, warehouse checks, duplicate-observation checks, recovery artifacts, and a public command-center review surface. Closure was defined with explicit engineering predicates: recovered records must equal isolated failures, open dead-letter records must be zero, duplicate observations must be zero, and all warehouse checks must pass. **Results:** In the reference incident, 500 inbound synthetic ORU-style messages were processed. The workflow isolated 218 terminology failures, recovered 218 remediated records, closed the dead-letter queue with 0 open records, completed 3 of 3 warehouse checks, and produced 0 duplicate observations. Repository validation included frontend type checking and static export, Python linting and type checking, automated tests, documentation build, and GitHub Pages deployment. **Conclusions:** The project demonstrates a compact recovery pattern for making failed synthetic health-interface messages visible, recoverable, and warehouse-verifiable. The contribution is not a production hospital interface or clinical model, but a bounded engineering artifact focused on the failure path rather than only the successful path.

Keywords: Health Information Technology; interoperability; observation results; terminology validation; dead-letter queue; idempotency; EHR data quality; synthetic data; no-PHI; warehouse verification; software reliability.

Key messages

- The project treats terminology failure as a recoverable health-interface incident, not as a discarded exception or uncontrolled retry.

- The reference incident processed 500 synthetic messages, isolated 218 terminology failures, recovered 218 records, closed the dead-letter queue, passed 3 of 3 warehouse checks, and created 0 duplicate observations.
- The paper's central technical object is a closure rule, not a dashboard: recovery requires count reconciliation, dead-letter closure, duplicate prevention, and warehouse verification.
- The system is synthetic and no-PHI by design. It is a reproducible engineering case study, not a clinical system, production EHR integration, or compliance product.

1 Introduction

Health Information Technology (HIT) systems depend on the reliable movement of electronic health information across clinical, operational, and analytic settings. Interoperability programs in the United States frame health data exchange around secure, standardized, and authorized exchange and use of electronic health information [1]. In practice, that national goal is realized through local implementation details: interfaces, message validators, terminology mappings, transformation layers, warehouses, dashboards, and operational procedures. A health-interface pipeline can therefore fail even when standards and policy frameworks are well defined.

Observation-result data are a useful target for studying this problem. Observations are central health information objects. In FHIR, the Observation resource can represent laboratory results, vital signs, imaging measurements, device measurements, clinical findings, assessments, and other point-in-time assertions [2]. These records often feed repositories, quality workflows, operational dashboards, and analytic systems. If an observation-result message arrives with an invalid or missing code, the issue is not simply a user-interface problem. It becomes a data-quality and recovery problem.

The difficult boundary is semantic rather than only syntactic. A message may be structurally present yet unsafe for downstream use if its observation code is local, unmapped, deprecated, malformed, or inconsistent with the expected terminology. LOINC provides a standard language for health measurements, observations, and documents, while SNOMED CT provides standardized clinical terminology for electronic exchange of clinical health information [3, 4]. Standards reduce ambiguity, but they do not remove implementation failure. Real receiving systems still need to decide what happens when an expected mapping is missing or a code cannot be trusted.

Data-quality literature makes this failure mode important. Electronic health record data and related operational data are not automatically fit for reuse simply because they exist electronically. Quality must be interpreted relative to intended use and may involve conformance, completeness, plausibility, provenance, and validation against known constraints [5–9]. In a health-interface pipeline, this means the receiving system should not simply accept every record into a warehouse. It should decide whether each record is usable, recoverable, or unsafe until remediated.

Health IT safety literature adds a second reason to focus on failure handling. Health IT can improve care, measurement, and coordination, but poorly designed or poorly monitored systems may introduce new kinds of risk [10, 14]. Sociotechnical models emphasize that HIT-related errors emerge through interactions among technology, workflow, people, organizations, and external rules, rather than through software code alone [11–13]. This manuscript does not claim patient-safety impact. It does, however, treat hidden interface failure, blind retry, and duplicate-generating recovery as operationally important failure modes that should be visible and bounded.

Many small HIT demonstrations emphasize successful-path functionality: a dashboard, a form, a FHIR viewer, or a static repository of data. Those artifacts can be useful, but they often do not show what happens when the interface fails. A more mature engineering artifact should make the failure path explicit: what failed, where it was isolated, what changed during remediation, what was recovered, whether downstream state was verified, and whether duplicates were avoided.

This paper presents a synthetic, no-PHI workflow for recovering failed health-interface messages caused by terminology errors. The workflow validates inbound observation-result messages, routes failures into a dead-letter queue, remediates the terminology condition, recovers the isolated records, verifies warehouse closure, checks for duplicate observations, and exposes the lifecycle through a public command-center review surface [23, 24]. The contribution is deliberately narrow: a compact, reproducible engineering case study for recovery-path design in Health Information Technology.

2 Related work

2.1 Health-interface exchange and observation data

HIT interoperability is often discussed at the level of national exchange, standards, policy, and certified technology. ONC describes interoperability as supporting secure and seamless electronic health information exchange among authorized users and as promoting standardized exchange and use of electronic health data [1]. Those policy goals depend on implementation details at the edge: local systems must validate, map, persist, and monitor exchanged records.

Observation-result workflows are especially relevant because observations are broad clinical data carriers. The HL7 FHIR Observation resource is used for vital signs, laboratory data, imaging measurements, device measurements, clinical findings, and assessment tools [2]. The present project does not claim formal FHIR conformance or production HL7 v2 conformance. Instead, it models a common reliability concern around synthetic ORU-style observation-result messages: records should be accepted into downstream storage only when their code semantics are safe enough for the intended use.

2.2 Terminology and semantic failure

Semantic interoperability depends on terminology systems as well as message formats. LOINC supports standardized naming and coding of health measurements, observations, and documents [3]. SNOMED CT is a designated standard for electronic exchange of clinical health information in U.S. federal systems [4]. In observation workflows, these terminologies often define the meaning of what was measured or asserted.

Standards do not make local implementations immune to failure. A local test code may lack a mapping, a mapping table may be incomplete, or a receiving system may receive a code it cannot interpret. HL7's Observation guidance notes that different code/value patterns can create challenges for semantic equivalence and safe interpretation of observations from different applications and users [2]. This project uses terminology failure as the reference failure mode because it sits at the boundary between syntactic message arrival and semantic data trust.

2.3 EHR data quality and fitness for use

EHR data-quality research provides the academic foundation for downstream verification. Weiskopf and Weng describe dimensions and methods of electronic health record data-quality assessment for reuse [5]. Kahn and colleagues propose a harmonized framework with categories such as conformance, completeness, and plausibility [6]. These concepts are directly relevant to warehouse closure: a recovered record should not merely be processed; its downstream representation should satisfy predefined checks.

Secondary-use literature also warns that operational health data require caution. Botsis and colleagues discuss data-quality issues and informatics opportunities in secondary EHR use [7]. Safran and colleagues outline the need for a national framework for secondary use of health data [8], and Hersh and colleagues describe caveats for using operational EHR data in comparative effectiveness research [9]. Although the present project is not a secondary-use research dataset, it borrows the same principle: downstream data should not be trusted solely because it exists. It should be checked against the intended use and production pathway.

2.4 Health IT safety and failure visibility

Health IT safety work argues that digital systems may create new hazards if they are poorly designed, configured, monitored, or integrated into workflows. The Institute of Medicine report on health IT and patient

safety emphasizes both safety benefits and risks introduced by health IT systems [10]. Ash and colleagues describe unintended consequences of health information technology, including errors related to patient-care information systems [14].

The present project does not study patient outcomes. It implements a safety-adjacent engineering pattern: failure states should be visible, bounded, and recoverable. Sittig and Singh propose a sociotechnical model for studying health IT in complex healthcare systems [11], and related work defines HIT-related errors and connects electronic records to patient-safety goals [12, 13]. In this manuscript, the command-center interface is not treated as a clinical operations console. It is a reviewer-facing surface that makes the recovery lifecycle inspectable.

2.5 Message recovery and idempotency

The engineering side of the project is grounded in reliable messaging patterns. Enterprise integration patterns describe message channels, routing, and failure handling as central concerns in distributed systems [18]. Helland emphasizes that distributed systems often need to tolerate boundaries where transactional guarantees do not extend cleanly [19]. Kleppmann discusses logs, retries, consistency, and the practical complexity of distributed data systems [20].

Controlled recovery is useful only when side effects are bounded. An idempotent consumer pattern addresses the possibility that a message may be delivered or processed more than once, while the transactional outbox pattern highlights the need to coordinate state changes and message publication [21, 22]. This project adapts those ideas to a synthetic health-interface setting. Failed records are isolated rather than dropped. Recovery is not treated as a blind retry. Closure requires count reconciliation, zero open dead-letter records, zero duplicate observations, and warehouse checks.

3 Scope and design contract

A major risk for a paper of this kind is overclaiming. The present artifact is strong only if its boundary is clear. Table 1 summarizes the design contract: what was implemented, what was reported with a bounded interpretation, and what is explicitly not claimed in the current artifact.

Table 1: Scope and claim boundary for the current artifact.

Implemented	Bounded interpretation	Not claimed
Synthetic ORU-style messages; terminology validation failure; dead-letter isolation; controlled recovery; warehouse closure checks; duplicate-observation check.	No-PHI synthetic data boundary; static command-center review surface; repository validation and deployment; single reference incident; engineering case-study scope.	Production EHR integration; formal HL7/FHIR conformance; concurrent recovery or load guarantees; MTTR or blast-radius telemetry; HIPAA or patient-safety outcome claim.

The implemented system includes a synthetic ORU-style message set, a terminology validation failure mode, dead-letter isolation and controlled recovery, warehouse closure checks, and duplicate-observation checks. The manuscript also reports the no-PHI design boundary, the public command-center review surface, repository validation, and deployment evidence. These are real parts of the artifact.

Several items are intentionally not claimed. The system does not implement production EHR integration, formal HL7/FHIR conformance testing, concurrent recovery against live traffic, MTTR or blast-radius

telemetry, HIPAA certification, or patient-safety outcome evaluation. These omissions are not hidden weaknesses. They are the edge of the present contribution.

Table 2: Implementation audit for the current artifact.

Capability	Current status	Interpretation
Operational boundary and no-PHI scope	Implemented	Synthetic data and no-PHI boundary are stated and reflected in the repository and public site.
Formal state sequence for isolation and recovery	Implemented	The reference incident follows a deterministic recovery sequence.
Warehouse closure predicates	Implemented	Closure predicates are defined and evaluated for the reference incident.
Concurrent recovery/race controls	Not implemented	Future work; the current paper must not claim production concurrency safety.
Programmatic blocking of downstream sync windows	Not implemented	The artifact gates technical closure, but does not model a production scheduler or warehouse sync blocker.
MTTR and blast-radius telemetry	Not implemented	The command center is a review surface; live operational telemetry is future work.

This audit changes the tone of the manuscript. The paper is not arguing that a single-person project substitutes for organizational Staff Engineer scope. It argues that a portfolio-scale artifact can still demonstrate senior-level engineering habits: defining failure modes, modeling state transitions, specifying closure predicates, validating repository quality, and refusing to overclaim.

4 System design

4.1 Design goal

The design goal was to build a reproducible synthetic workflow for failed health-interface message recovery. The workflow models the operational path that occurs after an observation-result message fails terminology validation. It is intentionally small enough to run as a public portfolio artifact, but structured enough to demonstrate recovery-path thinking.

The project has three hard boundaries. First, it uses synthetic messages only. Second, it does not connect to a real EHR, interface engine, laboratory information system, or patient database. Third, it does not claim certification, regulatory compliance, or clinical deployment. The point is to demonstrate a recoverable failure pattern in a transparent and shareable way.

4.2 Workflow components

The system has nine conceptual components. Synthetic inbound messages represent ORU-style observation-result events. A terminology validator checks whether observation codes can be mapped safely. Accepted records move toward downstream storage. Failed records are routed to a dead-letter queue rather than being discarded or forced into the warehouse. A remediation step corrects the mapping condition. A controlled recovery operation sends the failed records back through the pipeline. Warehouse checks verify downstream

closure. Duplicate checks verify that recovery did not create repeated observations. Recovery artifacts and a public command-center view make the lifecycle inspectable.

Table 3 presents the design as an event trace. The split after validation is the key design choice: accepted records and failed records are not treated the same. Failed records remain recoverable objects until remediation and controlled recovery satisfy closure criteria.

Table 3: Recovery event trace for the reference incident. The trace separates validation, recovery control, warehouse closure, and reviewer inspection while avoiding any claim of live-traffic concurrency control.

Step	Event	Control layer	Signal
1	Ingest synthetic observation-result messages	Interface validation	$N = 500$
2	Validate terminology mapping	Interface validation	accepted / failed split
3	Isolate failed records	Recovery control	$F = 218$
4	Remediate mapping and recover the isolated set	Recovery control	$R = 218$
5	Verify downstream warehouse state	Warehouse closure	$W/T = 3/3$
6	Export closure and inspect public review surface	Review surface	$O = 0, D = 0$

4.3 Synthetic data and privacy boundary

Synthetic data are appropriate because the goal is public reproducibility. No protected health information is processed, stored, redistributed, or displayed. The synthetic design prevents the manuscript from depending on restricted data access, institutional approvals, vendor contracts, or private patient records. It also makes the artifact suitable for repository-based review.

This boundary limits what the project can claim. Synthetic data cannot establish production throughput, real-world mapping coverage, clinical safety, vendor compatibility, or compliance. Those limitations are part of the artifact's honest scope. The claim is therefore an engineering claim: the workflow recovers a reference synthetic failure incident according to explicit closure criteria.

5 Recovery model and closure criteria

5.1 State model

The recovery path is modeled as a state sequence rather than an informal script. A terminology failure is detected; the failed messages are isolated; the mapping condition is remediated; the failed records are recovered; warehouse checks are executed; recovery artifacts are exported; and the incident is closed only after closure criteria are satisfied. This design makes the recovery path inspectable. A reviewer can ask where the incident currently sits, what evidence moves it to the next state, and what would prevent closure.

5.2 Closure variables and predicates

The closure criteria are expressed as simple engineering predicates. Let N be the number of inbound messages, F the number of failed messages isolated into the dead-letter queue, R the number of recovered messages, O the number of open dead-letter records after recovery, D the number of duplicate observations, W the

number of passed warehouse checks, and T the total number of expected warehouse checks. The incident is technically closed only when:

$$\text{Closed} \iff (R = F) \wedge (O = 0) \wedge (D = 0) \wedge (W = T). \tag{1}$$

For the reference incident, $N = 500$, $F = 218$, $R = 218$, $O = 0$, $D = 0$, $W = 3$, and $T = 3$. The notation is deliberately modest. It is not a mathematical theory of health information exchange. It is a compact way to specify what the workflow means by recovery.

Table 4: Closure predicate ledger for the reference incident.

Predicate	Expected	Observed	Failure prevented
Recovery reconciliation	$R = F$	218 = 218	Incomplete recovery of the isolated failure set.
DLQ closure	$O = 0$	0	Unresolved failed records after recovery.
Duplicate prevention	$D = 0$	0	Recovery side effects that create duplicate observations.
Warehouse verification	$W = T$	3 = 3	Unverified downstream state after remediation and recovery.

Table 4 is the technical contract of the workflow. The incident is not closed because a command finished; it is closed only when each predicate is satisfied.

5.3 Recovery artifacts

The manuscript uses the term recovery artifact in a concrete sense: a record, report, counter, check result, or interface view that helps a reviewer inspect the incident. It is not used as a vague synonym for trust. This follows provenance and reproducibility thinking: the path that produced a data state matters when judging reliability, not only the final value [15–17].

The recovery artifact set includes an incident summary, a DLQ report, a recovery report, warehouse check results, duplicate-observation counts, command-center views, and validation logs. Each artifact answers one operational question: what failed, what was preserved, what was recovered, what was verified, and what remains open.

6 Results

6.1 Reference incident outcome

The reference incident processed 500 inbound synthetic ORU-style messages. Of these, 282 were accepted initially and 218 failed terminology validation. The 218 failed records were isolated, remediated, and recovered. After recovery, the workflow reported 0 open dead-letter records, 3 of 3 warehouse checks passed, and 0 duplicate observations. Figure 1 presents the accounting path.

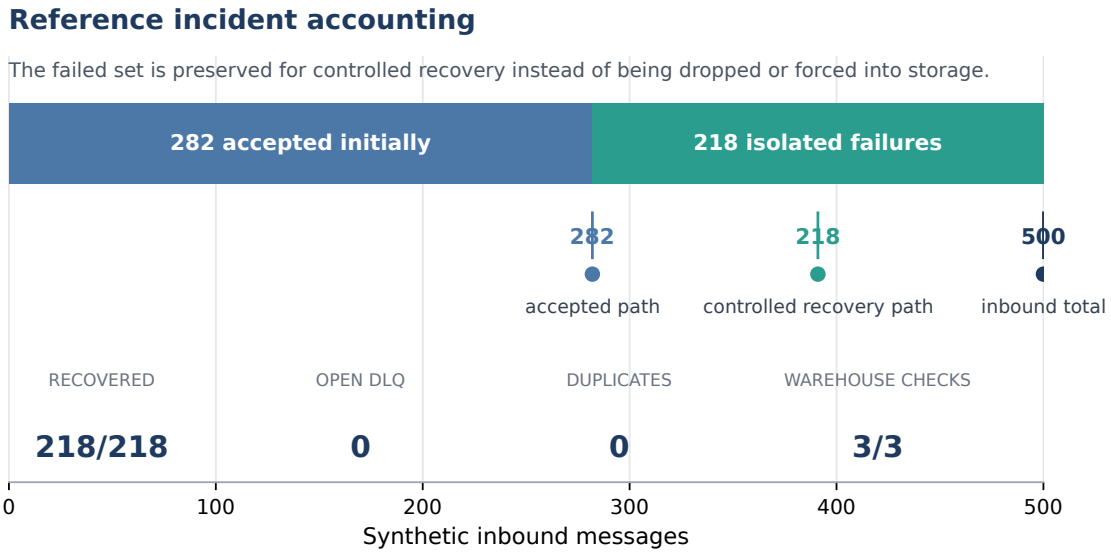


Figure 1: Reference incident accounting. The failure set is preserved and recovered instead of dropped. Closure is assessed by recovery reconciliation, open-DLQ count, duplicate count, and warehouse checks.

Table 5: Reference incident result.

Metric	Value	Interpretation
Inbound synthetic messages	500	Reference incident size.
Accepted initially	282	Messages accepted before remediation.
Terminology failures	218	Failure set isolated away from the warehouse path.
Recovered messages	218	All isolated failures were recovered after remediation.
Open DLQ after recovery	0	Closure condition satisfied.
Warehouse checks	3/3	Downstream verification passed.
Duplicate observations	0	Recovery side effects controlled.

6.2 Closure outcome

All closure predicates were satisfied. Recovery count matched failure count, the dead-letter queue closed to zero open records, duplicate observations remained zero, and all warehouse checks passed. This is the central result: the incident did not merely complete; it completed according to a predefined technical definition of closure. The predicate ledger in Table 4 is therefore the main result contract rather than a supporting decorative summary.

6.3 Review surface and validation evidence

The public site exposes the recovery lifecycle through an operator-style command-center interface with views for incident state, dead-letter queue state, recovery, and recovery artifacts [24]. This interface is not a live hospital operations console. Its purpose is to make the artifact legible to reviewers by showing the same

concepts that the workflow enforces: failure isolation, recovery accounting, warehouse verification, and closure.

The repository was also validated with frontend, backend, documentation, and deployment checks. These checks do not prove production readiness, but they support reproducibility and implementation discipline. Figure 2 summarizes the evidence stack.



Figure 2: Reproducibility evidence stack. The artifact is reported with synthetic data, behavioral tests, code-quality gates, build and documentation checks, deployment evidence, and a public review surface.

Table 6: Reproducibility validation checklist.

Check	Purpose	Result
Frontend typecheck	Verify TypeScript surface.	Passed
Frontend static build	Build public site artifact.	Passed
Python lint	Check backend code hygiene.	Passed
Python typecheck	Verify Python typing surface.	Passed
Automated tests	Validate pipeline behavior.	18 passed
Documentation build	Verify docs build strictly.	Passed
GitHub Pages deploy	Publish public artifact.	Passed

7 Discussion

The principal finding is that a compact synthetic HIT artifact can model failed-message recovery with explicit closure criteria. The project focuses on the recovery path rather than only the successful path. This matters because a health-interface system is not mature simply because good messages pass. It is more mature when bad messages fail safely, remain recoverable, and close with inspectable checks.

This makes the project meaningfully different from a dashboard-only HIT demonstration. A dashboard

can display final state without explaining whether upstream records were lost, duplicated, or corrected. The present workflow instead asks what happened when records failed. It isolates the failure set, applies remediation, recovers the failed records, checks warehouse state, prevents duplicate observations, and reports closure.

Terminology failure is an appropriate reference failure mode because it is both narrow and realistic. Observation-result workflows depend on code semantics. LOINC and SNOMED CT provide important terminology infrastructure, and FHIR Observation guidance shows that code/value representation can affect safe interpretation [2–4]. The project does not attempt to solve terminology management broadly. It demonstrates what a receiving pipeline can do when a terminology condition is not satisfied: isolate, remediate, recover, and verify.

The warehouse checks connect the workflow to EHR data-quality thinking. Kahn and Weiskopf/Weng emphasize that data quality is multidimensional and use-dependent [5, 6]. In this project, recovered records are not treated as ready for downstream use until downstream conditions pass. The result is a small but useful pattern: warehouse state is not trusted solely because a recovery command completed. It is trusted only after closure checks pass.

Side-effect control is the core engineering issue. A naive retry strategy may process failed records, but it can also create duplicates or leave unresolved failures. Reliable messaging literature and distributed-system patterns emphasize that message processing must tolerate retry and delivery uncertainty [18–20]. The closure predicates in Equation 1 translate that principle into the project: recovered records must equal failed records, open dead-letter records must be zero, duplicates must be zero, and warehouse checks must all pass.

The current command-center interface adds inspectability, not production telemetry. That distinction is important. The artifact reports incident state, DLQ count, recovery count, warehouse checks, duplicate count, and validation status. It does not yet report mean time to remediation, blast radius, or concurrency-race metrics; those belong in a future evaluation layer rather than in the current claim set.

The work should therefore be interpreted carefully. It is a no-PHI synthetic project. It does not connect to Epic, Cerner, a laboratory information system, a production HL7 interface engine, or a real FHIR server. It does not implement formal HL7 or FHIR conformance testing. It does not establish patient-safety benefit, clinical validity, compliance, or production scalability. Its value is narrower: it demonstrates a shareable health-interface recovery workflow with explicit closure rules.

That narrowness is also a strength. The manuscript does not need to claim a hospital-grade product. It shows a specific engineering pattern that can be reproduced and reviewed publicly. Future work can expand the artifact toward formal parsers, conformance fixtures, terminology-service mocks, multiple failure scenarios, concurrency testing, and stronger evidence bundles.

8 Limitations and future work

Several limitations are central to the interpretation of this work. First, the data are synthetic. This supports public reproducibility and avoids protected health information, but it does not establish performance in a real healthcare environment. Second, the evaluation uses a single reference incident. The result demonstrates one bounded workflow, not a broad benchmark. Third, the project does not integrate with a production EHR, interface engine, laboratory information system, or FHIR server. Fourth, it does not claim HIPAA compliance, certification, or clinical deployment readiness.

Fifth, the workflow does not include formal HL7 v2 conformance testing or FHIR conformance testing. It uses ORU-style language and observation-result concepts to model a realistic recovery path, but formal standards conformance would require additional fixtures, profiles, parsers, and validation tooling. Sixth, the public command-center interface is a static demonstration of the workflow, not a live operations console.

Seventh, the project has not yet been evaluated under load, concurrent recovery, recovery interruption, or multi-tenant scenarios. Eighth, the manuscript does not measure clinical outcomes or patient-safety effects.

Future work should proceed in layers. The first layer is standards realism: add formal HL7 v2 ORU fixtures, FHIR Observation export, and terminology-service mocks. The second layer is failure-scenario coverage: add missing mapping, deprecated code, warehouse outage, recovery interruption, and partial remediation cases. The third layer is robustness: add concurrency, idempotency, restart, and resume tests. The fourth layer is evidence strengthening: add checksums, signed artifact bundles, and one-command reproduction. The fifth layer is external evaluation: ask independent reviewers to clone the repository, run the synthetic incident, inspect the command center, and verify the closure artifacts.

9 Conclusion

This manuscript presents a synthetic no-PHI workflow for recovering failed health-interface messages caused by terminology errors. In the reference incident, the workflow processed 500 inbound messages, isolated 218 terminology failures, recovered 218 remediated records, closed the dead-letter queue, passed warehouse checks, and prevented duplicate observations. The central idea is simple: failed observation-result messages should remain visible and recoverable until explicit closure criteria are satisfied.

The project is not a clinical system, production EHR integration, or compliance product. It is a reproducible health-IT engineering artifact focused on the recovery path. By emphasizing terminology error isolation, recovery accounting, warehouse verification, duplicate prevention, and reviewer-facing inspectability, the project demonstrates a compact pattern for making synthetic health-interface failures visible, recoverable, and verifiable.

Ethics statement

This project uses synthetic data only. No protected health information, identifiable patient data, or real clinical records are processed, stored, redistributed, or displayed.

Data availability

The project uses synthetic data and aggregate recovery artifacts. The repository contains source code, documentation, and public-site source needed to inspect the implementation.

Code availability

Code and documentation are available at `https://github.com/KrishnaSaiPokala/pokala-healthops`. The public site is available at `https://krishnasaipokala.github.io/pokala-healthops/`.

Funding

No external funding was received.

Competing interests

The author declares no competing interests.

Author contributions

Krishna Sai Pokala conceived the project, designed the recovery workflow, developed the software, validated the implementation, prepared the public site, interpreted the results, and wrote the manuscript.

References

[1] Office of the National Coordinator for Health Information Technology. Interoperability. 2026. Available from: https://healthit.gov/interoperability/. Accessed June 13, 2026.

[2] HL7 International. FHIR Release 4: Observation Resource. FHIR v4.0.1. 2019. Available from: https://hl7.org/fhir/R4/observation.html.

[3] Regenstrief Institute. What LOINC Is. 2026. Available from: https://loinc.org/get-started/what-loinc-is/. Accessed June 13, 2026.

[4] National Library of Medicine. SNOMED CT. Last reviewed September 18, 2019. Available from: https://www.nlm.nih.gov/healthit/snomedct/index.html.

[5] Weiskopf NG, Weng C. Methods and dimensions of electronic health record data quality assessment: enabling reuse for clinical research. Journal of the American Medical Informatics Association. 2013;20(1):144–151. doi:10.1136/amiajnl-2011-000681.

[6] Kahn MG, Callahan TJ, Barnard J, et al. A harmonized data quality assessment terminology and framework for the secondary use of electronic health record data. eGEMs. 2016;4(1):1244. doi:10.13063/2327-9214.1244.

[7] Botsis T, Hartvigsen G, Chen F, Weng C. Secondary use of EHR: data quality issues and informatics opportunities. AMIA Joint Summits on Translational Science Proceedings. 2010;2010:1–5.

[8] Safran C, Bloomrosen M, Hammond WE, et al. Toward a national framework for the secondary use of health data: an American Medical Informatics Association white paper. Journal of the American Medical Informatics Association. 2007;14(1):1–9. doi:10.1197/jamia.M2273.

[9] Hersh WR, Weiner MG, Embi PJ, et al. Caveats for the use of operational electronic health record data in comparative effectiveness research. Medical Care. 2013;51(8 Suppl 3):S30–S37. doi:10.1097/MLR.0b013e31829b1dbd.

[10] Institute of Medicine. Health IT and Patient Safety: Building Safer Systems for Better Care. Washington, DC: National Academies Press; 2012. doi:10.17226/13269.

[11] Sittig DF, Singh H. A new sociotechnical model for studying health information technology in complex adaptive healthcare systems. Quality and Safety in Health Care. 2010;19(Suppl 3):i68–i74. doi:10.1136/qshc.2010.042085.

[12] Sittig DF, Singh H. Defining health information technology-related errors: new developments since To Err Is Human. Archives of Internal Medicine. 2011;171(14):1281–1284. doi:10.1001/archinternmed.2011.327.

[13] Sittig DF, Singh H. Electronic health records and national patient-safety goals. New England Journal of Medicine. 2012;367(19):1854–1860. doi:10.1056/NEJMsb1205420.

[14] Ash JS, Berg M, Coiera E. Some unintended consequences of information technology in health care: the nature of patient-care information system-related errors. Journal of the American Medical Informatics Association. 2004;11(2):104–112. doi:10.1197/jamia.M1471.

[15] World Wide Web Consortium. PROV Overview. 2013. Available from: https://www.w3.org/TR/prov-overview/.

[16] Wilkinson MD, Dumontier M, Aalbersberg IJ, et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. 2016;3:160018. doi:10.1038/sdata.2016.18.

[17] Talmon J, Ammenwerth E, Brender J, de Keizer N, Nykanen P, Rigby M. STARE-HI–Statement on reporting of evaluation studies in Health Informatics. International Journal of Medical Informatics. 2009;78(1):1–9. doi:10.1016/j.ijmedinf.2008.09.002.

[18] Hohpe G, Woolf B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Boston: Addison-Wesley; 2003.

[19] Helland P. Life beyond distributed transactions: an apostate's opinion. In: Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research; 2007.

[20] Kleppmann M. Designing Data-Intensive Applications. Sebastopol: O'Reilly Media; 2017.

[21] Richardson C. Pattern: Idempotent Consumer. 2026. Available from: https://microservices.io/patterns/communication-style/idempotent-consumer.html. Accessed June 13, 2026.

[22] Richardson C. Pattern: Transactional Outbox. 2026. Available from: https://microservices.io/patterns/data/transactional-outbox.html. Accessed June 13, 2026.

[23] Pokala KS. Pokala HealthOps repository. 2026. Available from: https://github.com/KrishnaSaiPokala/pokala-healthops.

[24] Pokala KS. Pokala HealthOps public site. 2026. Available from: https://krishnasaipokala.github.io/pokala-healthops/.