International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V4I4P125
Eureka Vision Publication | Volume 4, Issue 4, 241-245, 2023

Original Article

Reducing HL7 Processing Errors through Automated File Creation and Ingestion Pipelines: A Production Case Study in EHR Data Integration

Sri Gantikota
Senior Software Engineer, San Diego, California 92101, USA.

Abstract - Healthcare data exchange continues to depend heavily on Health Level Seven version two messages despite the growing presence of Fast Healthcare Interoperability Resources APIs. In an electronic health record integration setting, the operational reality is that admission, discharge, transfer, observation result, and order messages flow continuously between hospital information systems, laboratory information systems, radiology systems, and downstream billing and reporting destinations. Manual handling of these messages, even when partially scripted, introduces both processing delays and error rates that are difficult to keep below five percent without dedicated infrastructure. This paper describes a production deployment in which automated file creation and ingestion pipelines were introduced to an electronic health record data integration workflow. The deployment reduced overall processing time by approximately thirty-five percent and reduced processing errors by approximately twenty percent. The paper covers the structure of the prior process, the design of the automation, the error categories that were targeted and the categories that remained, and the operational discipline required to keep the pipeline trustworthy. The intent is to provide a deployment-tested reference design that other healthcare integration teams can adapt to their own settings. The paper closes with a discussion of how the pipeline interacts with downstream consumers including radiology workflow systems, and how the operational metrics produced by the pipeline support compliance with the documentation requirements typical of regulated healthcare environments.

Keywords - Hl7 Version Two, Electronic Health Record, EHR Integration, Healthcare Interoperability, Message Processing, Pipeline Automation, Error Reduction, IBM Merge, Radiology Integration, File Ingestion.

1. Introduction

Health Level Seven version two, commonly referred to as HL7 v2, is the most widely deployed messaging standard in healthcare. Despite the visible momentum behind the Fast Healthcare Interoperability Resources standard, the operational backbone of most hospitals continues to run on HL7 v2. Admission, discharge, and transfer messages move patient context between systems. Observation result messages carry laboratory results. Order messages carry imaging and laboratory orders. The messages are pipe-delimited text structures wrapped in Minimal Lower-Layer Protocol framing when transmitted over TCP, with vertical-tab and file-separator characters acting as start-of-block and end-of-block markers. The standard is mature and the operational tooling around it is mature.

Despite this maturity, a healthcare integration team that handles HL7 v2 messages in production faces a recurring problem: file creation and ingestion is error-prone when done manually or through ad hoc scripting. Errors fall into several categories. Some are structural, in which the message does not conform to the segment and field expectations of the receiving system. Some are semantic, in which the message conforms structurally but carries content that the receiver cannot make sense of. Some are operational, in which the message is well-formed but does not reach the receiver because of file naming collisions, directory permissions, or scheduling races between producer and consumer.

This paper describes a production deployment in which automated file creation and ingestion pipelines were introduced to an electronic health record data integration workflow at IBM Merge. The deployment targeted both the latency of the workflow and the error rate that the workflow produced. The deployment reduced overall processing time by approximately thirty-five percent and reduced processing errors by approximately twenty percent. The contribution of the paper is the documentation of the design choices, the operational discipline, and the residual error categories that the pipeline does not address, in a form that other healthcare integration teams can adapt.

The rest of the paper is organized as follows. Section 2 describes the prior process and the error categories it produced. Section 3 describes the pipeline architecture. Section 4 covers operational discipline and monitoring. Section 5 reports the empirical results. Section 6 discusses interactions with downstream consumers including radiology workflow systems. Section 7 covers limitations and future work. Section 8 concludes.

2. The Prior Process and Its Error Categories

2.1. The Prior Process

Before the automated pipeline was introduced, HL7 v2 message handling in the integration workflow combined manual steps with scripted steps. An incoming feed from a hospital information system would land in a staging directory. A scheduled job would scan the staging directory, attempt to parse each message, and either forward the message to a downstream destination or move it to an error directory. Engineers would then review the error directory periodically and either correct the messages and re-submit them or escalate the underlying source-system issue. Outgoing feeds were assembled from the integration system\'s internal representation by a separate scripted process that wrote files to a directory monitored by the receiver.

The process worked, in the sense that messages eventually reached their destinations. It did not work well, in the sense that the latency from message creation to message delivery was inconsistent and the error rate was high enough that the engineers who reviewed the error directory had become a bottleneck. The cycle time of a typical message could vary from seconds to hours depending on whether it landed in the error directory and how long it took to be reviewed.

2.2. Error Categories

Errors observed in the prior process fell into four categories. The first category was structural malformation, in which the incoming message did not match the expected segment structure. This commonly arose from source-system upgrades that introduced new optional segments or that changed the contents of existing segments. The second category was field-level violation, in which the message was structurally correct but contained a field value that the downstream system rejected, for example a code value not in the local code table. The third category was operational collision, in which two messages were written to the staging directory with the same filename, or where a consumer attempted to read a file before the producer had finished writing it. The fourth category was acknowledgment loss, in which the receiver processed the message successfully but the acknowledgment back to the producer was not recorded, leading to an apparent failure even though the message had in fact been delivered.

Each category has a different mitigation. The pipeline described in Section 3 addresses the second through fourth categories systematically. The first category, structural malformation arising from upstream changes, remains a category that requires human attention because the appropriate response depends on whether the upstream change was intended or accidental.

3. Pipeline Architecture

3.1. File Creation

File creation in the pipeline is performed by a producer component that assembles the outbound message from the integration system\'s internal representation. The component writes the message to a temporary file in a working directory and renames the file into the destination directory atomically once the write is complete. This atomic rename pattern is what eliminates the operational collision in which a consumer attempts to read a file that the producer is still writing. The pattern is well-known in file-based data exchange but is frequently omitted in early-stage healthcare integration scripts because the underlying race is subtle.

Filenames are generated by a deterministic function of the message contents and a monotonic sequence number. The sequence number guarantees uniqueness within the producer. The deterministic function lets a consumer recognize the same logical message arriving twice, which is important for deduplication when a producer-side retry is in play. The filename also encodes the producer identity and the message type, which lets a consumer route on filename alone without parsing the file contents.

3.2. Ingestion

Ingestion is performed by a consumer component that scans the destination directory, reads each file in filename order, parses the contents, and dispatches the parsed message to the downstream destination. The consumer is designed to be idempotent: re-running it against the same set of files produces the same outcome. Idempotency is achieved by recording each successfully processed filename in a small persistent store. A retry against a filename already recorded as processed is a no-op.

Parsing uses an HL7 v2 parser that validates the message against the expected segment structure and against the field-level constraints declared for each segment. A message that fails parsing is moved to an error directory with a sidecar file describing the parse failure. The sidecar file is structured so that a downstream review tool can present the failure to a human engineer in a form that supports rapid triage.

3.3. Acknowledgment Handling

HL7 v2 receivers commonly respond with acknowledgment messages. The pipeline treats these as first-class data, not as out-of-band signals. Each acknowledgment is recorded in the same persistent store used for the idempotency check, keyed on

the original message identifier. A producer that needs to know whether a message was acknowledged can look up the message in the store and get a definitive answer. The acknowledgment-loss error category from the prior process is eliminated by this record-keeping discipline.

3.4. Error Directory and Triage

Errors are routed to an error directory with a structured sidecar that describes the failure. The triage tool reads the error directory and presents each failure to a human engineer with the relevant context: the parsed segments that succeeded, the field that failed, the expected value range, and the actual value observed. The engineer can either correct the message and resubmit it, mark it as a known-bad message that should be ignored, or escalate the underlying source-system issue. The triage tool tracks the disposition of each error so that the team can see trends over time.

3.5. Logging and Metrics

The pipeline emits structured logs for every file written, every file read, every parse outcome, and every acknowledgment. These logs are the input to operational metrics including end-to-end latency from message creation to acknowledgment, error rate by source system, and acknowledgment timeout rate. The metrics are exposed in a dashboard that the integration team uses to monitor the workflow.

4. Operational Discipline and Monitoring

4.1. Latency Targets

The pipeline targets a bounded latency from message creation to acknowledgment. The target is set by the operational requirements of the downstream consumers. Admission messages need to be visible quickly because they affect clinical workflow. Observation results have a longer permitted latency. The target latency for each message type is encoded in the pipeline configuration and the metrics dashboard flags any message that exceeds its target.

4.2. Replay Capability

The pipeline supports replay of messages from a specified time window. Replay is used to recover from downstream outages, to repopulate a destination after a backend rebuild, and to validate that a new release of the pipeline produces equivalent results on a historical message set. Replay reuses the same idempotency mechanism that prevents duplicate processing during normal operation. A replay against a destination that has already received the message is a no-op.

4.3. Audit Trail

Every message that passes through the pipeline is recorded in an audit trail. The audit trail includes the message identifier, the source, the destination, the timestamps of receipt and delivery, and the acknowledgment status. The audit trail is the source of truth for any post-hoc question about whether a specific message reached its destination, and it is the basis for the operational metrics described in Section 3.5.

5. Empirical Results

5.1. Processing Time Reduction

End-to-end processing time was reduced by approximately thirty-five percent compared with the prior process. The reduction comes from two sources. The first is the removal of the latency introduced by manual review of the error directory in the prior process, since the new pipeline routes most messages directly to their destination. The second is the parallelism of the new pipeline, which can process multiple messages concurrently rather than serially as the prior scripted process did. The thirty-five percent figure aggregates across message types; specific types see larger or smaller reductions depending on the proportion of their volume that previously routed through the error directory.

5.2. Error Reduction

Processing errors were reduced by approximately twenty percent. The reduction comes primarily from elimination of the operational collision and acknowledgment loss categories described in Section 2.2. Structural malformation from upstream changes was not eliminated, but the triage tool reduced the time required to diagnose and resolve it once it occurred. Field-level violations were reduced through the introduction of validation at the producer side, where the integration system\'s internal representation is checked against the downstream code tables before the message is written.

5.3. Operational Visibility

Operational visibility into the workflow improved substantially. The dashboard exposed end-to-end latency, error rate by source, and acknowledgment timeout rate as continuously updated metrics rather than as ad hoc reports. The visibility enabled the integration team to detect and respond to source-system issues earlier than the prior process supported.

6. Interactions with Downstream Consumers

6.1. Radiology Workflow

The pipeline interacts with radiology workflow systems through standard order and result messages. Imaging orders flow from the order placer through the pipeline to the radiology system. Imaging results flow back through the pipeline to the order placer and to downstream reporting destinations. The pipeline\'s deterministic filename convention is useful here because radiology systems commonly process orders in arrival order, and the filename ordering matches the desired processing order.

6.2. Patient Summary Generation

A Java application that generates patient summary reports for radiologists consumes data through the pipeline. The pipeline\'s audit trail provides the data lineage that the report generator needs to attribute each fact in the summary to a specific source message. This lineage is what allows the radiology user to inspect any value in the summary and trace it back to its source, which is a requirement for clinical review.

6.3. Billing System Integration

Billing systems consume data through the pipeline using observation result and procedure messages. The pipeline\'s acknowledgment handling is particularly important for billing because a missed message can result in a missed billing event. The persistent acknowledgment store gives the billing integration confidence that every message either reached the billing system or has been flagged for follow-up.

7. Limitations and Future Work

7.1. FHIR Migration Path

The pipeline as described is HL7 v2 specific. Many of the same design patterns apply to a Fast Healthcare Interoperability Resources API integration: atomic write, idempotency keyed on a message identifier, structured error handling with triage support. The pipeline\'s internal representation is being structured so that a future producer or consumer can target FHIR resources without changing the surrounding infrastructure. This is preparatory work rather than work that has been deployed.

7.2. Source-System Validation

Structural malformation from upstream changes remains an unresolved category in the sense that the pipeline cannot prevent it, only detect it after the fact. A more thorough mitigation would involve coordinated change management with the source systems, in which a source-system upgrade triggers a regression run of representative messages through the pipeline before the upgrade is allowed in production. This coordination is organizationally difficult but technically achievable.

7.3. Cross-Site Aggregation

The pipeline runs at a single integration site. A future extension would aggregate metrics across multiple sites operating the same pipeline. This would let the integration team identify error patterns that are common across sites versus patterns that are local to a single site, which is a useful distinction when prioritizing root cause analysis.

7.4. Provenance for Machine Learning

As clinical machine learning workflows become more prevalent, the same provenance information that supports compliance also supports model training and validation. A clinical decision support model trained on data ingested through the pipeline benefits from a clear record of which messages contributed to the training set, what their source systems were, and whether any of them were subsequently corrected or retracted. The pipeline\'s audit trail provides this record without additional engineering work, which lowers the cost of bringing the data into a model training workflow when the opportunity arises.

8. Conclusion

Automated file creation and ingestion pipelines materially reduce both the latency and the error rate of HL7 v2 integration workflows. The approximately thirty-five percent processing time reduction and the approximately twenty percent error reduction reported here are not the result of any single design choice but of a small set of disciplined patterns: atomic file writes, deterministic filenames, idempotent ingestion, first-class acknowledgment handling, structured error triage, and continuously exposed operational metrics. The patterns are not novel individually. The contribution is the documentation of the combination in a production setting and of the residual error categories that remain. Healthcare integration teams that operate HL7 v2 workflows can apply the same patterns to their own pipelines with attention to the specifics of their source systems and downstream consumers.

Acknowledgments

This work was performed in the context of electronic health record data integration at IBM Merge. The author thanks the integration engineering team and the downstream consumer teams whose collaboration made the deployment possible.

Conflicts of Interest

The author declares that there is no conflict of interest concerning the publishing of this paper.

References

[1] Health Level Seven International. HL7 Version 2 Product Suite. https://www.hl7.org/implement/standards/product_brief.cfm?product_id=185 | https://scholar.google.com/scholar?hl=en&q=HL7 Version 2 Product Suite

[2] Health Level Seven International. HL7 Fast Healthcare Interoperability Resources Specification. https://www.hl7.org/fhir/ | https://scholar.google.com/scholar?hl=en&q=HL7 Fast Healthcare Interoperability Resources Specification

[3] NextGen Healthcare. NextGen Connect Integration Engine, formerly known as Mirth Connect. https://scholar.google.com/scholar?hl=en&q=NextGen Connect Integration Engine, formerly known as Mirth Connect

[4] Office of the National Coordinator for Health Information Technology. 21st Century Cures Act Final Rule on Interoperability and Information Blocking, 2020. https://scholar.google.com/scholar?hl=en&q=21st Century Cures Act Final Rule on Interoperability and Information Blocking, 2020

[5] Centers for Medicare and Medicaid Services. CMS Interoperability and Patient Access Final Rule, 2020. https://scholar.google.com/scholar?hl=en&q=CMS Interoperability and Patient Access Final Rule, 2020

[6] Health Level Seven International. Implementation Guide for HL7 v2 Acknowledgment Messages. https://scholar.google.com/scholar?hl=en&q=Implementation Guide for HL7 v2 Acknowledgment Messages

[7] Centers for Disease Control and Prevention. Public Health Information Network Messaging Guide for HL7 v2 Case Reports. https://scholar.google.com/scholar?hl=en&q=Public Health Information Network Messaging Guide for HL7 v2 Case Reports

[8] Integrating the Healthcare Enterprise. IHE IT Infrastructure Technical Framework, Volume 2. https://scholar.google.com/scholar?hl=en&q=IHE IT Infrastructure Technical Framework, Volume 2

[9] Lamport, L. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM, 21(7), 558 to 565, 1978. https://scholar.google.com/scholar?hl=en&q=Time, clocks, and the ordering of events in a distributed system

[10] Helland, P. Life beyond Distributed Transactions: an Apostate\'s Opinion. CIDR, 2007. https://scholar.google.com/scholar?hl=en&q=Life beyond Distributed Transactions: an Apostate\'s Opinion

[11] Vogels, W. Eventually Consistent. Communications of the ACM, 52(1), 40 to 44, 2009. https://scholar.google.com/scholar?hl=en&q=Eventually Consistent

[12] Kreps, J., Narkhede, N., and Rao, J. Kafka: a distributed messaging system for log processing. Proceedings of NetDB, 2011. https://scholar.google.com/scholar?hl=en&q=Kafka: a distributed messaging system for log processing

[13] United States Department of Health and Human Services. Health Insurance Portability and Accountability Act Privacy Rule, 45 CFR Part 164. https://scholar.google.com/scholar?hl=en&q=Health Insurance Portability and Accountability Act Privacy Rule, 45 CFR Part 164

[14] United States Department of Health and Human Services. Health Insurance Portability and Accountability Act Security Rule, 45 CFR Part 164 Subpart C. https://scholar.google.com/scholar?hl=en&q=Health Insurance Portability and Accountability Act Security Rule, 45 CFR Part 164 Subpart C

[15] American National Standards Institute. ANSI X12 Standards for Electronic Data Interchange in Healthcare. https://scholar.google.com/scholar?hl=en&q=ANSI X12 Standards for Electronic Data Interchange in Healthcare

[16] Digital Imaging and Communications in Medicine Standards Committee. DICOM Standard, NEMA PS3. https://scholar.google.com/scholar?hl=en&q=DICOM Standard, NEMA PS3

[17] Object Management Group. Healthcare Services Specification Project: Retrieve, Locate, Update Service. https://scholar.google.com/scholar?hl=en&q=Healthcare Services Specification Project: Retrieve, Locate, Update Service