Swift Securities Analyzer: A Deeper Look into Settlement Failures using Java Spring Boot and Python

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Rithwik Raman

Fall 2024

Rosanne Vrugtman, Department of Computer Science

Swift Securities Analyzer: A Deeper Look into Settlement Failures using Java Spring Boot and Python

CS4991 Capstone Report, 2024

Rithwik Raman
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
xak7jw@virginia.edu

ABSTRACT
Failures in securities settlements often occur because banks do not consistently adhere to Swift's recommended practices for using Securities View messaging. Analysis was needed to see how the banks are using these messages, which involved developing a Java Spring boot RESTful application to read in and parse daily securities messages. I then created a Python analyzer tool to read the data and visualize the findings. The project provided Swift's Securities View team with automated tools to analyze MT537 messages, identifying compliance gaps and patterns that contribute to settlement failures, ultimately enhancing transparency and adherence to guidelines while reducing delays and costs. This project can be expanded to using machine learning to predict and catch settlement failures before they happen.

1. INTRODUCTION
Settlement failures in the securities market are a pressing issue for financial institutions worldwide, as they pose risks to market stability and can cause increasing operational costs. Swift, a global leader in secure financial messaging, handles over 1.7 million securities messages daily. Despite stringent standards and practices, these settlements often fail due to non-compliance, miscommunication, and operational errors, resulting in delayed trade settlement and large monetary penalties.

To identify and mitigate settlement failures, it is crucial to understand the intricacies of Swift messages—specifically, the MT537 message used in handling penalties transactions. I analyzed how banks use these Swift MT537 messages and utilized a standalone Java Spring Boot RESTful application to ensure that Swift-recommended practices are being adhered to. This application parses, processes, and extracts key information from MT537 messages, and a separate Python analyzer extracts insights from the message data providing the Swift Securities team a deeper understanding of how financial institutions are using the MT537 message.

Enhancing the visibility into settlement practices empowers banks to adhere more closely to Swift’s guidelines, potentially reducing the frequency and impact of settlement failures. My solution also sets the stage for future machine learning ventures to predict and prevent failures from happening altogether.

2. RELATED WORKS
The analysis and prediction of settlement failures have garnered increasing attention among financial institutions. A key example of this is the collaboration between BNY Mellon, a multinational financial services company, and Google Cloud, to use artificial intelligence and machine learning to predict settlement failures in the US Treasury. According to

Reuters (2021), the technology uses historical data and advanced modeling to analyze various factors like supply/demand, trading velocity, and counterparty activity. It can predict up to 40% of certain settlements failures with an accuracy of 90%. This partnership highlights a key example of using AI and cutting-edge technology to solve complex financial problems.

To gain a better understanding of usage of the Swift MT537 message among financial institutions, Swift released documentation that specifies the exact structure of the message and all the recommended/optional fields. According to the MT537 scope, the MT537 message is sent by an account servicer to an account owner to provide updates on the applicable penalties for late securities settlements (Swift, 2024). The documentation contains the specific syntactical formats for all the fields and includes various specifications for different usage scenarios. The complete Swift message is written in the "MT" format, which is Swift's proprietary messaging schema. Understanding the complexity of these messages is a key step in my development of the Java application that parses and extracts information from these messages.

3. PROJECT DESIGN

The development of the securities analyzer application was a result of the Swift Securities View team need to gain insights into how banks are using MT537 messages—almost two million of these are sent daily. This section provides a detailed overview of the system architecture, the client needs, and the challenges encountered during the development process.

3.1 System Architecture

The overall system architecture for the Swift Securities Analyzer consists of two key components: a Java Spring Boot REST API and a Python analytics tool. The Java Spring Boot application handles message ingestion, parsing, and data extraction, while the Python tool focuses on processing the extracted data and generating visual insights.

The first step in the process is for the Java Spring Boot REST API controller to receive the raw MT537 messages in their original "MT" format, which is the Swift-proprietary message format. Once the message is received, it undergoes conversion from MT to MX format, which is XML-based and easier to work with programmatically. This conversion is performed using an in-house tool developed specifically for transforming MT messages into XML.

An XPath expression is applied to extract relevant fields. XPath allows for precise navigation and parsing through the XML structure, enabling the extraction of key information such as penalties, transaction details and settlement data. This extracted information is then compiled into a JSON object, which serves as the final output of the Java application. Each JSON object is then appended to log files for further analysis.

The next component is the Python analytics tool, which reads the log files containing all the individual JSON outputs generated by the Java application. The primary role of this tool is to check for the occurrence of Swift-recommended fields within each message. By analyzing the extracted data, the tool assesses how closely banks adhere to Swift's guidelines. Finally, the Python tool produces a visualization that shows the percentage of recommended fields being used by each institution. This visualization is crucial for providing Swift's Securities View team with actionable insights to inform future decision-making.

3.2 Client Needs

The primary client for this project is Swift's in-house Securities View team, as they needed a system that would analyze the usage of the MT537 message. The end product is the Python-based visualization that provides a clear indication of which recommended message fields are repeatedly being under-utilized.

3.3 System Limitations

One of the primary limitations encountered during the development process was the lack of access to production data. Due to the sensitive nature of financial messaging, no real-world MT537 messages were available for testing purposes. As a result, all test cases had to be hand-crafted, which required extensive research into the structure and syntax of the MT537 message format. This created an additional layer of complexity, as I needed to ensure that the test cases covered a wide range of potential scenarios and edge cases.

Another challenge was the intricacy of Swift's messaging system itself. The MT537 message format is highly structured and includes a wide array of mandatory, optional, and conditional fields. This complexity meant that even small mistakes in the parsing logic could lead to incorrect data extraction or missed fields.

3.4 Key Components

This section provides a breakdown of the Swift Securities Analyzer's main functionalities, specifications, challenges, and solutions; these components highlight the system's architecture, designed to provide meaningful insights despite limitations to data access.

3.4.1 Specifications

The project specifications were loosely defined, with the main goal simply to provide insights into customer adherence to Swift's recommended practices. The only specification was that the analyzer must be a Java-based application, as the bulk of the team's existing backend functionality is written in Java. The key deliverable was the visualization, which clearly showed how the banks were using the MT537 message fields in their transactions.

3.4.2 Challenges

The most significant challenge was the lack of production data for testing, which made it tedious to validate the accuracy of the solution against real-world scenarios. In addition, there was a steep learning curve associated with learning the ins and outs of Swift's messaging format when testing functionality of the MT-XML conversion and the XML field extraction steps.

3.4.3 Solutions

To address the lack of production data, I created a series of test cases designed to mimic real-world MT537 messages. By studying Swift's documentation, I was able to construct a variety of messages that included both common and edge-case scenarios, ensuring that the Java application would handle a wide range of inputs.

Finally, containerizing the entire system using Docker significantly streamlined the packaging and deployment process. By encapsulating all the components within a single Docker container, I was able to efficiently ship my application to the Securities View team for testing and review.

4. ANTICIPATED RESULTS

The anticipated results revolve around providing the Swift Securities View team with valuable insights into how financial institutions are using MT537 messages. By analyzing the data, this project delivered a clearer picture of which institutions adhere to Swift's recommended practices, and identified patterns that could lead to settlement failures. The expected outcomes include increased transparency into the settlement process and improved adherence to Swift guidelines by banks.

The Java Spring Boot application, along with the Python analyzer, significantly reduce the manual effort required to review and analyze the MT537 messages. By automating the extraction of key fields and visualizing their usage, the system enables Swift's Securities View team to quickly identify problematic trends or compliance gaps. For example, the team will be able to see if certain banks consistently omit critical fields, allowing them to address the issues more proactively. This will also help in reducing the frequency of settlement failures, which could avoid penalties and delays, saving financial institutions both time and money.

5. CONCLUSION

The Swift Securities Analyzer project is important because it offers a streamlined and automated way to assess how banks are adhering to Swift's MT537 message guidelines for securities settlements. With the global impact of settlement failures, this tool provides much-needed transparency, allowing Swift's Securities View team to monitor message usage closely and identify any gaps that could lead to settlement issues. By parsing, analyzing and visualizing MT537 message data, the analyzer empowers Swift and its clients with actionable insights, ultimately promoting more consistent compliance and reducing operational risk.

By integrating Java Spring Boot for data parsing and Python for visualization, this project brings together the best of both languages in a neat and robust solution. As Swift and other financial institutions continue to automate compliance and improve message accuracy, the analyzer's insights will be increasingly valuable, supporting operational efficiency, regulatory adherence and reduced financial penalties in the securities market.

6. FUTURE WORK

This project can be expanded to utilizing real-time data to refine its accuracy and ensure full compatibility with a wide range of MT537 message scenarios. With access to actual data, the system could further identify more nuanced patterns of MT537 messages, which would in turn enhance the predictive value of its analysis. Additionally, machine learning algorithms could be used to proactively detect settlement failures before they actually occur, allowing financial institutions to address potential issues before they escalate.

This analyzer could also be adapted to other Swift message types outside of MT537 messages. Further product enhancements could involve more sophisticated data visualizations or a user-friendly dashboard for the Swift team, allowing them to monitor trends in real-time. Such upgrades could extend the value of the analyzer to additional departments within Swift, improving transparency and efficiency across different financial operations.

7. ACKNOWLEDGMENTS

I would like to acknowledge my mentors, Venkata Mylavarpu and Sumanth Dachepally, and the rest of the Swift Securities View team for their invaluable guidance and support throughout the project.

REFERENCES

Reuters. (2021, February 4). *BNY Mellon, Google Cloud technology to predict Treasury settlement failures*. Reuters. https://www.reuters.com/article/technology/bny-mellon-google-cloud-technology-to-predict-treasury-settlement-failures-idUSKBN2A42GO/

Swift ISO 20022. (2024).| *ISO20022*. ISO20022. https://www.iso20022.org/15022/uhb/finmt537.htm