Whitepaper

Enhancing Data Imports and Storefront Publishing in Spryker Cloud Commerce OS

Xiphias, June 2024



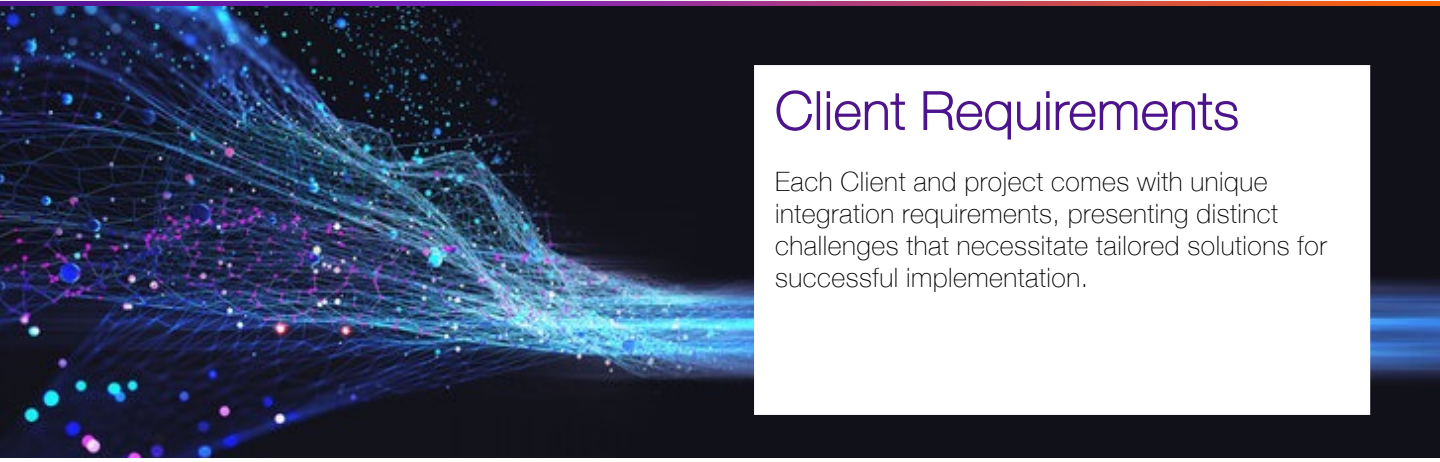
Introduction

Spryker Cloud Commerce OS is a robust and highly scalable e-commerce platform favored for its flexibility. However, efficiently processing large datasets presents significant challenges, particularly for enterprise-level businesses.

This whitepaper examines the data processing limitations within Spryker Cloud Commerce OS. Additionally, it showcases the substantial performance and reliability improvements achieved for leading Dehner cross-channel retailer through the implementation of our enhanced data processing methods.

Contents

1. Dehner Requirements **P03**
2. Challenges in Data Processing **P04**
3. Overview of Improved Mechanism: Data Importer **P05**
4. Overview of Improved Mechanism: Data Publisher **P06**
5. Comparative Analysis **P07**
6. Conclusion **P08**



Client Requirements

Each Client and project comes with unique integration requirements, presenting distinct challenges that necessitate tailored solutions for successful implementation.

Dehner Overview

Dehner is a leading German cross-channel retailer in the "Green Industry," with over 130 brick-and-mortar stores and an extensive online presence.

Spryker Cloud Commerce OS was selected as e-commerce platform to support online business which is currently operating four online stores with approximately 45.000 SKUs per store.

Given the scale of Dehner's operations and ambitions for further growth, we needed to develop a solution that addresses their specific integration requirements, ensuring seamless scalability and uninterrupted business operations.

Dehner Integration Requirements

API Integration

- Glue API Utilization: All product Price and Stock information must be integrated via the Glue API and data is provided from SAP
- CSV: Product information is provided in CSV file

Search Engine

- Fact Finder: Fact Finder will be used as the search engine instead of Elastic Search

Data Volume

- Large Dataset Handling: The system must efficiently process up to 6 million records for Price and Stock
- Performance: High-throughput, processing capabilities and synchronization without disruptions that could impact business.
- Data Integrity: Data consistency and integrity must be ensured
- Archiving and Cleanup: Archiving and cleaning up old data for maintaining system performance and managing storage efficiently.

Addressing the specific needs of Dehners large-scale operations and ambitious growth goals required careful and thorough analysis, including test of Spryker Standard Implementation before determining the exact implementation approach.



Challenges in Data Processing

Efficient data processing is crucial for maintaining the integrity and performance of an e-commerce platform. This section outlines the significant challenges faced when processing large datasets within Spryker Cloud Commerce OS.

Spryker Cloud Commerce OS Performance Degradation

Handling large datasets (50k+ records) using the Spryker Standard Implementation leads to substantial performance degradation. This issue was notably observed in the project when we first run the integration performance tests using Spryker Standard Implemention.

Key challenges and observations:

- Extended Import Times: Import times increase significantly as the number of records grows, primarily due to row-by-row processing
- Processing Bottlenecks: The high volume of events created related to the import, sometimes duplicated, acts as a bottleneck, slowing down the entire system. This issue not only impacts import processes but also prevents timely processing of other events
- Rabbit MQ & Jenkins Stability: Resource-intensive operations have led to occasional crashes of Rabbit MQ and Jenkins.
- Publishing Delays: There are delays in updating product prices and stock availability on the storefront, resulting in discrepancies where customers may see outdated information.

Metric	Observed Value
Number of Records	**758k**
Import and Processing Time	**60min**
Publishing Time to Storefront	**120min**
Total Time from Import to Storefront	**180min**

Business Impact

The repercussions of observed challenges and inefficient data processing extend beyond technical performance and resource wastage, affecting business operations and customer satisfaction and has following impact:

Customer Dissatisfaction

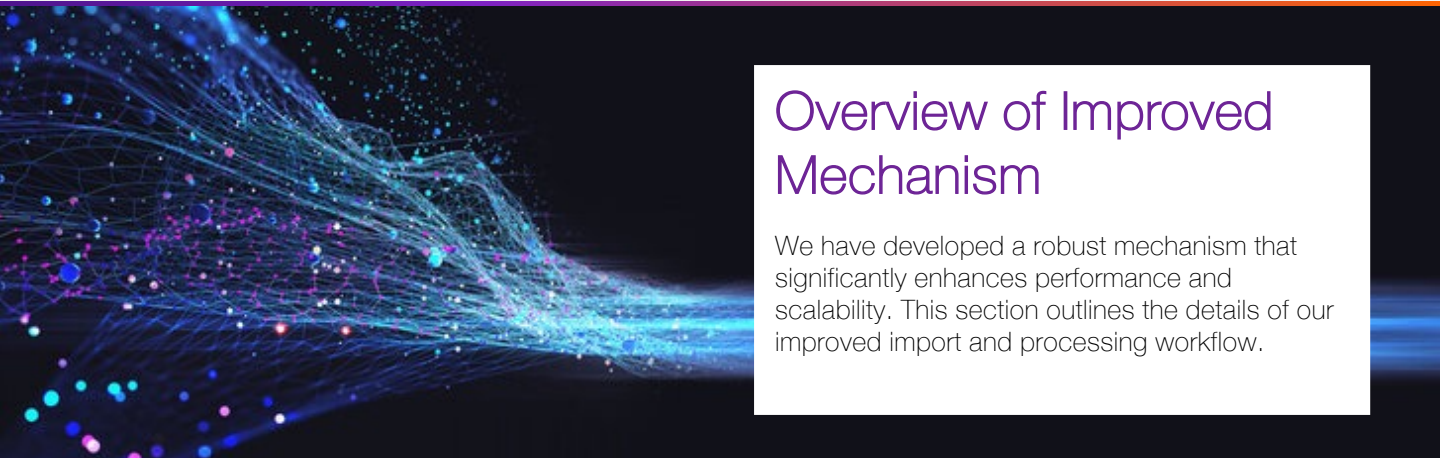
Inaccurate information, such as incorrect product prices or stock availability, almost certainly lead to poor customer experiences and frustration, consequently leading to reduced customer satisfaction and trust in brand.

Reputation Damage

Persistent and repeating issues with data accuracy and timeliness can have negative impact on the brand, making it difficult to retain existing customers and attract new ones.

Sales Impact

Incorrect product prices and availability can lead to missed sales opportunities, negatively impacting revenue. For instance, incorrect stock information might prevent potential customers from making purchases.



Overview of Improved Mechanism

We have developed a robust mechanism that significantly enhances performance and scalability. This section outlines the details of our improved import and processing workflow.

Improved Mechanism

To address Dehner's Integration Requirements and overcome the limitations of the Spryker Standard Implementation, we have decided to cover the end-to-end process using two key components: the Data Importer and Data Publisher. These components consist of smaller steps to enhance error handling and processing efficiency.

This approach allows us to refine and enhance performance while effectively implementing project-specific adjustments in the future, catering not only to this specific Client needs but also to any Client with similar requirements.

Data Importer: Price and Stock Overview

Receive data from SAP via Glue API: data is received and written into import table

Batch Size Parameter: parameter for defining chunk size, in our case the chunk size is defined to 50k records

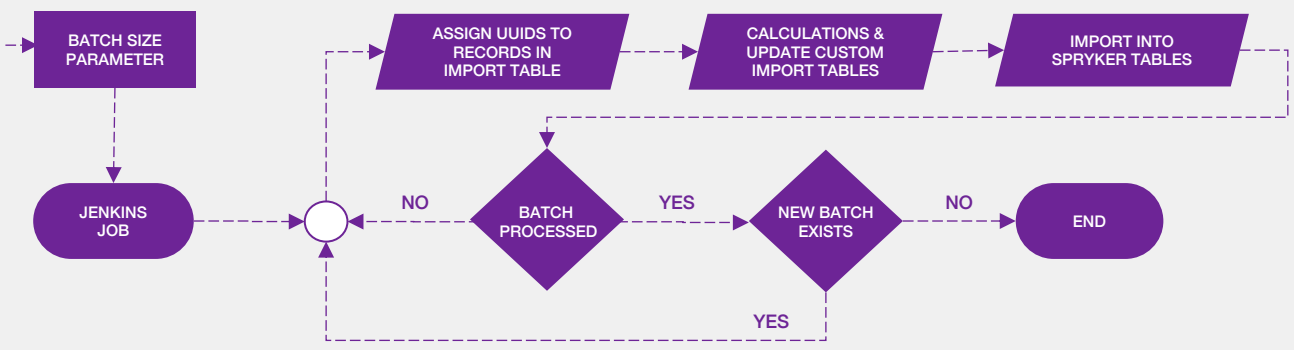
Jenkins Job execution: The Jenkins job executes a command to start the process. All subsequent steps are executed in a loop until all records in the import table are processed. In case of an error, the loop exits, and the error message is displayed in the Jenkins output console.

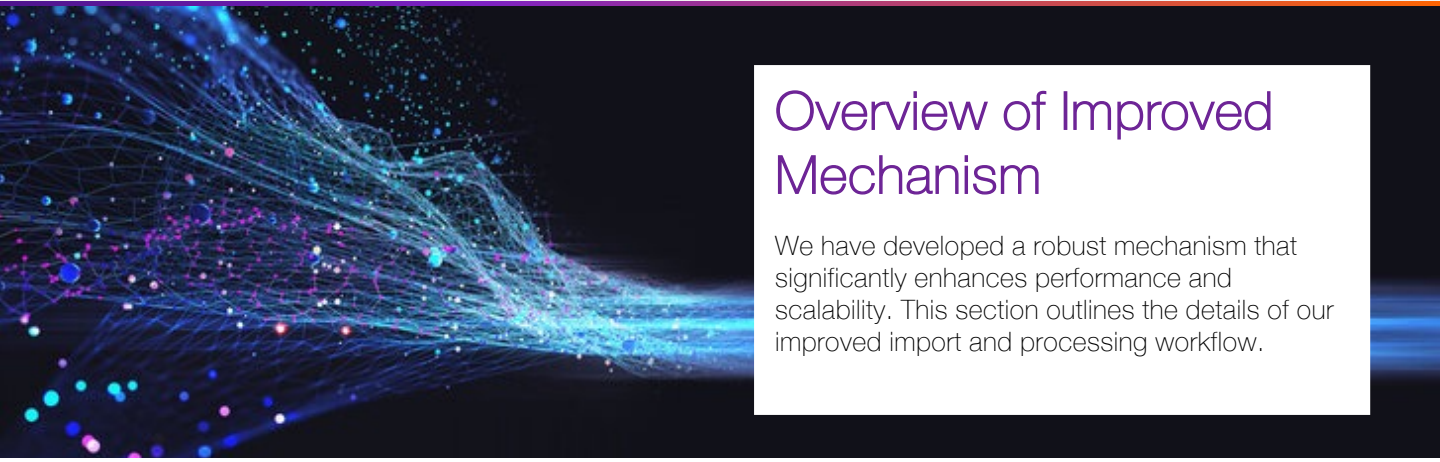
Mark the records for processing: Assign UUIDs to records in the import table, based on the chunk size. All subsequent steps and logic are applied only to that specific record set, i.e., chunk.

Execute Calculations: This step prepares the data and updates the import table with information such as availability, offer references, foreign keys, etc. This allows us to leverage only import table in following steps reducing the probability of database deadlocks on Spryker tables.

Import Data into Spryker Tables: The enriched data is imported into Spryker tables. All the necessary data is taken from the import table, ensuring no additional calculations are required during this step.

Data Importer: Price and Stock Process SimplifiedFlow Chart Overview





Overview of Improved Mechanism

We have developed a robust mechanism that significantly enhances performance and scalability. This section outlines the details of our improved import and processing workflow.

Data Publisher: Price and Stock Overview

We have implemented a switch allowing for two different use cases to optimize data publishing based on the size of data sets:

- Full Import or Mass Data Imports: Utilizes our Improved Mechanism to efficiently process large amounts of data in a timely manner.
- Delta Imports: Continues to use the Spryker Standard Implementation for smaller, incremental imports.

Batch Size Parameter: parameter for defining chunk size, in our case the chunk size is defined to 50k records

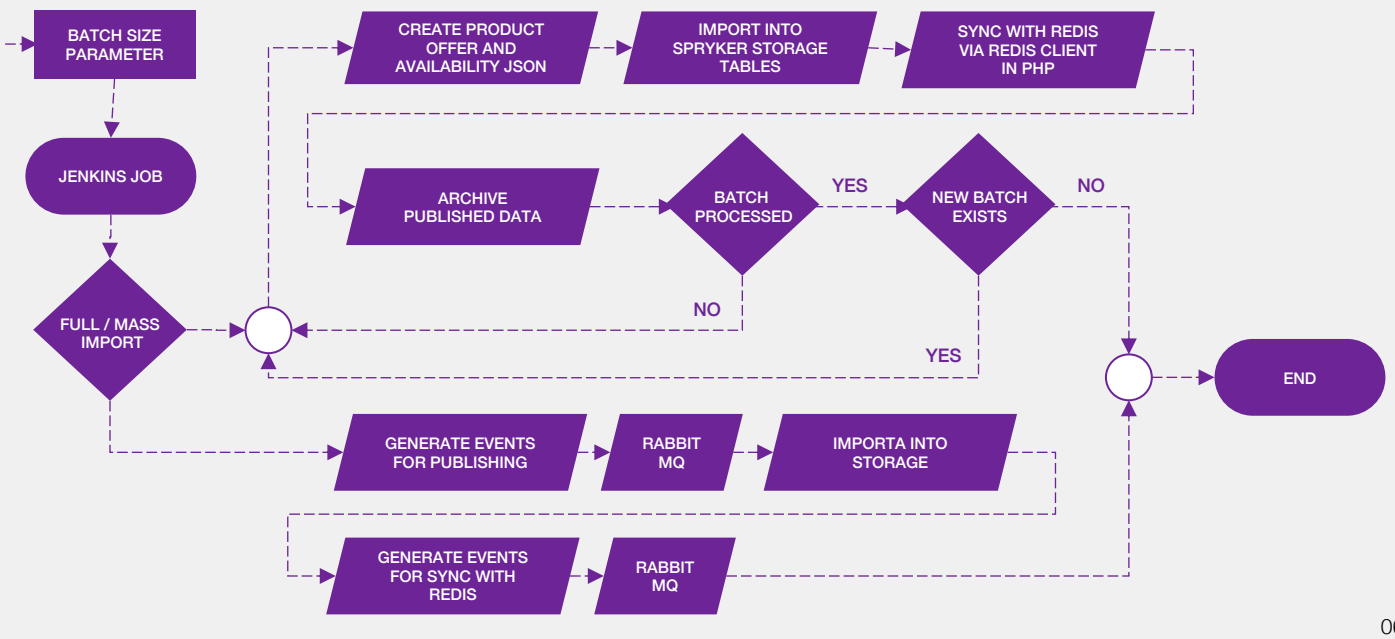
Build JSON Payload: JSON payloads for product price and stock are built using SQL

Import JSON Payload into Spryker Tables: JSON Payloads are saved into Spryker storage tables. This approach allows us to avoid RabbitMQ, thereby reducing the creation of excess events and long processing times.

Synchronization to Redis: Data is fetched from the Spryker storage tables, formatted, and sent directly to Redis using a Redis Client in PHP. This bypasses the need for message queuing via Rabbit MQ.

Archive Import Table: Processed data is removed from the custom import table and archived. This keeps the import table small, thereby enhancing performance and preventing bottlenecks.

Data Publisher: Price and Stock Process Simplified Flow Chart Overview



A Comparative Analysis

Our innovative implementation has demonstrated a substantial enhancement in processing times. This acceleration is not only important for maintaining timely product updates but also ensures uniformity across the storefront.

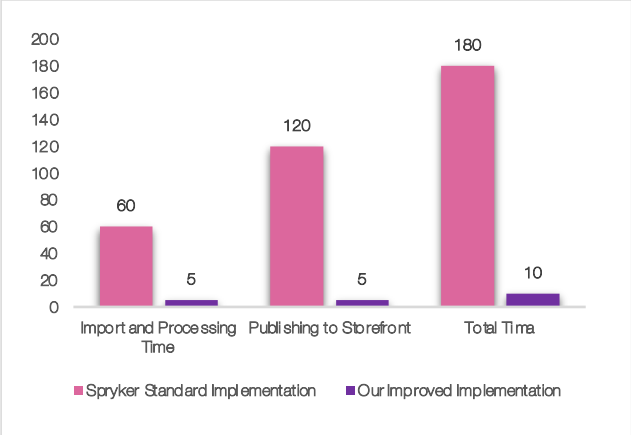
Comparative Analysis

In test case, we evaluated the system's performance with a substantial data set of 758k records. The results were impressive.

If we extrapolate from our test case, to achieve Dehner's goal of importing 6 million records, using the Spryker Standard Implementation it would approximately take a full day (24 hours). However, with our Improved Implementation, the same task could be accomplished in a mere 80 minutes.

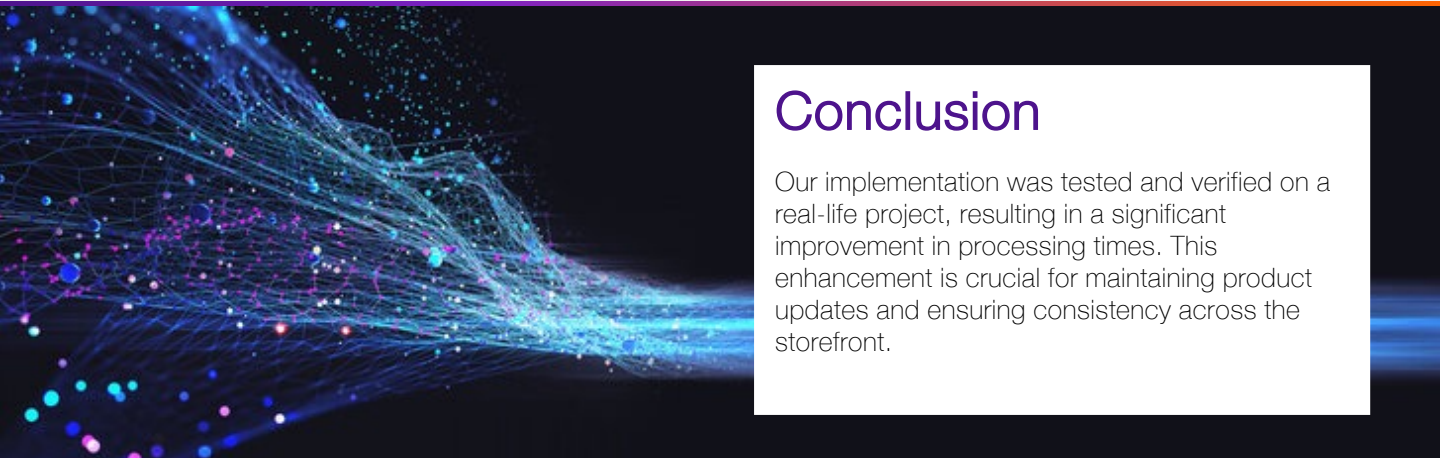
This contrast underscores the efficiency and effectiveness of our Improved Implementation, making it a powerful tool for large-scale data processing. Notably, we are further looking into optimizing the overall process thus potentially making the overall time even shorter.

Metric	Spryker Standard Implementation	Our Improved Implementation
Number of Records	**758k**	**758k**
Import and Processing Time	**60min**	**10min**
Publishing Time to Storefront	**120min**	
Total Time from Import to Storefront	**180min**	**10min**



Comparative Import and Processing Times for 750k records
(Spryker Standard Implementation vs. Improved Implementation)

94% Overall Performance Gains = **18x** Faster Overall Performance



Conclusion

Our implementation was tested and verified on a real-life project, resulting in a significant improvement in processing times. This enhancement is crucial for maintaining product updates and ensuring consistency across the storefront.

Summary of Findings

Our changes to the data processing in Spryker Cloud Commerce OS have led to a big 94% increase in overall performance. That's 18 times faster than Spryker Standard Implementation. This means businesses can handle more data, update their online stores quicker, and give their customers the most up-to-date information.

By getting rid of the less efficient parts of the Spryker Standard Implementation we've cut down on wasted resources, aligning with Spryker's commitment to sustainability. By optimizing resource usage, we're contributing to a more sustainable and energy-efficient e-commerce ecosystem.

Additionally, our solution is flexible. It's designed to support business growth and change, making sure it stays efficient and reliable in the long run. Whether we are supporting a small business planning to grow, or a large business looking to optimize, our solution can adapt to their needs.

Furthermore, this mechanism will be continuously improved and adjusted according to specific project needs, ensuring a tailored fit for every unique client and requirement.