E-COMMERCE INTEGRATIONS AS ORDER MANAGEMENT SYSTEM

MuleSoft Case Study

Abstract

A company leader in the weight loss and maintenance industry wanted to expand their retail business and sell its products in an E-Commerce site. The IBM WebSphere Commerce (WCS) platform was selected to build the digital store and the MuleSoft Anypoint Platform to develop integrations between WCS and external systems like third-party logistics (3PLs). The E-Commerce solution did not include an order management system (OMS), thus the order entry and processing responsibilities were divided between the WCS platform and the Mule applications.

Dr. Javier Navarro-Machuca

javier.navarro@ioconnectservices.com

CASE STUDY

Mule integrations as OMS

A company leader in the weight loss and maintenance industry wanted to expand their retail business and sell its products in an E-Commerce site. The IBM WebSphere Commerce (WCS) platform was selected to build the digital store and the MuleSoft Anypoint Platform to develop integrations between WCS and external systems like third-party logistics (3PLs). The E-Commerce solution did not include an order management system (OMS), thus the order entry and processing responsibilities were divided between the WCS platform and the Mule applications.

The Challenge

Every month, the order settlement and reconciliation reports showed hundreds of orders that were incomplete, and therefore they couldn't be settled on time. The E-commerce team had to troubleshoot the incomplete orders manually to identify their current status and find the cause of the problem. One common issue was that orders went "lost" in the system when one of the integration applications failed. The failures in the integration apps were very inconsistent and difficult to reproduce for a root analysis.

The applications that were developed to integrate the client's WCS platform with external systems presented an erratic behavior and multiple problems that were difficult to trace. In fact, most integrations did not report errors due to naïve implementations on error handling and fault tolerance practices like swallowing exceptions for the sake of guaranteeing application uptime. The failures were very inconsistent and some of them were hard to analyze as well. Extensive manual troubleshooting and analysis on the poor application logs were needed to identify the cause of the problem.

One critical problem was that the integrations apps used to "lose" processing orders when they failed to connect with an external system like a Web Service or FTP server location. Since the apps did not report exceptions, the monitoring tools were wrongly showing good health status checks. These missing orders were caught by the stakeholders when running monthly settlement and reconciliation reports. Once they were identified, it required too many manual hacks in the WCS platform and integration apps to enable a re-transfer to the 3PLs systems. Moreover, the client wanted to onboard new 3PLs which introduced a new challenge since the applications were implemented as monolithic integrations. Adding a new system to the existing flows required to add excessive conditional logic to route the order line items, forcing full regression testing to the apps for every change.

The Solution

A new architecture approach based on Microflows is needed to improve the integrations between WCS and the external systems. The new approach should allow to query the current status and history of the fulfillment integration for the orders and their line items. The integrations will be monitored and alert controls will be placed to systematically report failures, performance degradation, and potential order item shipment delays.

The existing implementation was analyzed thoroughly to determine the best

Headquarters: Berkley Heights, NJ

Industry: Professional Services

Website: www.ioconnectservices.com

Challenge Overview

- Orders are lost when an integration app fails to connect to an internal or external system.
- The integrations show inconsistent failure behaviors.
- The integrations do not report errors. Extensive manual troubleshooting and log analysis are needed to detect failures.

Solution

- A complete overhaul of the architecture of the integration apps is required. Microflows are used as the architecture approach.
- Integration database is introduced to record order status and activity history.
- Error handling and fault tolerance best practices are implemented.
- Activities can be re-played from the point of failure.

Results

- The solution monitors, tracks, and alerts when the integrating systems are not available without losing orders.
- Incomplete orders have been reduced by 99.5%
- Preventive measures like potential shipping delays are analyzed from the integration database to alert stakeholders, improving order settlement significantly.
- E-commerce revenue increased 35%

tactic to address all the presented problems: a complete overhaul of the architecture of the integration applications was required. Due to the criticality of the use cases that the integrations needed to execute, Microflows implemented in Mule apps have been selected as the architecture approach. For this reason, the existing applications were analyzed, decomposed, refactored, and in some cases replaced with new flows to facilitate the Microflows migration.

The implementation of Microflows follows the single responsibility principle, this means that each app will be in charge of one single transaction and it will execute the transaction via the unit of work pattern to guarantee accurate state control in case of a failure. Also, a database schema in a MySQL server cluster has been introduced to store the statuses of the orders and the activity logs while the orders are being processing by the Microflows. The integrating Microflows record the current activity status of the orders and the line items upon transaction completion, they also register the activity and results to the database and the application logs to facilitate any future analysis. The inter-Microflows communication is carried out by passing permanent messages in permanent queues in an ActiveMQ cluster to ensure that the messages and the queues definitions will be available regardless of a server downtime or a system restart. The setup of the queues and the cluster of the message broker makes this solution very resilient as the delivery and consumption are guaranteed. This is possible since the messages are stored in the hard drive and replicated to all the ActiveMQ server nodes.

"Since we went live with the solution, no order has got lost in the Mule integrations, and we are able to track each one of the line items from the moment the order is placed by the customer to when the products are delivered to the destination."

-Al Nolan, E-Commerce Solution Architect

If a Microflow has difficulties to integrate with any of the 3PLs external systems, it retries the delivery for a pre-configured number of times. In the case that the retry limit is reached, the message is placed in to a dead letter queue. A dead letter queue is an instance of the dead letter channel pattern implemented with a message queue. Failed messages in a dead letter queue can be further analyzed either manually or systematically to report the cause of the failure. In fact, there is one dead letter queue for each "normal" message queue defined in the message broker. WCS is the only system that places order documents in the Mule Microflow apps, therefore no bad-formed content is expected, but on the condition that a message contains a "poisoned" payload, instead of sending it to a formal poison queue, the message gets tagged as poisoned and sent to the corresponding dead letter queue to be further examined. All the dead letter queues are observed by an ActiveMQ plugin of the Datadog monitoring service. Technical stakeholders get automated alerts as soon as one message arrives to any of the dead letter queues. Other alert controls have been configured to detect integration downtime and performance degradations. Each Microflow app can be scaled out or down independently, this helps to react on transactional bursts without affecting the processing throughput and to free resources in case of a low system demand.

In the event that a downtime of an external system was the reason of messages being placed in a dead letter queue, the messages can be moved back to the normal queue to resume the integration processing. All the Microflows also expose an https endpoint that triggers the execution of the integration flow. This trigger mechanism enables other applications and systems to interact with them safely for re-play purposes. Manual re-plays can also be executed in extreme scenarios.

The Results

The monthly reports show that incomplete orders have been reduced by 99.5% and the remaining incomplete orders are identified systematically. Stakeholders receive alerts on potential shipping delays so that preventive measures can be taken to complete the fulfillment of the order within the expected shipping date timeframe. The revenue generated by the E-commerce sales increased by 35% since the solution was launched.

No more lost orders have been reported since the solution went live. In the event of a failure, the stakeholders receive a notification instantly via the monitoring console, email or the corporate messaging system. To integrate a new 3PL system, the team just needs to develop a new Microflow adapter and subscribe it to its dedicated queue that is dynamically created based on the new 3PL name. Besides, the test coverage can be reduced safely to the new Microflow and any external system it connects since the functionality implemented in each Microflow is decoupled from the others – this is due to the nature of the asynchronous communication via message queues.

MuleSoft®

IOCONNECT SERVICES