World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:18, No:10, 2024

End To End Process to Automate Batch Application

Nagmani Lnu

Abstract**—**Often, quality engineering refers to testing the applications that either have a User Interface (UI) or an Application Programming Interface (API). We often find mature test practices, standards, and automation regarding UI or API testing. However, another kind is present in almost all types of industries that deal with data in bulk and often get handled through something called a batch application. This is primarily an offline application companies develop to process large data sets that often deal with multiple business rules. The challenge gets more prominent when we try to automate batch testing. This paper describes the approaches taken to test a batch application from a financial industry to test the payment settlement process (a critical use case in all kinds of FinTech companies), resulting in 100% test automation in test creation and test execution. One can follow this approach for any other batch use cases to achieve a higher efficiency in their testing process.

Keywords**—**Batch testing, batch test automation, batch test strategy, payments testing, payments settlement testing.

I. Introduction

APPLICATION under test is a payment processing application that runs in batch mode to settle payment transactions between the payment processor and merchant. It processes thousands of transactions in less than 15 minutes, settling millions of dollars every time. The batch is expected to settle transactions based on rules such as card brand (Master, Visa, Discover, etc.), types of cards (debit, credit), etc. It was observed that during regression, the quality engineer mainly triggered the batch to check if the batch got executed without any problem. This is a crucial validation; however, since the team was not validating all possible scenarios, they very often used to get bugs in production. Testing all scenarios was complex as the team required at least five to six hours to key those scenarios manually every time. This is a widespread problem across the industry where the quality engineering team is building a robust testing approach around UI and API, mostly real-time applications, but lacks the same for batch applications. Batch is significant and present in almost every kind of industry. As per AI multiple, depending on the industry type, the company is designing a batch application to automate their specific use case, for example, to process the lead and order management in marketing and sales or to perform end of day processing in the financial sector [1]. In the Amazon Web Services (AWS) website for batch processing, batch process systems are used to process various data types and requests. Some of the most common batch-processing jobs include weekly/monthly billing, payroll, inventory processing, report generation, data conversion, subscription cycles, and supply chain fulfillment [2].

Nagmani Lnu is with Director of Quality Engineer at a FinTech Company, San Antonio, USA (e-mail: nagmanijobs@gmail.com).

With this, we can confidently say that the batch application is critical and processes essential job functions for the company. Any lack of testing can result in a massive loss in company reputation and even create a financial loss. So, the company must put in place a dedicated strategy that includes both functional testing and the right automation strategy.

II. Domain and Technology Background

Before we talk about the solution, it is important to explain the application briefly in terms of its functionality and technical components. As mentioned in Section I, the application pertains to the payment process, which typically falls within the domain of the FinTech industry. From the domain/functionality perspective, payment process application specifically for cards has three components that are *authentication, reconciliation,* and *settlement,* as shown in Fig. 1.

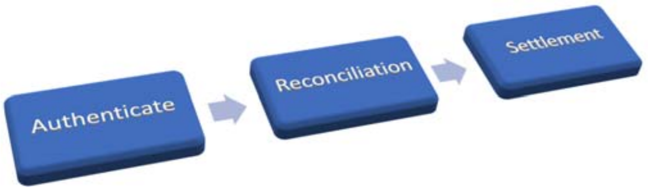


Fig. 1 Card Transactions Flow

Authentication involves approval or rejection of a card transaction that a card processor receives through the payment gateway from credit card companies such as Visa, Master, Discover, and AMEX. Card processors bundle all the approvals and send them to the card network for reconciliation. This happens via a batch process. Once the reconciliation process is completed, the settlement process starts that eventually deposits the funds to the merchant bank account [3].

With the above explanation, we understand that the transaction authentication process occurs in real-time, while the settlement and reconciliation processes are conducted entirely through batch processing. As for the design of such systems, there are various approaches that companies may take. Some companies may opt to have everything built as one extensive monolithic system, while others may adopt a modular approach, leveraging microservices architecture or other modern design patterns. Given the dynamic nature of FinTech and the emergence of numerous startups new, there is diversity in how process payments are designed and implemented. As per the demand sage, there are 26,300 fintech startups worldwide as of 2024, of which 10,755 are in the USA. Experts project that the compound annual growth rate (CAGR) of the Fintech industry will grow at 11.7% from 2017 to 2024 [4]. In many cases, startups in the FinTech industry adopt the latest tech stacks like cloud computing and build this application as microservices, which is similar to the approach take in our project. Fig. 2 from AWS shows how a batch microservice can be developed using AWS Lambda and AWS Batch [5].

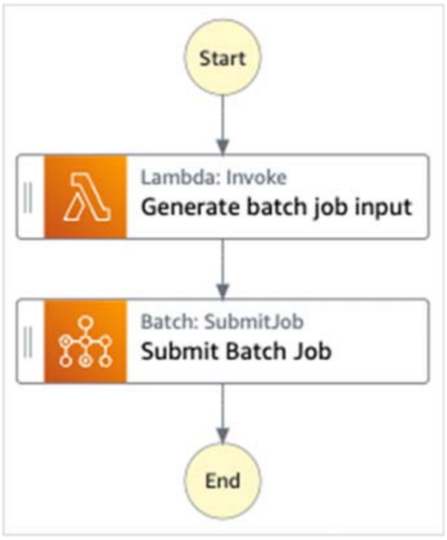
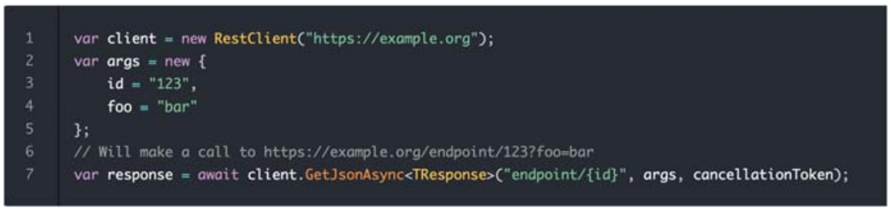


Fig. 2 Image to Process Batch Job from AWS [5]

III. Solution

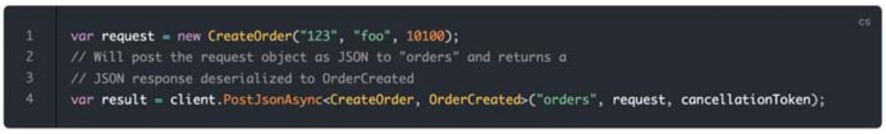
The project team was asked to design an automated solution that can perform end-to-end (authentication to settlement) testing with all business rules. Authentication testing was easy as it covered API. The team created an automated test suite to test all authentication-related functionalities. Testing reconciliation and the settlement process was tricky as it was batch built with AWS stack. As mentioned in the previous section, the reconciliation process gathers all the transaction approvals during the day and sends them to the card network for reconciliation. To test this process, quality assurance must ensure, 1) to create all kinds of transactions, 2) trigger batch/ Lambda to bundle those transactions' approval in a text file, 3) validate text files, which are input files sent to the card network and response files received from the card network.

- *Create all kinds of transactions:* This is a crucial step to create all possible inputs for the batch process, which is often overlooked or not thoroughly planned, and ultimately ends up getting some edge case scenarios failing in the production. It is observed that most of the time, quality assurance engineers focus on ensuring that the batch runs with some scenarios and then presume that it will work for every condition. Transactions can be multiple types; for example, a card transaction can be submitted for either for Visa, Master or Discover card, and the transaction can be a debit transaction or credit transaction, etc. The card processor and card network treat all these types differently and process as per the rule designed for them. Once the team has collected all the transactions requirements, the next step was to create them automatically. Considering the transaction is an API system, the team had chosen RestSharp tool to automate the authentication step. RestSharp is a very popular HTTP client library for .NET featuring automatic serialization and deserialization, request and response type detection, as well as a variety of authentications and other useful features. It is being used by hundreds of thousands of projects [6]. Quality engineers can use RestSharp libraries to retrieve the merchant details by doing a GET call and eventually post a transaction by making a POST request, as shown in the sample code in Figs. 3 and 4 taken from the RestSharp library [7].



```
var client = new RestClient("https://example.org");
var args = new {
    id = "123",
    foo = "bar"
};
// Will make a call to https://example.org/endpoint/123?foo=bar
var response = await client.GetJsonAsync<TResponse>("endpoint/{id}", args, cancellationToken);
```

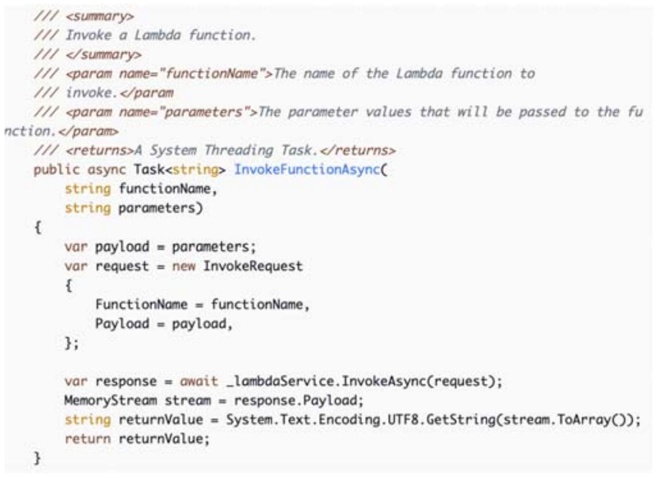
Fig. 3 Sample Code from RestSharp to Perform Get Request [7]



```
var request = new CreateOrder("123", "foo", 10100);
// Will post the request object as JSON to "orders" and returns a
// JSON response deserialized to OrderCreated
var result = client.PostJsonAsync<CreateOrder, OrderCreated>("orders", request, cancellationToken);
```

Fig. 4 Sample Code from Rest Sharp to Post Request [7]

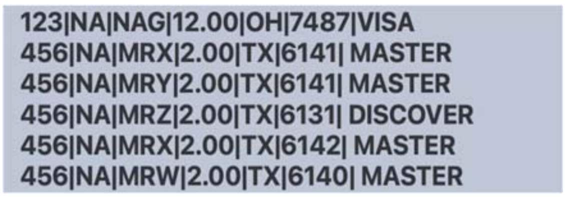
- *Trigger batch/Lambda to bundle those transactions' approval in a text file:* Once the input transactions are submitted, quality assurance engineers must trigger the Lambda function or AWS Batch to process all the transactions approved from the authentication process. There are two ways to do this: 1) logging to the AWS console and manually triggering the required Lambda function or job, or 2) processing it automatically. Most of the time, AWS Lambda gets associated with the API gateway, and one can use RestSharp (as explained above) to invoke it. However, this setup mostly happens for online applications. As the batch process is offline, we must invoke a trigger to begin the process. To do this, the team decided to use AWS SDK. This tool or set of .NET libraries given by AWS that allows us to develop an application that interacts with AWS [8]. Using this, the team was able to create automation to invoke the AWS Lambda function, as shown in the sample code in Fig. 5 taken from the AWS website [11]



```
    /// <summary>
    /// Invoke a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// invoke.</param
    /// <param name="parameters">The parameter values that will be passed to the fu
nction.</param>
    /// <returns>A System Threading Task.</returns>
    public async Task<string> InvokeFunctionAsync(
        string functionName,
        string parameters)
    {
        var payload = parameters;
        var request = new InvokeRequest
        {
            FunctionName = functionName,
            Payload = payload,
        };

        var response = await _lambdaService.InvokeAsync(request);
        MemoryStream stream = response.Payload;
        string returnValue = System.Text.Encoding.UTF8.GetString(stream.ToArray());
        return returnValue;
    }
```

Fig. 5 Sample Code from AWS to Trigger Lambda Functions [11]

- *Validate Text files, which are Input files sent to the card network and Response Files received from the card network:* This is the last but very critical step in the overall batch process. Batch output files generally separate values by pipe as shown in Fig. 6. The quality engineer needed to ensure the integrity of the file by performing validations like: ensuring all submitted approval records are available, no duplicate records are present, proper values are passed as per the requirement, for example, some company will have a requirement to show full state name instead of abbreviation or VI instead visa, calculations happen correctly if there are any, and validation for business rules as per the requirement.

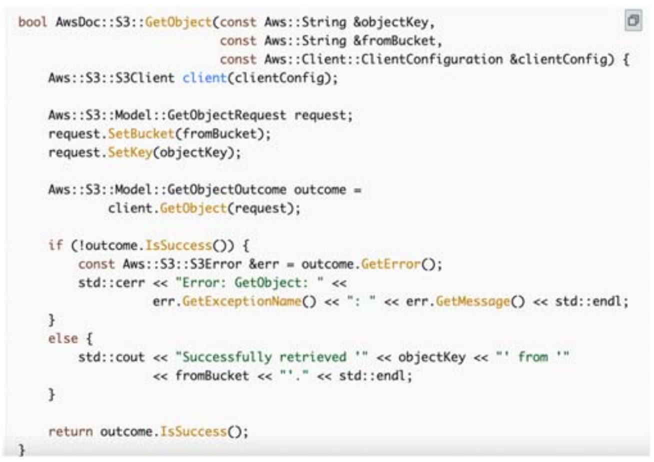


```
123|NA|NAG|12.00|OH|7487|VISA
456|NA|MRX|2.00|TX|6141| MASTER
456|NA|MRY|2.00|TX|6141| MASTER
456|NA|MRZ|2.00|TX|6131| DISCOVER
456|NA|MRX|2.00|TX|6142| MASTER
456|NA|MRW|2.00|TX|6140| MASTER
```

Fig. 6 Sample Batch Output File

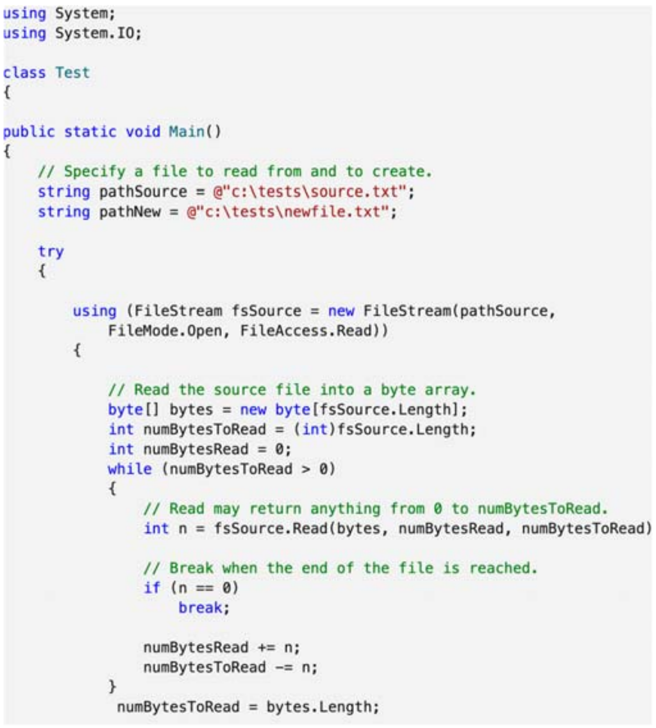
In most designs with AWS, files get dropped in an S3 bucket, a service created by Amazon to store files [9]. Same as Lambda, the quality engineer used AWS SDK to connect to S3, as shown in the sample code in Fig. 7 [10] taken from the AWS website to retrieve the file.

Once the file gets accessed through code, to validate all the business rules, the quality engineer uses the .NET system IO FileStream library to read the file, as shown in the sample code shown in Fig. 8 taken from Microsoft [12], to store the relevant values in their corresponding array after splitting the contents from the pipe delimiter. For example, all names were held in a name array, where the card was in card type. It was expected that the quality engineer should store the expected values while creating the transactions for authentication so they can compare them with the file value to validate all the business rules. The number of records and duplicate values were also validated through standard .NET code.



```
bool AwsDoc::S3::GetObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
            client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
                  << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

Fig. 7 Sample Code from AWS to Access S3 [10]

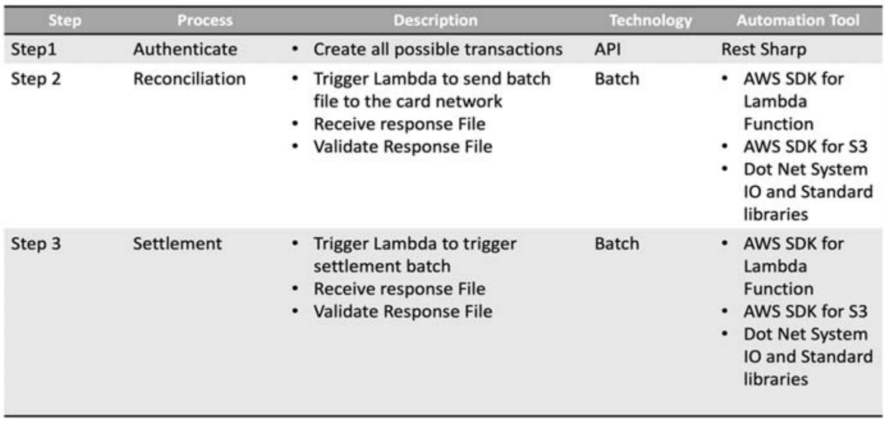


```
using System;
using System.IO;

class Test
{

public static void Main()
{
    // Specify a file to read from and to create.
    string pathSource = @"c:\tests\source.txt";
    string pathNew = @"c:\tests\newfile.txt";

    try
    {

        using (FileStream fsSource = new FileStream(pathSource,
            FileMode.Open, FileAccess.Read))
        {

            // Read the source file into a byte array.
            byte[] bytes = new byte[fsSource.Length];
            int numBytesToRead = (int)fsSource.Length;
            int numBytesRead = 0;
            while (numBytesToRead > 0)
            {
                // Read may return anything from 0 to numBytesToRead.
                int n = fsSource.Read(bytes, numBytesRead, numBytesToRead)

                // Break when the end of the file is reached.
                if (n == 0)
                    break;

                numBytesRead += n;
                numBytesToRead -= n;
            }
             numBytesToRead = bytes.Length;
```

Fig. 8 Sample Code from Microsoft to Read File

It completes the validation for the reconciliation process. After the reconciliation process, quality engineers validate and eventually automate the settlement process, which is also a batch process and involves similar steps of invoking Lambda functions to start the batch process and then receiving files back from the card network ensuring that all transactions are settled. Money will be deposited to the merchant account after deducting the required fees.

By using multiple tools in a proper combination, the team could automate the entire payment process within a single automation script, as shown in Fig. 9, of overall test automation steps.



Step	Process	Description	Technology	Automation Tool
Step1	Authenticate	• Create all possible transactions	API	Rest Sharp
Step 2	Reconciliation	• Trigger Lambda to send batch file to the card network • Receive response File • Validate Response File	Batch	• AWS SDK for Lambda Function • AWS SDK for S3 • Dot Net System IO and Standard libraries
Step 3	Settlement	• Trigger Lambda to trigger settlement batch • Receive response File • Validate Response File	Batch	• AWS SDK for Lambda Function • AWS SDK for S3 • Dot Net System IO and Standard libraries

Fig. 9 Overall Test Automation Steps

IV. Other Testing Type

So far, we have seen the approach for automating the functional testing of the batch applications. However, when we talk batch, we always talk about volume. While functional testing is essential, we must ensure that the batch processes large data sets without problems. The quality engineering team needs to work with the business to understand the peak volume, and accordingly, use the same automation processes described above to generate the required transaction loads, authenticate those, and then finally, test to see if those get successfully reconciled and settled.

V. Conclusion and Results

With this approach, the team met the business demand and achieved 100% automation to validate different payment batches, which was almost considered a manual testing scope earlier. Overall, the test execution effort is cut down to 50%, and simultaneously, the team can confidently release the functionality.

This solution can be replicated in any batch application if the team follows a similar tool/technology stack to develop it.

References

[1] “40+ Best Batch Processing Applications You Must Know,” research.aimultiple.com. https://research.aimultiple.com/batch-processing-applications/
[2] “What is Batch Processing? - Enterprise Cloud Computing Beginner’s Guide - AWS,” Amazon Web Services, Inc. https://aws.amazon.com/what-is/batch-processing/
[3] “How Credit Card Processing Works: Understanding Payment Processing,” CardFellow Credit Card Processing Blog, Jul. 01, 2011. https://www.cardfellow.com/blog/how-credit-card-processing-works/
[4] D. Ruby, “43 Fintech Statistics For 2024 (Startups, Financials & Trends),” Dec. 30, 2023. https://www.demandsage.com/fintech-startups-statistics/#:~:text=as%20of%202023.- (accessed Jan. 17, 2024).
[5] “AWS Batch with Lambda - AWS Step Functions,” docs.aws.amazon.com. https://docs.aws.amazon.com/step-functions/latest/dg/sample-batch-lambda.html (accessed Jan. 17, 2024).
[6] “RestSharp,” restsharp.dev. https://restsharp.dev/
[7] “Usage | RestSharp,” restsharp.dev. https://restsharp.dev/usage.html#json-requests (accessed Jan. 17, 2024).
[8] “AWS SDK for .NET,” Amazon Web Services, Inc. https://aws.amazon.com/sdk-for-net/
[9] AWS, “Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service,” Amazon Web Services, Inc., 2023. https://aws.amazon.com/s3/
[10] “Get an object from an Amazon S3 bucket using an AWS SDK - Amazon Simple Storage Service,” docs.aws.amazon.com. https://docs.aws.amazon.com/AmazonS3/latest/userguide/example_s3_GetObject_section.html (accessed Jan. 17, 2024).
[11] “Lambda examples using AWS SDK for .NET - AWS SDK for .NET,” docs.aws.amazon.com. https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/csharp_lambda_code_examples.html (accessed Jan. 17, 2024).
[12] dotnet-bot, “FileStream.Read Method (System.IO),” learn.microsoft.com. https://learn.microsoft.com/en-us/dotnet/api/system.io.filestream.read?view=net-8.0 (accessed Jan. 17, 2024).