**Design Document: Cloud-based Data Integration Solution**

**1. Introduction**

This design document outlines a suitable solution for integrating user data from multiple systems of Company A into a single system hosted on Company B's cloud-based environment. The data from Company A is inconsistent and might contain duplicate records due to acquisitions and mergers. The two data sources include a REST-based API and a file placed into a known location for processing. The solution aims to address data consistency, resiliency to network changes, error handling, and efficient use of cloud resources.

**2. Solution Overview**

The proposed solution involves a cloud-based data integration platform that performs data extraction, transformation, and loading (ETL) operations to consolidate user data from Company A's systems into Company B's system. The solution will be hosted on a modern cloud platform (e.g., AWS, Azure, Google Cloud). The key components of the solution are as follows:

**2.1. Implementation**

The solution will take advantage of and will be implemented on SSIS in ASP .NET. For demonstration purpose, there will be no SQL server involved in this project as the target SQL server environment is unclear. All data manipulation will be carried out through text files stored locally. Deployment requires user to change an SSIS variable that modifies the data files location once the solution is downloaded

**2.2. Data Structure**

Upon review, JSONPlaceholder’s user data structure is as follows:

Class Geo{

Decimal lat

Decimal lng

}

Class Address {

String street

String suite

String city

String zipcode

Geo geo

}

Class Company{

String name

String catchphrase

String bs

}

Class user{

Int id

String name

String username

String email

Address address

String phone

String website

Company company

}

A custom generated csv data is named users.csv and located at the root directory. The data is only consist of a flat data structure

String First\_name

String Last\_name

String email

String phone

String company\_name

Int id

String phone

The two sets of data have quite different data fields and we aim to integrate two sets of data into one.

**2.3. Data Consistency**

As the sets of data are quite different. To simplify the demonstration, we only the phone numbers to identify a unique user. Although names and email address could also be considered to identify a person, these factors are disregarded in this case.

Within this project, when the same phone number appears in both sets of data, the corresponding row will be sent to a separate location for evaluation. The two separate CSV files for the two data sources are "csv\_duplicates.csv" and "json\_duplicates.csv." They will be generated under /root/csv/manual\_process. Data integrator can now decide further on processing the data or inserting the data by inspecting the original data sources.

The common fields shared by two data sets are decided as follows, they will be posted to the mock API:

String FirstName

String LastName

String Email

String Phone

String CompanyName

Remaining data will be abandoned.

**2.4. Data Integration Layer**

The data integration layer will be responsible for orchestrating the data integration process. It will include the following components:

2.4.1. Data Extraction and Processing

* For the REST-based API source, HttpClient will be used to be called from a script component to consume JSONPlaceholder’s user API.
* As names in JSONPlaceholder is mingled with first and last, they will be processed into First\_name and Last\_name before the data is staged to the SSIS data flow.
* As there are extensions numbers after some phone entries in JSONPlaceholder. The extension number will be trimmed for comparison. Also hyphen and parentheses will be removed to form a new column phone\_main to enable direct comparison.
* For the file source, native SSIS Flat File source will be used to read the CSV file.
* Finally both processed data sets will be saved as CSV files for the next stage process.

2.3.2. Data Merging and Posting through RESTful API

* Data merging is separated from the data extraction process above because a cache file will need to be created first before SSIS starts a lookup action. Therefore all data will be saved as csv files (csv\_userdata\_before\_merge, json\_userdata\_before\_merge.csv) for the next merge process.
* After a lookup is performed, rows without duplicated phone numbers will be merged into a table (FirstName, LastName, Email, Phone, CompanyName) from JSONPlaceholder and the generated CSV file..
* Once all data is gathered, SSIS will fire a script component to post EngageUser json object to the Company B API (<https://localhost:7181/PostUser>). The programme will record all HTTP response message in PostResults.txt under the csv folder.

**2.4. Resiliency to Network Changes**

As the deployment environment will be different from the development environment, the solution is unable to use SQL server (due to change of connection strings and SQL version issues) and to determine a designated file location for the temporary files created. Deployment will have to reconfigure the root CSV folder location.

**2.5. Error Handling**

Robust error handling mechanisms will be incorporated throughout the data integration pipeline:

* If any JSON object fails to be parsed, script component will fire exception to inform extraction failure.
* If any of the CSV rows are failed to parse, it will log error to Phone\_Conversion\_Error.txt.
* If SSIS fails to post to the API, it will fire exception to inform the user.
* All POST results will be recorded. Users will be find out if there are any errors during the communication between SSIS and the API.

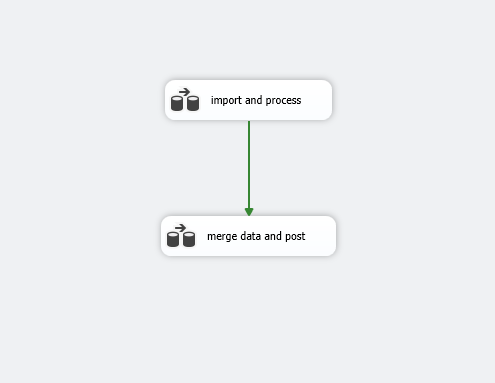
**2.6. Efficient Use of Resources**

* **SQL Server deployment**: SQL server can be used to efficiently perform the merge action. Due to environment restriction. Only CSV files are used as a media to read/write between Control Flows. As the scale of the data is very small. Lookups can be performed considerably very quickly in this case. In reality, data is involved with relational database, SQLServer is required.
* **Asynchronous read/write**: as script components are designed to run synchronously, if users have to be posted one by one, for large amount of data to be transmitted via remote network, the process can take a long time.

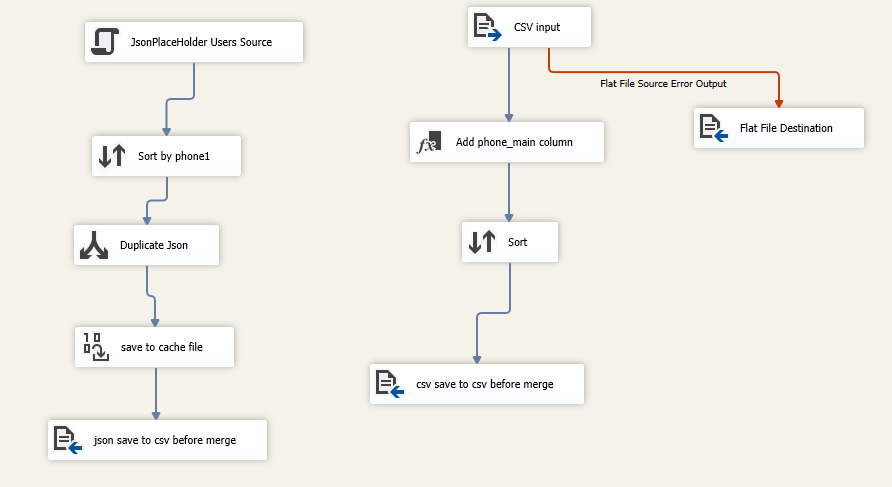
**3. High-Level Architecture**

The high-level architecture of the proposed solution is depicted below:

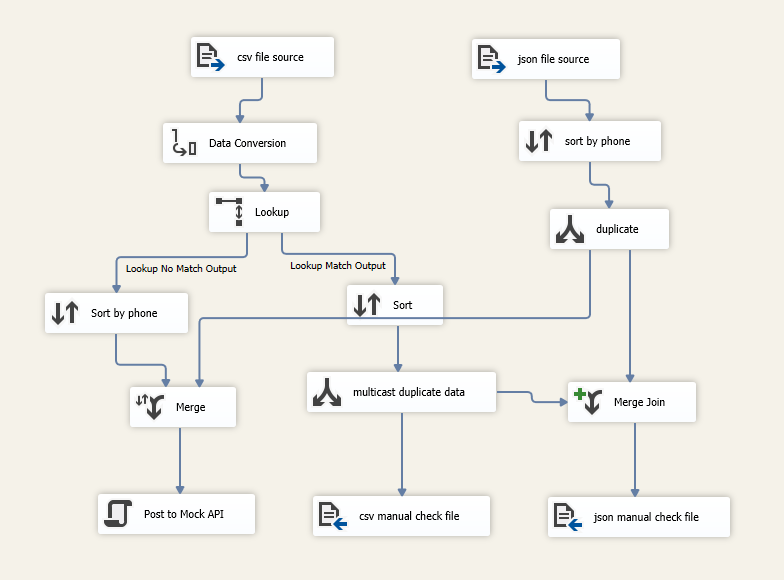
**Control Flow:**



**Data Import and Processing**



**Merge Data and POST**

****

**4. Conclusion**

The cloud-based data integration solution outlined in this design document provides a comprehensive approach to integrate user data from multiple systems of Company A into Company B's system. The solution efficiently resolved data consistency, tackled error handling using SSIS data redirection mechanism. This implementation avoided cumbersome code and provided a clear picture of data processing execution.