|  |
| --- |
| HMS |
| Prepay 2.0 |
| Architecture |

|  |
| --- |
| Tenney, Robert  9-20-2016 |

# Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Performer(s) |
| 05/23/2019 | 1.0 | Document creation | Robert Tenney |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

[1 Document Revision History 1](#_Toc9531745)

[2 Table of Contents 2](#_Toc9531746)

[3 Introduction 4](#_Toc9531747)

[3.1 Purpose 4](#_Toc9531748)

[3.2 Scope 4](#_Toc9531749)

[3.3 Audience 4](#_Toc9531750)

[4 Application Overview 5](#_Toc9531751)

[4.1 Diagram 5](#_Toc9531752)

[4.2 Description 5](#_Toc9531753)

[5 Technology stack 6](#_Toc9531754)

[5.1 Apache Hadoop 6](#_Toc9531755)

[5.2 Apache Hive 6](#_Toc9531756)

[5.3 Apache Nifi 6](#_Toc9531757)

[5.4 Apache Spark 6](#_Toc9531758)

[5.5 Apache Sqoop 6](#_Toc9531759)

[5.6 Azure Service Bus 6](#_Toc9531760)

[5.7 Microsoft SQL 6](#_Toc9531761)

[5.8 Scala 6](#_Toc9531762)

[6 Logical architecture 7](#_Toc9531763)

[6.1 Diagram 7](#_Toc9531764)

[6.2 Components 7](#_Toc9531765)

[6.2.1 Hadoop Data Lake (DLK) 7](#_Toc9531766)

[6.2.2 Azure Service Bus 7](#_Toc9531767)

[6.2.3 Nifi 7](#_Toc9531768)

[6.2.4 SQL Server 8](#_Toc9531769)

[6.2.5 Hadoop Applications 8](#_Toc9531770)

[6.2.6 HDFS (app/pi, app/fwa) 8](#_Toc9531771)

[6.2.7 Azure Blob Storage 8](#_Toc9531772)

[6.2.8 PI Portal Web Services 8](#_Toc9531773)

[6.2.9 Email 8](#_Toc9531774)

[7 Processes 9](#_Toc9531775)

[7.1 Data Flow Diagram 9](#_Toc9531776)

[7.1.1 Azure Service Bus 9](#_Toc9531777)

[7.1.2 Prepay Job Phases 10](#_Toc9531778)

[7.1.3 Prepay Process 11](#_Toc9531779)

[7.1.4 Edit Process 12](#_Toc9531780)

[7.2 Components 12](#_Toc9531781)

[7.2.1 Controller 13](#_Toc9531782)

[7.2.2 Data Preparation Area 13](#_Toc9531783)

[7.2.3 Edit Engine 13](#_Toc9531784)

[7.2.4 Exclusion Engine 13](#_Toc9531785)

[7.2.5 Ranking Engine 13](#_Toc9531786)

[7.2.6 Output Generator 13](#_Toc9531787)

[7.2.7 Rule & Configuration 13](#_Toc9531788)

[7.2.8 Approval (Future) 13](#_Toc9531789)

[8 Physical architecture 14](#_Toc9531790)

[8.1 Diagram 14](#_Toc9531791)

[8.2 Components 14](#_Toc9531792)

[9 Data Architecture 14](#_Toc9531793)

[9.1 Diagram 14](#_Toc9531794)

[9.1.1 Original Design 14](#_Toc9531795)

[9.1.2 Current State 15](#_Toc9531796)

[9.2 Data structure 15](#_Toc9531797)

[9.2.1 Application Data 15](#_Toc9531798)

[9.2.2 Configuration Data 15](#_Toc9531799)

[9.2.3 Rule Data 15](#_Toc9531800)

[10 Security 15](#_Toc9531801)

# Introduction

## Purpose

This document describes the initial design for the Prepay 2.0 Application; it lists and describes the different components involved in the system and their position in the logical design. Finally, it layouts and support the application physical design.

## Scope

This document is intended to be the primary reference to the Prepay 2.0 application architecture and high-level design. The document:

* Describes the technologies involved.
* Provides an architectural overview.

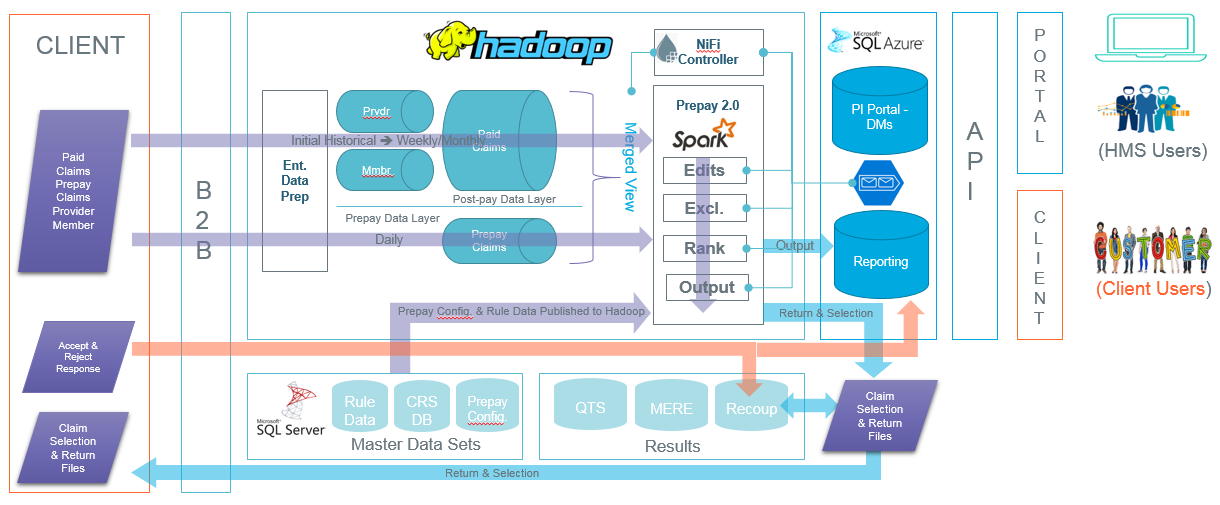
## Audience

The directed audiences of this document are:

* Enterprise and application architects
* Infrastructure team
* Development team
* Testing team
* Project managers

# Application Overview

## Diagram



## Description

# Technology stack

## Apache Hadoop

Hadoop is an ecosystem of open source components that fundamentally changes the way enterprises store, process, and analyze data. Unlike traditional systems, Hadoop enables multiple types of analytic workloads to run on the same data, at the same time, at massive scale on industry-standard hardware. CDH, Cloudera's open source platform, is the most popular distribution of Hadoop and related projects in the world (https://www.cloudera.com/products/open-source/apache-hadoop.html).

## Apache Hive

Apache Hive is a data warehouse system built on top of Apache Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in various databases and file systems (https://mapr.com/products/apache-hive/).

## Apache Nifi

Apache NiFi supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic (https://nifi.apache.org/).

## Apache Spark

Apache Spark™ is a unified analytics engine for large-scale data processing (https://spark.apache.org/).

## Apache Sqoop

Apache Sqoop(TM) is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases (https://sqoop.apache.org/).

## Azure Service Bus

Keep connected with Azure Service Bus, a cloud messaging system for connecting apps and devices across public and private clouds (https://azure.microsoft.com/en-us/services/service-bus/).

## Microsoft SQL

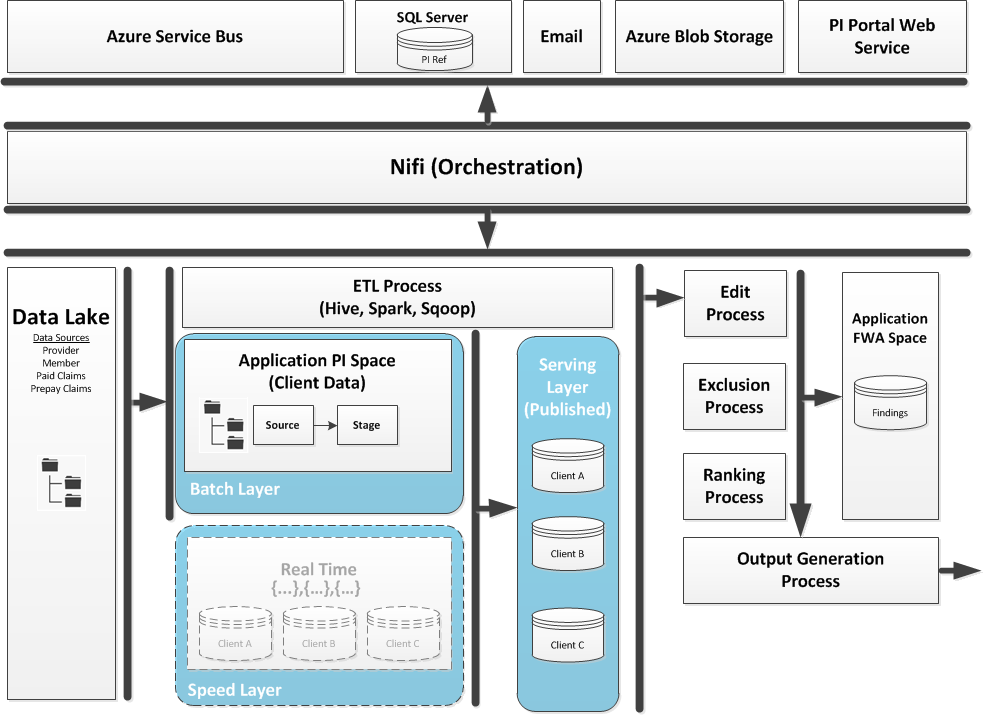
SQL Server is the foundation of Microsoft's data platform, delivering mission-critical performance with in-memory technologies and faster insights on any data, whether on-premises or in the cloud.

## Scala

Scala combines object-oriented and functional programming in one concise, high-level language. Scala's static types help avoid bugs in complex applications, and its JVM and JavaScript runtimes let you build high-performance systems with easy access to huge ecosystems of libraries (https://www.scala-lang.org/).

# Logical architecture

## Diagram



## Components

### Hadoop Data Lake (DLK)

All client files are landed by the B2B / EIM team in the Hadoop Data Lake. This is the gold copy of all data available to the application. As files land, messages with file details are posted to the Azure Service Bus EIM Topic.

### Azure Service Bus

ASB topics are used for communication between all decoupled processes.

* Prepay subscribes to the EIM topic to be notified of any new files related to PI or FWA.
* Each Prepay component receives start instructions from an ASB Topic subscription and posts its own status messages back to the topic.

### Nifi

Nifi is used to coordinate all processes within the application. It is responsible for:

* Reading and publishing all messages to the Azure Service Bus
* Moving data from the Data Lake to the Prepay application storage
* Executes all scripts and Spark packages
* Posting return files back to the client
* Moving output files to Azure Blob Storage to PI Portal for incremental data loads and edit reporting
* Logging all activity to SQL Server
* Sending all notification emails

### SQL Server

SQL Server is responsible for:

* Storing logs of file notifications from EIM
* Storing logs of status changes for all processes
* Storing client configuration data
* Detecting duplicate files received from DLK
* Detecting missing files expected from a client
* Detecting when all file shave been received from a client
* Storing all rules required by the edits, exclusions, and ranking

### Hadoop Applications

Most of the work of the application is implemented using various tools in Hadoop.

* Hive scripts are used to create all tables and execute ETL tasks
* Sqoop scripts are used to move configuration data between SQL Server and Hadoop
* Spark jobs execute the edits, exclusions, and ranking. Spark is also used to generate output files for the clients and the PI Portal.

### HDFS (app/pi, app/fwa)

Each client has its own Hive database named app\_pi\_{client\_code}. These databases hold tables for source files, staging layer, and publish layer.

There are two application databases:

* app\_pi\_reference holds configuration data and rules
* app\_fwa\_prepay holds Prepay findings.

### Azure Blob Storage

Stores files to be loaded to PI Portal in Azure SQL including:

* Members, Providers, Claims, & Rx Claims
* Edit Findings, Support, Rules, and Exclusions

### PI Portal Web Services

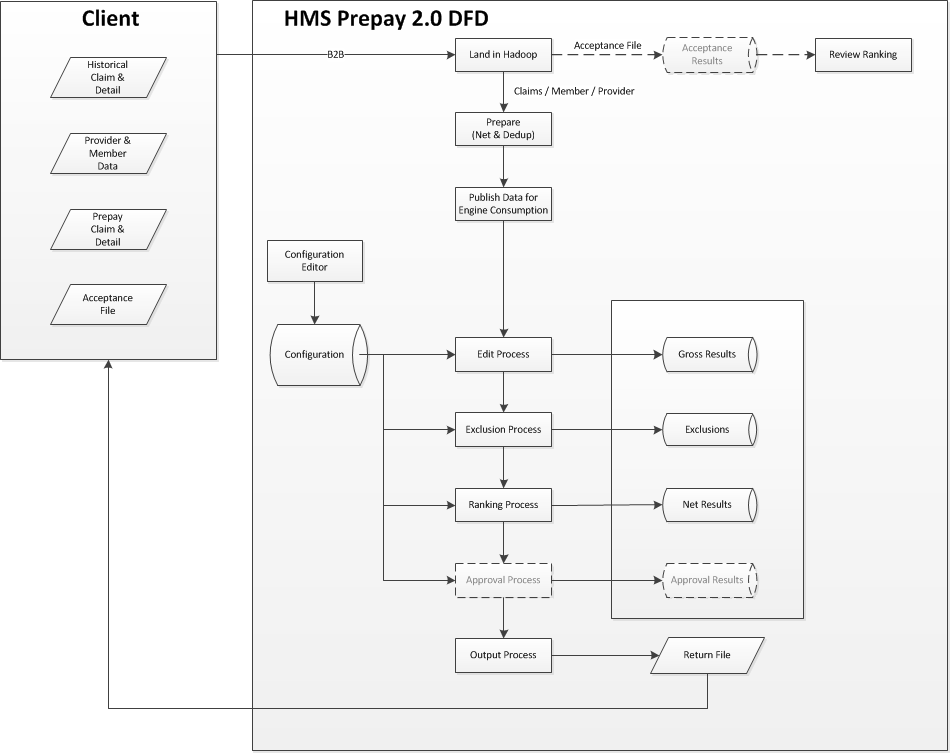
Web services are called by Nifi after uploading files to blob storage to initiate a data load to PI Portal.

### Email

All client and team notifications are sent via HMS SMTP services.

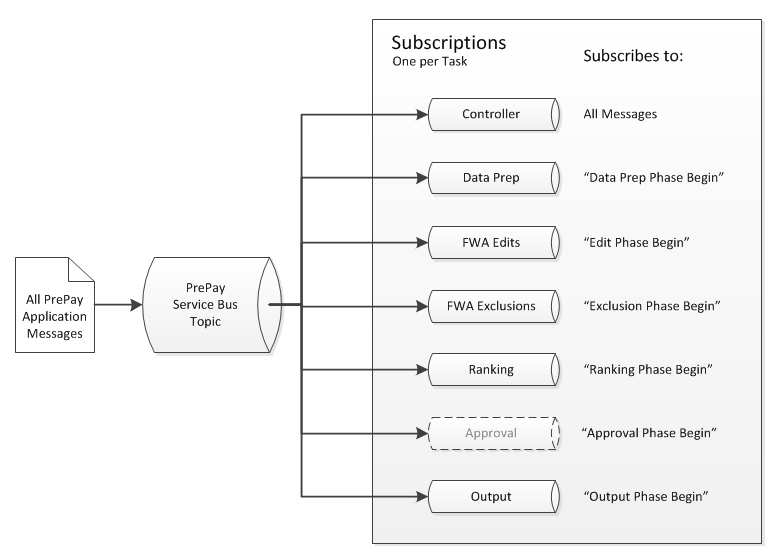
# Processes

## Data Flow Diagram



### Azure Service Bus

A pub-sub pattern is leveraged to facilitate an asynchronous workflow and to ensure decoupled subsystems. Upon data arrival a message is published in which the Nifi Controller is a subscriber. This launches the Data Preparation process. Once complete a message is published in which the next subscriber executes. This is repeated until the output results are generated.



### Prepay Job Phases

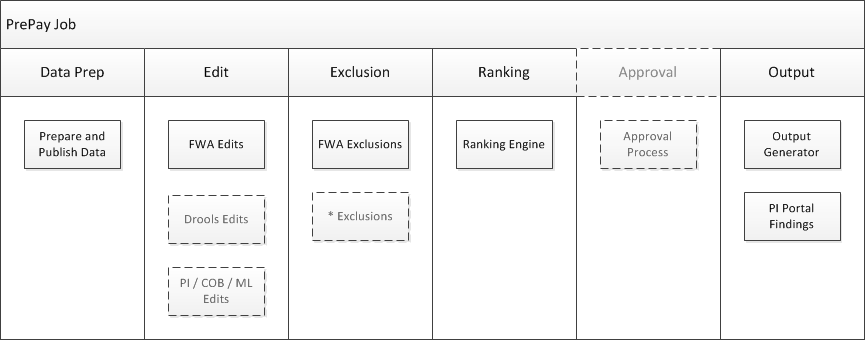
#### Data Prep - The prepay data preparation and publishing process performs the data preparation, deduping and netting of both paid and prepay claims.

#### Edit – The edit process executes all edits against all claims and stores the results of the edits. Each edit may have multiple edits hits.

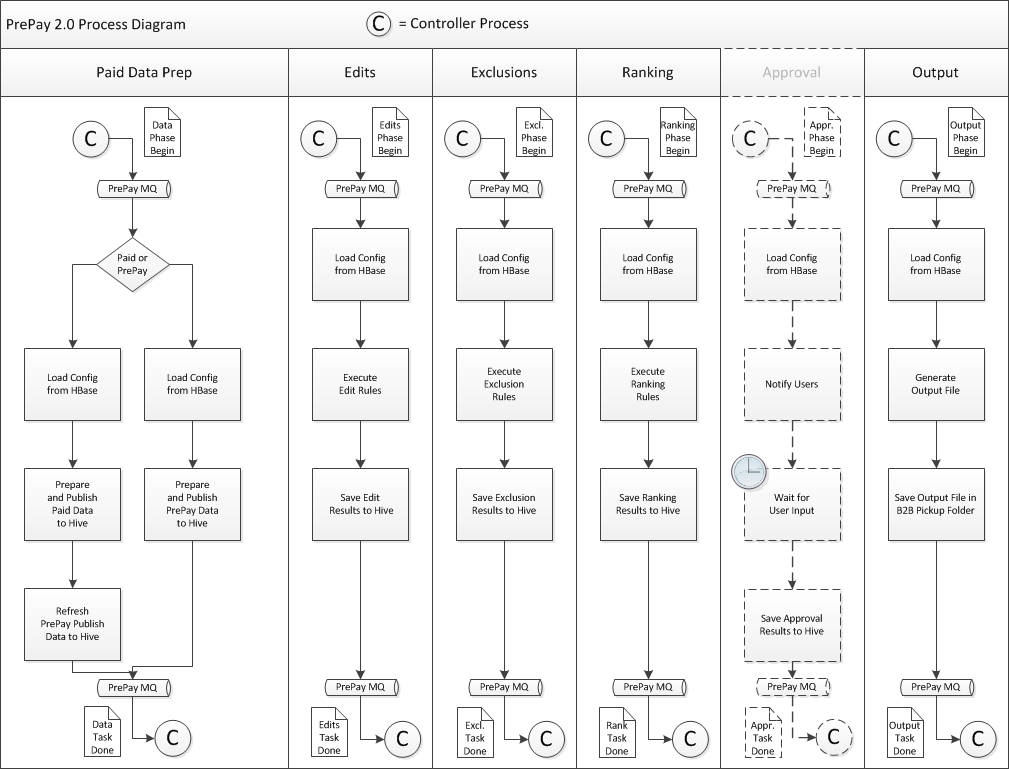
#### Exclusion – The exclusion process applies all exclusions against edit results. Each exclusion is recorded, and each finding may have multiple exclusions.

#### Ranking – The ranking process pulls in the client ranking configuration and considers all edit and exclusion results and recommends the strongest recommendation of a edit hit.

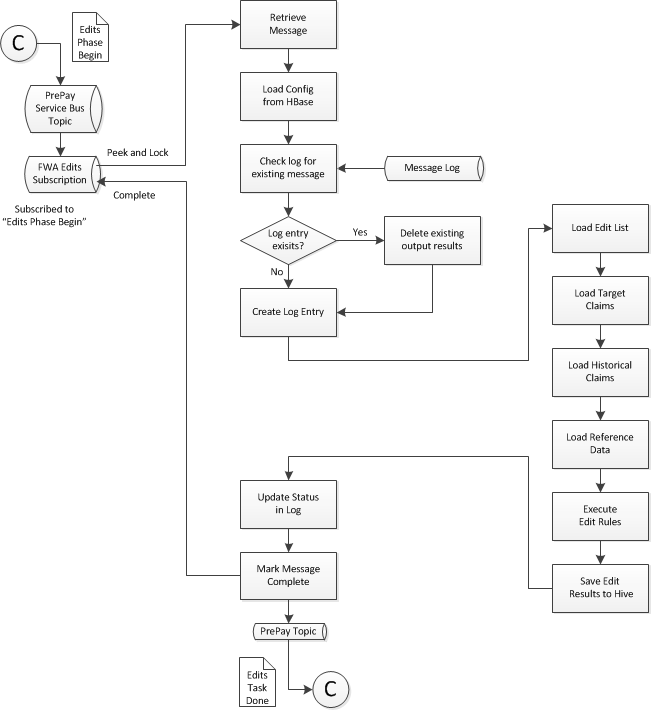
#### Output – The output process creates a client formatted result file to be returned to the client.



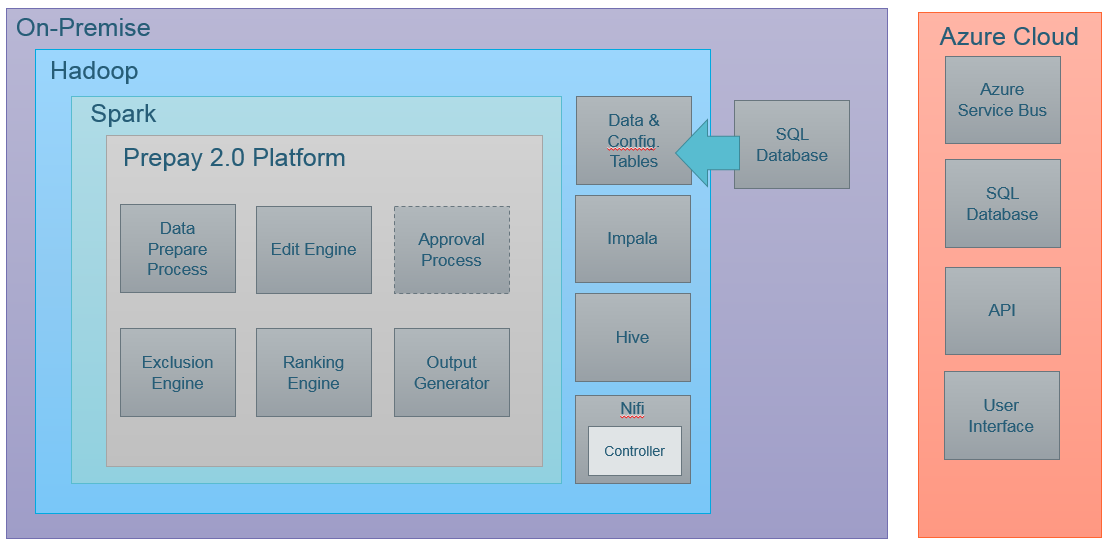
### Prepay Process



### Edit Process



## Components



### Controller

This high-level component represents the Prepay 2.0 controller which orchestrates the execution of each component.

### Data Preparation Area

This high-level component is an abstraction of the process needed to extract the data from the Corporate Data Lake. Once extracted to the PI Application space, deduping and netting of the data is performed.

#### File Detection & Grouping

#### Change Data Capture

### Edit Engine

This high-level component represents the edit engine which ultimately executes all edits across all data under evaluation.

### Exclusion Engine

This high-level component represents the exclusion engine which applies all exclusion rules against the edit results.

### Ranking Engine

This high-level component represents ranking engine which applies client specific ranking and applies determines the best edit recommendation amongst multiple edit hits per claim.

### Output Generator

This high-level component represents the output engine which applies client specific format requirements to produce a return file.

### Rule & Configuration

This component represents a collection of configuration tables that are published to HDFS and leveraged with Prepay 2.0. It includes:

* Client Ranking Configuration
* Edit Rule & Regulation Data
* Exclusion Rule Data

### Approval (Future)

This high-level component represents the ability to allow for validation of findings. This is a future component.

# Physical architecture

## Diagram

## Components

# Data Architecture

## Diagram

### Original Design



### Current State



## Data structure

### Application Data

Claim & Line

Provider

Member

### Configuration, Rule, & Reference Data Tables

• piportal\_claim\_condition\_code

• piportal\_clia\_certification

• piportal\_clia\_certification\_type

• piportal\_clia\_facility

• piportal\_clia\_number

• piportal\_cob\_elgf

• piportal\_cob\_eligibility

• piportal\_cob\_fusp

• piportal\_edit\_rule

• piportal\_edit\_rule\_legacy

• piportal\_exclusion\_rule

• piportal\_mcd\_mue\_pra

• piportal\_nhpri

• piportal\_npi

• piportal\_nucc\_taxonomy

• piportal\_optum\_icd10cm\_base

• piportal\_optum\_icd10pcs\_base

• piportal\_payor

• piportal\_payor\_lob

• piportal\_payor\_spp\_hierarchy

• piportal\_place\_of\_service

• piportal\_prepay\_edit\_dependency

• piportal\_prepay\_procedure

• piportal\_provider\_siu\_type

• piportal\_ranking

• piportal\_regulation

• piportal\_regulationtest

• piportal\_service\_code

• piportal\_service\_tos

• piportal\_service\_tos\_group

• piportal\_taxonomy\_client\_crosswalk

• piportal\_type\_of\_bill

• piportal\_type\_of\_bill\_legacy

# Security