Virgin Trains West Coast – Single Customer View (SCV)

Data Quality Matching Issues – Change Request

Commercial in confidence

Author:

Russell Jackson – Solution Architect

Confidential

## Table of Contents

[Table of Contents 2](#_Toc527960332)

[**Document** **Management** 3](#_Toc527960333)

[1. Document Purpose 4](#_Toc527960334)

[2. Overview 4](#_Toc527960335)

[3. Solution Overview 5](#_Toc527960336)

[4. Functional Approach 5](#_Toc527960337)

[4.1. Functional Approach – ReferenceMatchValueCount 5](#_Toc527960338)

[4.2. Functional Approach – Validation 6](#_Toc527960339)

[5. Consideration, constraints and dependencies 7](#_Toc527960340)

## **Document** **Management**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version Control** | | |  |
| **Version** | **Date** | **Author(s)** | **Section Changed** |
| 0.1 | 18/10/2018 | Russell Jackson | Initial Draft |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Review** | | |
| **Version** | **Date** | **Review By** |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Supporting Documents** | | |
| **Document Name** | **Document Purpose** | **Document Link** |
| Merkle VTWC Technical Design | Technical Specification | [Click here](https://teams.microsoft.com/_#/vsd/viewer/teams/https%3A~2F~2Fmerkleinc.sharepoint.com~2Fsites~2FVirginTrainsWestCoast-CRMProject~2FShared%20Documents~2FGeneral~2FDevelopment%20Documentation~2FDesign%20Documentation~2FTechnical%20Overview~2FVTWC%20-Process%20Overview%20v0.1.vsd?thread) |
| Address Matching, Merging and Cleansing - VTWCv2 | Address matching, merging and cleansing specification | [Click here](https://teams.microsoft.com/_#/docx/viewer/teams/https%3A~2F~2Fmerkleinc.sharepoint.com~2Fsites~2FVirginTrainsWestCoast-CRMProject~2FShared%20Documents~2FGeneral~2FDevelopment%20Documentation~2FDesign%20Documentation~2FTechnical%20Overview~2FAddress%20Matching%2C%20Merging%20and%20Cle) |

## Document Purpose

The purpose of this document is to provide details on the issues the Pheonix solution is experiencing and the mitigating actions required to overcome those issues.

## Overview

The Phoenix solution receives majority of its data from the Train Line (TL).

The feed contains customer, transaction and journey attributes, and is loaded daily.

The inbound data is received by the solution where the ingestion process attempts to match the data to existing customers/individuals in the database, using the matching logic defined in this [document](https://teams.microsoft.com/_#/docx/viewer/teams/https%3A~2F~2Fmerkleinc.sharepoint.com~2Fsites~2FVirginTrainsWestCoast-CRMProject~2FShared%20Documents~2FGeneral~2FDevelopment%20Documentation~2FDesign%20Documentation~2FTechnical%20Overview~2FAddress%20Matching%2C%20Merging%20and%20Cle).

Due to the TL supplying poor-quality information, and the matching logic being rudimental, the solution is over matching i.e. matching inbound records to existing customer/individuals incorrectly.

The solution attempts to match on TSCCustomerID (Trainline only), then email, mobile number and finally on a match key made up of initial, surname, postcode and first line of the address.  The issue arises when TL supply repetitive/dummy values for ‘guest customers’ (customer who have not created a membership).

The following provides example records for those guest customers, please note Address1 and Postcode fields;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TSCCustomerID | Title | Forename | Surname | Email | Mobile Number | Address 1 | Postcode |
| 1235 | Mr | Russell | Jackson | [Russell.Jackson@Merkleinc.com](mailto:Russell.Jackson@Merkleinc.com) | 79162747472 | Incomplete | Incomplete |
| 5531 | Mr | James | Moore | [James.Moore@merkleinc.com](mailto:James.Moore@merkleinc.com) | 79162747777 | Incomplete | Incomplete |
| 9931 | Mr | Rodger | Jackson | [bigeat@burgers.com](mailto:bigeat@burgers.com) | 122121222 | Incomplete | Incomplete |

The current process will attempt to match on TSCCustomerID, then email and then mobile, but will be unsuccessful as the values are different.  The matching process will then attempt to match on Namadd (initial, surname, postcode and first line of the address), and will successfully (but incorrectly) match Russell and Rodger, due to the match Namadd match key being the same, although they’re two distinct individuals.

The functional approach detailed in this document addresses this issue.

## Solution Overview

This section provides a written description of how Merkle intends to overcome this issue.

Step 1. Create/populate a reference table counting all email, mobile numbers and Namadd (address1 and postcode separately) values, called reference.MatchValueCount from the relevant [Staging] tables

Step 2. The matching process will validate whether the inbound data matches any of the values in the reference.MatchValueCount table (above) and if the count is greater than *n,* the inbound value will not be used in the matching process.

## Functional Approach

The following section outlines how, from a functional perspective, the Phoenix solution will overcome this issue. The following section will be broken down into two parts;

1. Steps to create a list of values and a count of how many times those values exist in the solution
2. Steps to validate inbound information against those values which have a count greater than *n.*

## Functional Approach – ReferenceMatchValueCount

|  |  |
| --- | --- |
| **Ref** | **Description** |
| REF-001 | Create a table in the CRM database named; reference.MatchValueCount |
| REF-002 | The reference.MatchValueCount table will be structured as per the following ;   |  |  | | --- | --- | | Name | Type | | FieldValue | Varchar (200) | | AddressType | Varchar (20) | | Count | Int | |
| REF-003 | Populate the table reference.MatchValueCount with the following business logic  Group count;  1. All AddressLine1 and PostalCode values from the [CRM].[Staging].[STG\_Address] table and insert the results in the reference.MatchValueCount table. Update the Addrestype to ‘Namadd’  2. All Address values from the [CRM].[Staging].[STG\_ElectronicAddress] where AddressTypeID <> and insert the results in the reference.MatchValueCount table. Update the Addrestype to either ‘Mobile’ where [CRM].[Staging].[STG\_ElectronicAddress].AddressTypeID = 4 and ‘Email’ where [CRM].[Staging].[STG\_ElectronicAddress].AddressTypeID = 3 |
| REF-004 | Create a schedule to populate the reference.MatchValueCount table on a daily basis as the following time;  This step will be governed by a SQL Service Agent job, named; UpdateMatchValueCount |

## Functional Approach – Validation

|  |  |
| --- | --- |
| **Ref** | **Description** |
| REF-001 | Update the following stored Procedure; Preprocessing.tocplus\_customer\_process  On inbound match, validate whether the inbound values exist in the reference.MatchValueCount table with a count less than *n.* If the count is equal to or greater than *n* ignore the value as a possible match value. |
| REF-002 | Add parameters to the Preprocessing.tocplus\_customer\_process stored procedure so threshold can be dynamically updated when there is a change. |

## Consideration, constraints and dependencies

|  |  |
| --- | --- |
| **Ref** | **Description** |
| CCD-001 | The approach detailed in this document will support the issue surrounding repetitive mobile numbers such as 0121212121 |
| CCD-002 | The effort/costs to deliver the Change Request has been supplied by Dhana |
| CCD-003 | Thresholds need to be agreed by client and Merkle client partner |