



Connected Identity




User Guide

Release 4.9.0
B035-0000-9712
March 2024

Copyrights or Trademarks

All copyrights and trademarks used in Teradata documentation are the property of their respective owners. For more information, see [Section : “Teradata Trademark and Trademark Attributions.”](#)

Product Safety

Safety Type	Description
	Indicates a situation which, if not avoided, could result in damage to property, such as to equipment or data, but not related to personal injury.
	Indicates a hazardous situation which, if not avoided, could result in minor or moderate personal injury.
	Indicates a hazardous situation which, if not avoided, could result in death or serious personal injury.

Warranty Disclaimer

Except as may be provided in a separate written agreement with Teradata or required by applicable law, the information available from the Teradata Documentation website or contained in Teradata information products is provided on an "as-is" basis, without warranty of any kind, either express or implied, including the implied warranties of merchantability, fitness for a particular purpose, or noninfringement.

The information available from the Teradata Documentation website or contained in Teradata information products may contain references or cross-references to features, functions, products, or services that are not announced or available in your country. Such references do not imply that Teradata Corporation intends to announce such features, functions, products, or services in your country. Please consult your local Teradata Corporation representative for those features, functions, products, or services available in your country.

The information available from the Teradata Documentation website or contained in Teradata information products may be changed or updated by Teradata at any time without notice. Teradata may also make changes in the products or services described in this information at any time without notice.

Feedback

To maintain the quality of our products and services, e-mail your comments on the accuracy, clarity, organization, and value of this document to: docs@teradata.com.

Any comments or materials (collectively referred to as "Feedback") sent to Teradata Corporation will be deemed nonconfidential. Without any payment or other obligation of any kind and without any restriction of any kind, Teradata and its affiliates are hereby free to (1) reproduce, distribute, provide access to, publish, transmit, publicly display, publicly perform, and create derivative works of, the Feedback, (2) use any ideas, concepts, know-how, and techniques contained in such Feedback for any purpose whatsoever, including developing, manufacturing, and marketing products and services incorporating the Feedback, and (3) authorize others to do any or all of the above.

Teradata Trademark and Trademark Attributions

Teradata, BYNET, Claraview, Covalent, DecisionCast, IntelliBase, IntelliCloud, IntelliFlex, IntelliSphere, nPath, QueryGrid, SQL-MapReduce, Stacki, "Teradata" logo, Teradata Analytics Platform, Teradata Decision Experts, "Teradata Labs" logo, Teradata ServiceConnect, and Teradata Vantage are trademarks or registered trademarks of Teradata Corporation or its affiliates in the United States and other countries.

Adaptec and SCSISelect are trademarks or registered trademarks of Adaptec, Inc.

Amazon Web Services, AWS, Amazon Elastic Compute Cloud, Amazon EC2, Amazon Simple Storage Service, Amazon S3, AWS CloudFormation, and AWS Marketplace are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.

AMD Opteron and Opteron are trademarks of Advanced Micro Devices, Inc.

Apache, Apache Avro, Apache Hadoop, Apache Hive, Hadoop, and the yellow elephant logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries.

Apple, Mac, and OS X all are registered trademarks of Apple Inc.

Axeda is a registered trademark of Axeda Corporation. Axeda Agents, Axeda Applications, Axeda Policy Manager, Axeda Enterprise, Axeda Access, Axeda Software Management, Axeda Service, Axeda ServiceLink, and Firewall-Friendly are trademarks and Maximum Results and Maximum Support are servicemarks of Axeda Corporation.

CENTOS is a trademark of Red Hat, Inc., registered in the U.S. and other countries.

Cloudera and CDH are trademarks or registered trademarks of Cloudera Inc. in the United States, and in jurisdictions throughout the world.

Data Domain, EMC, PowerPath, SRDF, and Symmetrix are either registered trademarks or trademarks of EMC Corporation in the United States and/or other countries.

GoldenGate is a trademark of Oracle.

Hewlett-Packard and HP are registered trademarks of Hewlett-Packard Company.

Hortonworks, the Hortonworks logo and other Hortonworks trademarks are trademarks of Hortonworks Inc. in the United States and other countries.

Intel, Pentium, and XEON are registered trademarks of Intel Corporation.

IBM, CICS, RACF, Tivoli, IBM Spectrum Protect, and z/OS are trademarks or registered trademarks of International Business Machines Corporation.

Linux is a registered trademark of Linus Torvalds.

LSI is a registered trademark of LSI Corporation.

Microsoft, Azure, Active Directory, Windows, Windows NT, and Windows Server are registered trademarks of Microsoft Corporation in the United States and other countries.

NetVault is a trademark of Quest Software, Inc.

Novell and SUSE are registered trademarks of Novell, Inc., in the United States and other countries.

Oracle, OpenJDK, Java, and Solaris are trademarks or registered trademarks of Oracle and/or its affiliates.

QLogic and SANbox are trademarks or registered trademarks of QLogic Corporation.

Quantum and the Quantum logo are trademarks of Quantum Corporation, registered in the U.S.A. and other countries.

Red Hat is a trademark of Red Hat, Inc., registered in the U.S. and other countries. Used under license.

SAP is the trademark or registered trademark of SAP AG in Germany and in several other countries.

SAS and SAS/C are trademarks or registered trademarks of SAS Institute Inc.

Sentinel® is a registered trademark of SafeNet, Inc.

Simba, the Simba logo, SimbaEngine, SimbaEngine C/S, SimbaExpress and SimbaLib are registered trademarks of Simba Technologies Inc.

SPARC is a registered trademark of SPARC International, Inc.

Unicode and the Unicode logo are registered trademarks of Unicode, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Veritas, the Veritas Logo and NetBackup are trademarks or registered trademarks of Veritas Technologies LLC or its affiliates in the U.S. and other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

Purpose

Welcome to Teradata's Master Data Management (MDM) Product. Teradata's Master Data Management product can be thought as the set of methods or processes and procedures used to manage, reference, and synchronize correct and consistent Master Data across an Enterprise. The Teradata MDM product can provide the user with the capability to manage, integrate and consolidate Master Data with or without having to replace existing systems. Master Data can be defined as Data that is important to the company, may be referenced in transactional data, and changes over time (hence must be managed), and is needed in multiple enterprise systems by multiple users.

Teradata's MDM product provides the business users with the capability to create and manage the data. Teradata's MDM provides the ability to create flexible business workflows that accurately reflect the specific business needs of the customer and to provide this accurate and consistent information to anyone in the enterprise. Teradata's MDM is built upon an open architecture known as Service Oriented Architecture (SOA) and the Teradata platform itself, which provides the necessary performance, scalability, and reliability attributes of a World Class Master Data Management product.

With the Teradata's MDM solution you can stage, consolidate, validate, cleanse, store, augment, cross-reference, and publish data to systems in and across your enterprise. By ensuring cross-system data consistency, Teradata's Master Data Management can enable flawless and timely execution of business processes – while leveraging existing investments and reducing the total cost of ownership to manage business critical data.

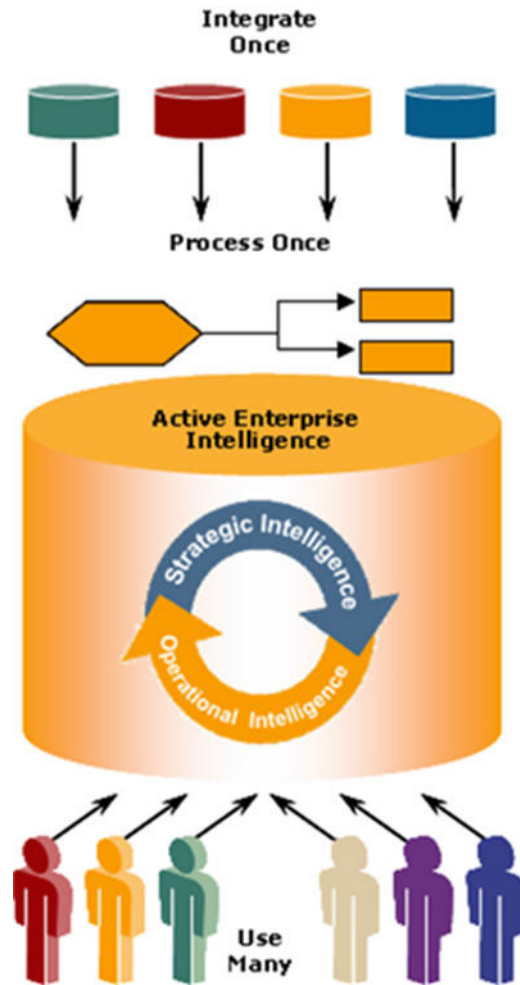
Teradata's MDM provides a framework for building business workflows by defining the necessary data in workflows via X-Docs and expressing the business logic that operates on the data via X-Rules. This framework enables the definition of business process workflows in a configurable manner with little programming effort.

Topics:

- [About Teradata's Master Data Management](#)
- [About This Book](#)
- [Related Documentation](#)
- [Customer Support](#)
- [Documentation Feedback](#)

About Teradata's Master Data Management

Teradata's Master Data Management (MDM) helps you manage key data elements within the Enterprise Data Warehouse (EDW), within an Active Data Warehouse (ADW), as well as, across various enterprise systems and geographies.



Teradata's MDM enables this with the following high-level key features:

- Data Staging for Loading, Cleansing, and Validation.
- Master Data Life-Cycle Manager for Data Maintenance and Business Specific Workflows and Processes.
- Publishing, Versioning, and Cross Referencing of Master Data.
- Data Model Extensibility and Integration with the Enterprise Data Model.
- Hierarchy Management.

About This Book

This document describes how to set up the CDI and is designed to help you understand CDI and provides detailed description of the user interface, features, and procedures associated with specific business processes.

Target Audience

This document is intended for System Administrators and Technical Support personnel responsible for configuring the Master Data Management application and for MDM application users.

What You Should Know

This document assumes that you have prior experience in working on Unix platforms: Solaris/HPUX/AIX, or the Windows 2003 server platforms, as applicable. It also assumes that you have a basic knowledge about creating code values, data load workflow, table editors, and managing hierarchies in MDM.

Document Structure

This book contains the following sections:

- [Chapter 1: “Introduction.”](#) provides an introduction to MDM’s Connected Identity (CI).
- [Chapter 2: “Connected Identity Deployment.”](#) describes the CI installation steps.
- [Chapter 3: “Connected Identity Data Model.”](#) describes the CI data model details.
- [Chapter 3: “Connected Identity Profiles.”](#) describes the CI cleansing & Standardization, matching and survivorship profiles.
- [Chapter 4: “Customer 360 UI and Dashboard.”](#) describes the customer 360 degree UI and dashboard.
- [Chapter 5: “Connected Identity Web Services.”](#) describes the CI Web services APIs.
- [Chapter 6: “Connected Identity Web Analytics.”](#) describes the CI Web Analytics dashboard.

Changes to This Book

The following changes were made to this book in support of the current release. For a complete list of changes to the product, refer *Master Data Management Release Definition* associated with this release.

Date and Release	Description
April 2019, 4.3.0	New Guide.
November 2019, 4.4.0	Included few additional steps in Chapter 2 CI_Deployment chapter and included Data Lineage section in Chapter 5 Customer 360 UIs and Dashboard.

Date and Release	Description
June 2020, 4.5.0	Updated few changes in Installing Connected Identity Solution in Chapter 2 CI_Deployment chapter and updated Data Lineage section in Chapter 5 Customer 360 UIs and Dashboard.
March 2021, 4.6.0	Screenshots replaced in Chapter 4_CI_Profiles as per the new UI changes.
February 2022, 4.7.0	Installation procedure updated in Chapter 2. Screenshots updated and new topics updated in Chapter 4.
November, 2022, 4.8.0	Updated Chapter 2: Connected Identity Deployment, based on the new Installer. Release Version Updated
March, 2024, 4.9.0	Release Version Updated

Related Documentation

For more information on MDM, refer the following documents:

- *Master Data Management Release Definition*
(Master Data Management 4.9.0 Release Definition.pdf)
- *Master Data Management Server Guide*
(Master Data Management 4.9.0 Server Guide.pdf)
- *Master Data Management Studio User Guide*
(Master Data Management 4.9.0 Studio User Guide.pdf)
- *Master Data Management Developer Guide*
(Master Data Management 4.9.0 Developer User Guide.pdf)
- *Master Data Management Installation Guide*
(Master Data Management 4.9.0 Installation Guide.pdf)
- *Master Data Management Reference Guide*
(Master Data Management 4.9.0 Reference Guide.pdf)

The above Teradata documents are available at: <https://docs.teradata.com>

To Read The Documentation

To read the .pdf files, you must have Adobe Acrobat Reader, version 4.0 or higher. If you do not have Acrobat Reader on your machine, you can download it from Adobe's Web site at <http://www.adobe.com>.

Customer Support

Customer support is available at the Teradata customer support Web site (<https://access.teradata.com>), where you can:

- Request shipment of software.
- Download software documentation.
- Submit new issues or cases.
- Track the status of current issues or cases.

Documentation Feedback

Please share your thoughts and ideas:

- Send feedback to docs@teradata.com.
- Navigate to <https://teradata-documentation.ideas.aha.io/ideas/new> and provide your ideas.

Table of Contents

Purpose	iii
About Teradata's Master Data Management	iv
About This Book	v
Target Audience	v
What You Should Know	v
Document Structure	v
Changes to This Book	v
Related Documentation	vi
To Read The Documentation	vi
Customer Support	vi
Documentation Feedback	vii

Chapter 1: Introduction 1

About Teradata MDM Connected Identity	1
Connected Identity Process Flow	2

Chapter 2: Connected Identity Deployment 6

Introduction	6
Connected Identity Package Structure	6
Installing Connected Identity Solution	8
Pre-Requisites	8
Installation	10
Cleanup Workflow (Optional)	10
Connected Identity UI Login	11

Chapter 3: Connected Identity Profiles 12

Introduction	12
Cleansing & Standardization Profiles	12
Matching Profiles	14

Survivorship Profiles	16
Child Data Population	17
Final Archive and Clean-Up	18
Audit Balance and Check Groups	19
CI Process Orchestration	23
User Security and Reference Codes	23

Chapter 4: Customer 360 UI and Dashboard.....26

Introduction	26
Connected Identity Dashboard	26
Data Ingestion Dashboard	31
Search Customer	34

Chapter 5: Connected Identity Web Services.....43

Introduction	43
Web Service Implementation Details	44
Request Types	44
Search Web Services	51
SearchCustProfile	52
Sample Rest Request/Response	52
GetCustomerAddress	54
Sample Rest Request/Response	54
GetCustomerUAL	55
Sample Rest Request/Response	56
CustManageLyltyAccount	56
Sample Rest Request/Response	57
CustManagePymntAccount	57
Sample Rest Request/Response	58
CustManageTrait	59
Sample Rest Request/Response	59
Admin Web Services	60
..... MktgPrgMaintenanceRequest60	
Sample Rest Request/Response	60
Sample Rest Request/Response	61
Sample Rest Request/Response	62
Sample Rest Request/Response	62
..... ChannelMaintenanceRequest63	
Sample Rest Request/Response	63

Sample Rest Request/Response	64
Sample Rest Request/Response	65
Sample Rest Request/Response	65
.....PromotionMaintenanceRequest	66
Sample Rest Request/Response	66
Sample Rest Request/Response	67
Sample Rest Request/Response	68
Sample Rest Request/Response	68
.....CampaignMaintenanceRequest	69
Sample Rest Request/Response	69
Sample Rest Request/Response	70
Sample Rest Request/Response	71
Sample Rest Request/Response	71
.....UALMaintenanceRequest	72
Sample Rest Request/Response	72
Sample Rest Request/Response	73
Sample Rest Request/Response	74
Sample Rest Request/Response	74
TraitMaintenanceRequest	76
Sample Rest Request/Response	76
Sample Rest Request/Response	77
Sample Rest Request/Response	78
Sample Rest Request/Response	78
Update Web Services	80
Archiving	81
CustManageLyltyAccount	81
Sample Rest Request/Response	81
Sample Rest Request/Response	82
Sample Rest Request/Response	83
CustManagePymntAccount	83
Sample Rest Request/Response	83
Sample Rest Request/Response	84
Sample Rest Request/Response	85
CustManageTrait	85
Sample Rest Request/Response	86
Additional Information	86
Source Code Description	86

Chapter 6: Connected Identity Web Analytics 89

Introduction	89
Connected Identity Product	91
Connected Identity UI Web Metrics	92
Server Side Configurations	99

Access Token or Client Secret files	99
Implementation Details	100
List of Accounts, Property and Profile Information	101
Metrics and Dimensions to Fetch to Database.	101
Database Development.	102

List of Figures

Figure 1: Connected Identity Solution	2
Figure 2: Process Flow of CI Solution	3
Figure 3: Process Flow of CI Solution	4
Figure 4: Connected Identity Package Structure	7
Figure 5: Manage C & S Profiles	13
Figure 6: Matching Profiles	15
Figure 7: Survivorship Workbench	16
Figure 8: Registration of a New Source	20
Figure 9: Configure Job Group.	20
Figure 10: Configure Job Groups	21
Figure 11: Sequence Execution of Job Group	21
Figure 12: ABC: Group Execution Details.	22
Figure 13: Executed Logs of ABC Group	22
Figure 14: CI Process Orchestration Workflow	23
Figure 15: CI User Role Activities.	24
Figure 16: CI Related Code Sets	25
Figure 17: Connected Identity Dashboard	27
Figure 18: Connected Identity Dashboard	27
Figure 19: Golden Customer Records	28
Figure 20: Enrollment by Channel	29
Figure 21: Enrollment Trend	29
Figure 22: Demographics	30
Figure 23: Consent	30
Figure 24: Data Ingestion Dashboard.	31
Figure 25: Data Load	32
Figure 26: Data Load	33
Figure 27: Data Volume by Source System	33
Figure 28: Source Systems	34
Figure 29: Customer Profile	35
Figure 30: Customer 360 View.	36
Figure 31: Customer 360 View—Profile Details	37
Figure 32: Customer 360 View—Demographics	38

Figure 33: Customer 360 View—Campaigns.	39
Figure 34: Customer 360 View—Cross-Reference	40
Figure 35: Customer 360 View—Data Lineage.	41
Figure 36: Customer 360 View—Household	42
Figure 37: CI Web Services	43
Figure 38: Teradata Web Service APIs	44
Figure 39: How Web Site Works	90
Figure 40: Web Analytics Summary	93
Figure 41: Web Analytics Summary—Impression Sources.	94
Figure 42: Web Analytics Details—Engagement and Tiers.	95
Figure 43: Web Analytics Details—Customer Channels and Customer Trends	96
Figure 44: Web Analytics Summary—Demographics.	97
Figure 45: Web Analytics Details.	98
Figure 46: Web Analytics Details.	99
Figure 47: Server Side Configurations	100

List of Tables

Table 1: WS_MDL_MAP API Column Description	45
Table 2: WS_REQ_ATT_MDL_COL_MAP API Column Description	45
Table 3: WS_REQ_TYPE_MAP API Column Description	47
Table 4: WS_VIEW_LINK API Column Description	48
Table 5: Source Code Descriptions	87
Table 6: Source Code Descriptions	87

CHAPTER 1 Introduction

What's In This Chapter

This chapter provides an overview of MDM and its features.

Topics include:

- [About Teradata MDM Connected Identity](#)
- [Connected Identity Process Flow](#)

About Teradata MDM Connected Identity

Connected Identity (CI) is a vital part of getting a 360-degree picture of the customer. Connected Identity is the process of collecting, organizing and synthesizing business and customer data, in order to identify bigger trends and paint a more complete picture of a business' customers. Data about your customers and prospective customers can come from every part of the business. Sales data, viewership data, social listening data, website data, customer service data, and many other data sources. Each data source provide specific part of a customer's experience. By integrating data from more than one data source, you will get a single view of your customer, which help in better informed decision making.

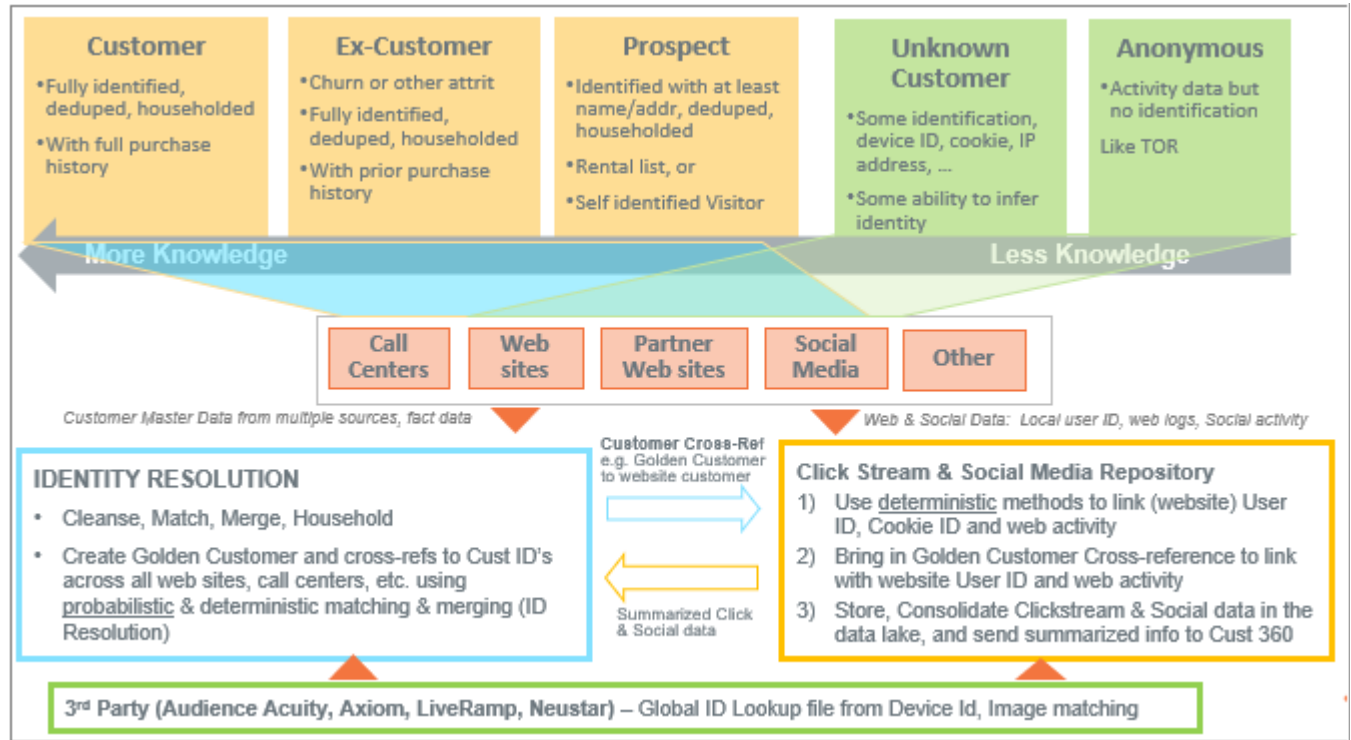
Although many companies have been gathering customer data for a good number of years, it has not always been managed effectively. As a result, companies may maintain outdated, redundant and inconsistent customer data that was gathered via phone calls, emails, websites, web chats, surveys or in-person engagements with customers. The CI policies can help establish order over the data generated by these disparate source systems and can lead to the following benefits:

- Improved sales: accurate customer data enables organizations to better understand customers and focus on personalizing cross-selling and upselling opportunities.
- Better customer service: customer service agents can better understand the entire customer journey when responding to service calls.
- More efficient data management on an ongoing basis: once CI policies and data quality efforts are in place, it becomes easier to update customer records, manage real-time data collection and consolidate data silos.

Teradata MDM offers CI package that provides end to end solution of Customer Data Integration and Management. [Figure 1](#) displays CI solution overview. MDM CI solution captures entire life cycle of customers. Customer data come from all different sources and the CI process match and merge those to create golden customer records. The creation of golden customer record involves Identity Resolution that uses Cleanse, Match, Merge and Household

process of MDM and maintains Cross Reference between Golden Customer ID and IDs of different source system associated with it. The golden customers are also linked to their different click streams and social media interactions associated with customers.

Figure 1: Connected Identity Solution



Connected Identity Process Flow

MDM CI solution is implemented to provide End to End processing of the customer data. The captured Source data can be staged, validated, and brought into the Golden Master table. Source Data can be brought directly into the CI database and the specific CI process flow can initiate there, to attain the Golden Master table.

The basic and most important concept is to understand that all of the various CI rules and profiles are based on a source of data. The Figure 2 and Figure 3 represents the technical process flow of the CI solution.

This process flow diagram represents all of the steps required to attain the Golden Master, many times referred to as the Golden Record.

- **Source:** Any table within the Teradata system to be used as a Source, these tables must be Registered into CI, via the Admin feature "Model Builder". An implementation can have multiple sources of data that contributes to the Golden Master. Each of these sources can be implemented with their own set of CI Rules.

Figure 2: Process Flow of CI Solution

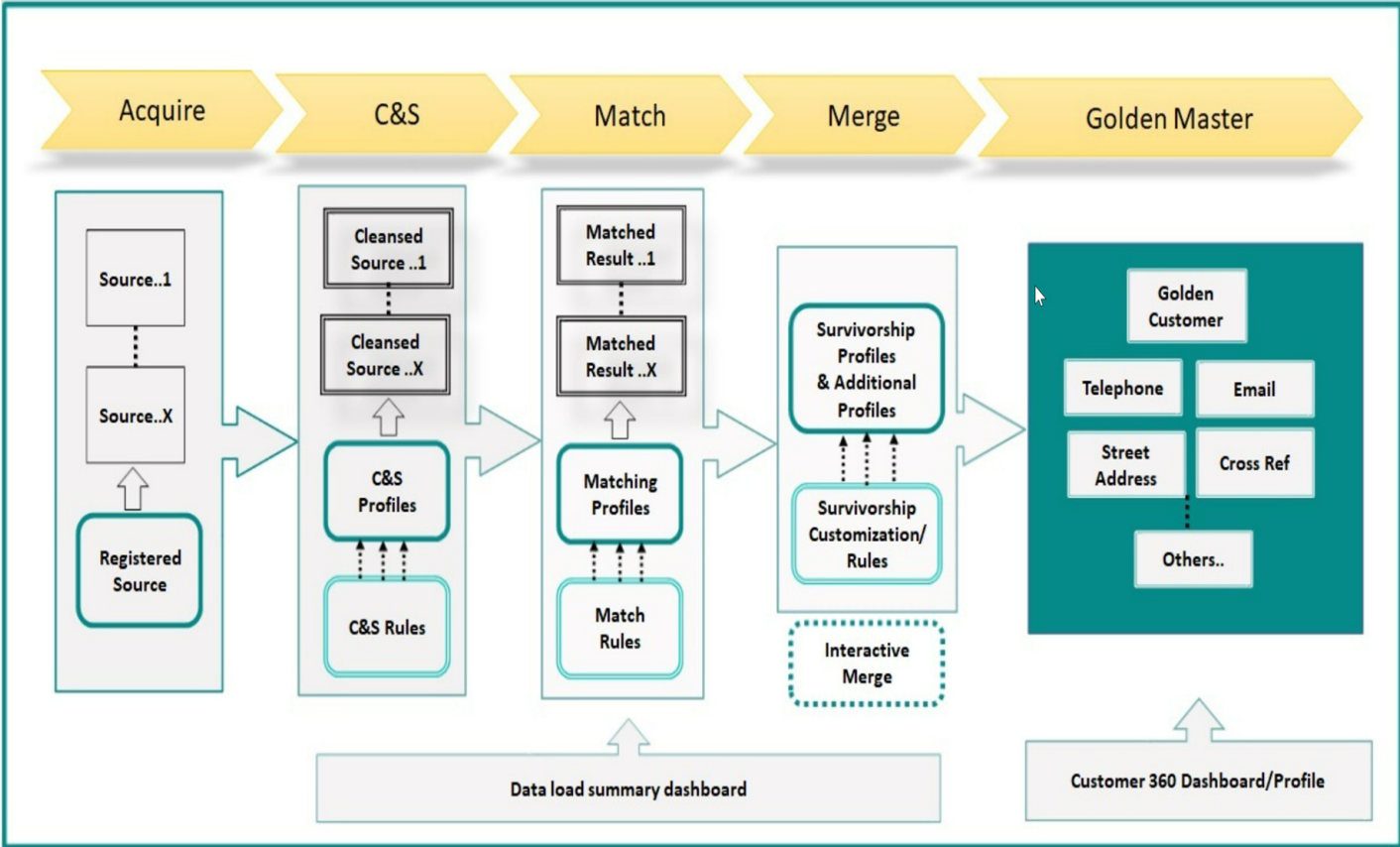
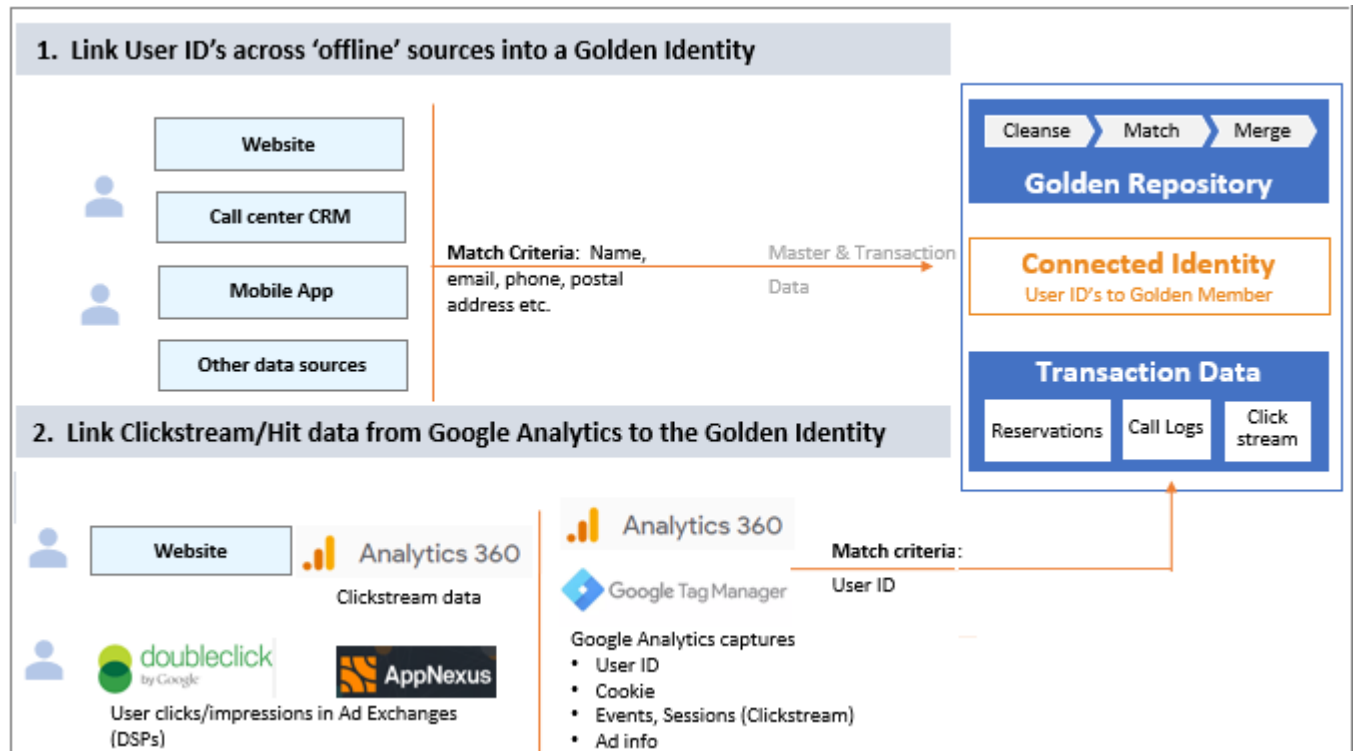


Figure 3: Process Flow of CI Solution



- **Cleansing and Standardization (C&S) Profile:** The CI solution provides per-defined C&S profiles for each Source, which contains Cleansing and Standardization Rules. The C&S rules utilize the MDM Business Rules Editor. When the C&S Profile is executed, the C&S Rules are applied to the source records. Once executed, a C&S Dashboard containing various Key Performance Indicators or Metrics (KPI's) is available for viewing.
- **Matching Profile:** The CI solution provides per-defined Matching profiles for each Source. When the Matching Profile is executed, internal Matching Results tables are populated, based upon the Matching Profile. The Matching Profile compares the Cleaned Source Data to the specified Golden Records.
- **Survivorship Profile:** The CI solution provides per-defined Survivorship profile for each Source. The Survivorship Profile, may contain Customization Rules (IDGEN (Unique Identifier Generation), Lookup (Reference Code usage), Transformation (Changing the value). The Survivorship Profile, may also contain Survivorship Rules (such as Most Recent Record, Trusted Source). Survivorship rules are typically applied in cases where multiple records (in or across Sources) with the same or similar attribute values. When the Survivorship Profile is executed the Cleansed Source Data, Matching Results data, and the current Golden Records are used to create and merge Master Golden Records. Source records that are deemed to be Duplicates do not survive.

MDM provides an out of the box feature called Manage Trust Framework that provides a facility to assign priorities across multiple sources of data for a single attribute.

- **Manage the Golden Master:** interfaces are provided to manage the various Golden Master records over the life cycle of the Golden Master. A Dashboard is provided to view the various KPI's and overall state of the Golden Master. The interface also provides UIs for

360-degree view of Golden customer records and Web Analytic solution which links click streams of customers in websites/social media with their 360 views.

CHAPTER 2 **Connected Identity Deployment**

What's In This Chapter

This chapter provides guidelines for installing CI solution.

Topics include:

- [Introduction](#)
- [Connected Identity Package Structure](#)
- [Installing Connected Identity Solution](#)

Introduction

The Connected Identity (CI) Solution is built as a Custom Application of MDM which needs to be installed on top of base MDM. The CI solution is available as a TAR file. The Package of CI contains an extensive Data Model to store customer data with a set of pre-defined C&S, Matching and Survivorship profiles and some wrapper workflows to orchestrate the profile executions for movement of data from source to master areas of underlying data model. In addition to this, the package also contains: additional static data for User Security & Reference Codes, UIs for 360-degree view of Golden customer records, Web Service APIs to access Customer data from External Systems, Web Analytic solution which links click streams of customers in websites/social media with their 360 views.

Connected Identity Package Structure

The [Figure 4](#) displays the CI package Structure. The CI package contains the following:

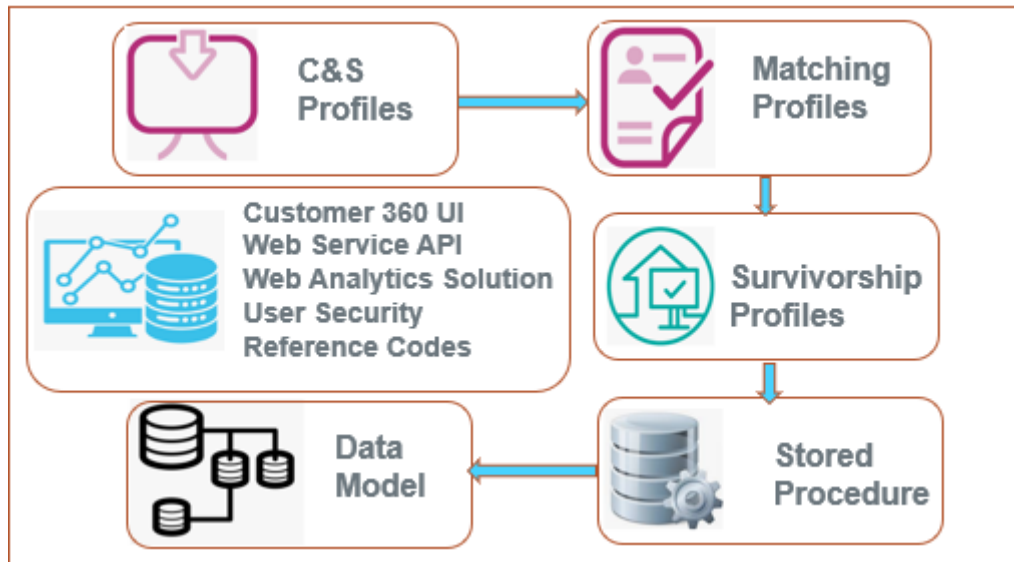
Workflows or Profiles

The CI package contains a set of profiles or workflows that needs to be run in a pre-defined sequence to match or merge customer data coming in from different sources to create new records and merge existing golden records. Workflows are created to run the profiles in a pre-defined sequence.

Static Data

The CI package also contains static reference data for different code sets used in the data model. It also provides user security related static data that handles CI specific user interface authentication. CI specific user role will have the following CI activities enabled (ci-WebAnalyticsSummaryPage, ci-WebAnalyticsDetailPage, ci-SearchCustomerPage, ci-DataIngestionDashboardPage, ci-CustomerHubDashboardPage and CI Specific Activity).

Figure 4: Connected Identity Package Structure



Data Model

The CI solution provides an out of the Box Data Model that can be used by the implementation. This data model is built or created by reviewing and analyzing many different Customer reference implementations and many industry logical data models. The CI solution also provide the capability to extend the deployed data model, as well as, create and manage various pertinent reference codes, such as Demographics. The CI data model has two different staging areas: Source and Master. The Source staging area contains data from all different sources and these data are moved to Master staging area to be populated under various logical groups through different CI processes.

Customer 360 UI

These UIs are built on top of CI data model. These UIs provide 360 view of entire customer portfolio. In addition to this, there are Dashboards available to view the Data Ingestion details and Customer enrollment trend.

Web Service APIs

The Web service APIs are developed on CI model to enable External Systems (For Example: Call center) to interact with Customer Data stored in CI systems. Web services created for this implementation is based on MDM framework rules. The different types of web services requests include search/view request and Add, Update and Delete request types. In case of search /view requests, the clients can retrieve data by providing search parameters. In case of Add, Update and Delete request types, client admins can add, update or delete data of admin tables by passing required parameters per web service.

Web Analytics Solution

The Web analytics solution tracks user behavior across a spectrum of known and unknown entities. Connected Identity ingests data from web analytics and social media platforms to track user interactions using a cross reference of known and unknown identities. Connected

identity provides the viewpoint on every interaction which a user might have with online and offline data.

Installing Connected Identity Solution

Installation of CI solution involves the following main steps:

- [Pre-Requisites](#)
- [Installation](#)
- [Demo Data Load](#)
- [End to End Workflow Execution](#)
- [Connected Identity UI Login](#)

Pre-Requisites

Before performing the CI installation, the following steps needs to be completed.

- 1 Install base MDM.

For details on installing MDM, refer to *Master Data Management Installation Guide.pdf*.

- 2 The following UDFs must be installed for the cleansing profiles to work:

For details on installing UDFs, refer to section “*Installing Out of the Box Match Attribute Functions*” in *Chapter: RDM Processes in Master Data Management Server Guide.pdf*

- UF_ADDRSIM
- UF_DOUBLEMETAPHONE
- UF_DOUBLEMETAPHONE_P75
- UF_DROPALPHA
- UF_DROPNONALPNUM
- UF_DROPNUMBER
- UF_DTEDIF1
- UF_DTEDIF2
- UF_EDITDISTANCE_U
- UF_EDITDISTANCE_U_OPT
- UF_EXDA
- UF_EXMO
- UF_EXYR
- UF_FORMATNUMBER
- UF_FORMATOUTOFCOUNTRY
- UF_FRST2CHAR
- UF_FRST3CHAR

- UF_FRSTCHAR
 - UF_GENDER
 - UF_GETCARRIER
 - UF_GETCARRIER_US
 - UF_GETLOCATION
 - UF_GETLOCATION_US
 - UF_GETNAME
 - UF_GETTALLNUMINFO
 - UF_GETTALLNUMINFODELIMITED
 - UF_ISVALIDPHONE
 - UF_ISVALIDPHONE_US
 - UF_JAROC
 - UF_JARIDX
 - UF_METAPHONE_P75
 - UF_METAPHONEMDM
 - UF_NAME_ARY
 - UF_NICKNAME
 - UF_SDX,UF_SNDX
 - UF_SNDX3
 - UF_SQUEEZE
 - UF_SQUISH
 - UF_STD_PHONE_E164
 - UF_STD_PHONE_INTERNATIONAL_US
 - UF_STD_PHONE_ISO_TO_US
 - UF_STD_PHONE_NATIONAL_US
 - UF_STRINGCOMP
 - UF_STRSIM,UF_STRSIM2
 - UF_STRSIM3
 - UF_STRSIM4
 - UF_VALIDEMAIL
 - UF_VALIDPOSTCODE_US
- 3 The user should manually provide access or execution permissions for all files inside <MDM_Install_Location>/custom and <MDM_Install_Location>/web folders in case of Linux environment.



In MDM 4.7, Staging tables creates under the Master table database.

Installation

Perform the following steps for CI installation:

- 1 Extract the custom application **CI_indep_noarch.04.09.00.00.tar.gz** file in MDM Installation folder available at location <MDM_Installation_Location>\
A copy of CI Custom Application creates in MDM Installation folder
- 2 Run below commands from Linux terminal/Windows git bash.

```
cd MDM4.9.0.0
tar -zxvf CI_indep_noarch.04.09.00.00.tar.gz
cd MDM4.9.0.0/custom/cdh/install
sh./installCI.sh
```

- 3 To update the Metadata configuration., execute ciSetupMetaConf.bat\sh file.

Collapsed Setup

- a Modify <MDM_Install_Location>\web\mdmclient\WEB-INF\classes\x2.properties file (Add CDHServices.xml to xcore.xservices variable)
- b Set JAVA_HOME in mkcustomcolocjar.bat file available at:
<MDM_Install_Location>web\mdmclient\bin\custom\CDH
- c Run
<MDM_Install_Location>web\mdmclient\bin\custom\CDH\mkcustomcolocjar.bat
(or.sh)
- d Start Application Server

Colocated Setup

- a Start MDM Server from <MDM_Install_Location>\custom\cdh\bin\startServer.bat(or .sh)
 - b Start Application server.
 - c Start services.
- 4 To upload CI configuration, execute ciProfilesImport.bat\sh file.
 - 5 To create views for Web services and Web analytics Setup, Demo data load, End to end workflow execution and Web Analytics move data to target, execute webAnalyticsServiceSetup.bat\sh file.

Cleanup Workflow (Optional)

Run workflow CleanUpDemoData to clear data from the master tables populated by the execution of CI workflow.

Connected Identity UI Login

Perform the following steps to login to CI user interface:

- Login to the Connected Identity UI with username as “ciuser” and password as “ciuser”.

Note: To enable i18nization on Angular UI's in CI, check *Master Data Management Server Guide*.

CHAPTER 3 Connected Identity Profiles

What's In This Chapter

This chapter provides information on out of box CI profiles.

Topics include:

- [Introduction](#)
- [Cleansing & Standardization Profiles](#)
- [Matching Profiles](#)
- [Survivorship Profiles](#)

Introduction

The Connected Identity (CI) solution contains pre-packaged profiles of Cleansing & Standardization (C & S), Matching and Survivorship processes. These profiles are configured for movement of data from Source staging tables to Target Master tables. Profiles are created to match and merge data coming in from multiple sources. A set of sample data is also packaged along with the solution for testing and demo purpose. Sample data available in file system can be loaded to staging tables using the load scripts packaged within the solution. You will be able to customize exiting profiles to fit your requirement. New profiles can also be added for additional requirements. The below sections describes the profiles.

The CI solution has a set of pre-defined workflows that executes the C&S, Matching, Survivorship profiles along with some additional steps for Child data preparation & Archival process.

Cleansing & Standardization Profiles

The C & S profiles are considered as the collective set of business data rules that apply to the data domain. Cleansing processes can be considered as modifying the data value to correct data irregularities. Standardization are basically the rules that manage the consistency of the data for consumers of the data. Standardization are considered to be a class of data rules that change the incoming data for various reasons, such as formats, lengths, valid values.

The C&S profiles are added as part of OOB CI package to Cleanse, Standardize and Validate incoming customer data sitting in source staging area of CI data model. CI package provides the following different types of C&S profiles:

- **Personal Validation profiles:** this type of profile validates date of birth (like incoming age should be less than 120 Yrs) and Gender Code of Customers.
- **Electronic Address Validation profiles:** this type of profile validates the format of email ID of customers.
- **Street Address Standardization profiles:** this type of profile validates the zip code or postal code as per US zip code format.
- **Telephone standardization:** this type of profile validates the telephone numbers of customer as per US telephone standardization. The standardization rule converts the telephone number to E165 format.

The [Figure 5](#) displays the profiles available on the Manage C & S Profiles UI. You can select any existing profile and execute and view execution details, modify the profile details and delete the profile. You can also create new C & S profiles as per specific needs. For details on creating profiles and other activities that can be performed on Manage C & S Profile UI, *refer to Master Data Management Server Guide*.

Figure 5: Manage C & S Profiles

Cleansing and standardization				Q	IMPORT	EXPORT	CREATE NEW
<input type="checkbox"/>	Name	Description	Profile type	Last modified date ↓			
<input type="checkbox"/>	StreetCleansingStandardization	Street Cleansing and Standardization	INTERNAL	Jan 11, 2022, 9:05:05 AM			
<input type="checkbox"/>	PhoneCleansingStandardization	Phone Cleansing and Standardization	INTERNAL	Jan 11, 2022, 9:05:04 AM			
<input type="checkbox"/>	PersonaCleansingStandardization	Persona Cleansing and Standardization	INTERNAL	Jan 11, 2022, 9:05:02 AM			
<input type="checkbox"/>	EmailCleansingStandardization	Email Cleansing Standardization	INTERNAL	Jan 11, 2022, 9:05:00 AM			

Items per page: 25 1 – 4 of 4

Out of the Box C & S Profile Description

- **Profile Name:** PersonaCleansingStandardization

Description: this profile defines validation rules on persona staging. It defines two types of validation. One for gender type code and another for date of birth like age validation as the incoming age of the person to be less than 120 yrs.

- **Profile Name:** EmailCleansingStandardization

Description: this profile defines validation rules on electronic email address. The following email validation will be performed as part this profile execution:

- Email addresses consist of a local part, the "@" symbol, and the domain.
- Text can contain alphabetic, numeric, and these symbols: !#\$%'+-./=?^_`{|}~
- A domain consists of labels separated with periods. No period can start or end a domain name. No two periods in succession can be in a domain name.
- A label may contain hyphens, but no two hyphens in a row. A label must not start nor end with a hyphen.

- The maximum length of a email address is 255 characters.
- Email will be validated for its syntax.
- **Profile Name:** StreetCleansingStandardization
Description: this profile defines validation rules on zip code or postal code and validates as per US zip code format.
- **Profile Name:** PhoneCleansingStandardization
Description: this profile defines validation rules for standardizing the telephone numbers as per US telephone standards. The standardization rule converts the telephone number to E165 format.

Matching Profiles

Matching process refers to the process of identifying matching Primary and secondary source(s) data records based on the predefined set of attributes and functions and producing score or results to perform the survivorship and/or merge process based on the results. Matching process helps in identifying new records, duplicate or exact match records or cross referencing existing master records with source system records.

Matching profiles are added as part of OOB CI package to match the incoming source data with that of target. OOB package contains data coming in from four different sources (Legacy, Customer Centre, Web1 & Web2). CI package provides the following different types of matching profiles for each source:

- A matching profile for each Telephone, Email and Street address matching.
- A Reconciled match profile to reconcile results of Telephone, Email and Street address matching.
- A match profile to perform Last name and First Name matching of Customers where reconciled match results are provided as partition.

The [Figure 6](#) displays the profiles available on the Matching Profiles UI. You can select any existing profile and execute and view execution details, modify the profile details and delete the profile. You can also create new matching profiles as per specific needs. For details on creating profiles and other activities that can be performed on Matching Profiles UI, *refer to Master Data Management Server Guide*.

Figure 6: Matching Profiles

Matching profile				
				<input type="text"/> IMPORT EXPORT CREATE NEW
<input type="checkbox"/>	Name	Description	Profile type	Last modified date ↓
<input type="checkbox"/>	NameMatching	Name Matching	Match	Jan 11, 2022, 9:07:43 AM
<input type="checkbox"/>	AddressReconciliation	Address Reconciliation	Reconciliation	Jan 11, 2022, 9:07:41 AM
<input type="checkbox"/>	StreetMatching	Street Matching	Match	Jan 11, 2022, 9:07:40 AM
<input type="checkbox"/>	EmailMatching	Email Matching	Match	Jan 11, 2022, 9:07:39 AM
<input type="checkbox"/>	PhoneMatching	Phone Matching	Match	Jan 11, 2022, 9:07:39 AM
				Items per page: 25 1 – 5 of 5

Out of the Box Matching Profile Description

- **Profile Name:** StreetMatching

Description: This profile creates matching records by comparing the source and target records based on the predefined set of matching address attributes and produces score or results to perform the survivorship process based on the match results.

Key Match Attributes: ADDR_LN_1_TXT

- **Profile Name:** AddressReconciliation and StreetMatching

Description: This profile creates matching records by reconciling the matching output from multiple match profiles. This profile takes the reconciled match results of individual match profile of ELEC_ADDR_MTCH_PRFL, STREET_ADDR_MTCH_PRFL and TEL_NUM_MTCH_PRFL as partition and perform Last name and First Name matching of Customers.

Key Match Attributes: PRSNA_ID and CUST_ID

- **Profile Name:** NameMatching

Description: This profile creates matching records by comparing the source and target records based on the predefined set of matching attributes (Customer first and last name) and produces score or results to perform the survivorship process based on the match results.

Key Match Attributes: LAST_NAME and FRST_NAME

- **Profile Name:** EmailMatching

Description: This profile creates matching records by comparing the source and target records based on the predefined set of matching email address attributes and produces score or results to perform the survivorship process based on the match results.

Key Match Attributes: ELECTRNC_ADDR_TXT

- **Profile Name:** PhoneMatching

Description: This profile creates matching records by comparing the source and target records based on the predefined set of matching telephone number attributes and produces score or results to perform the survivorship process based on the match results.

Key Match Attributes: TLPHN_E164_NUM

Survivorship Profiles

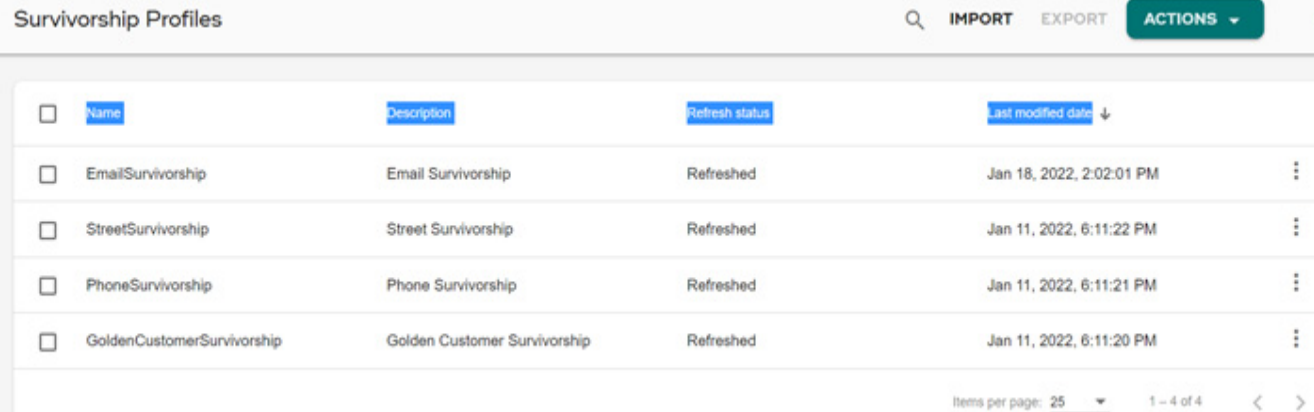
Survivorship process refers to the process of reviewing matching results (scores/weights) and performing merge, duplicate resolution, creating cross reference and creating new records. The Survivorship Profile, may contain Customization Rules and Survivorship Rules (such as Most Recent Record, Trusted Source). Survivorship rules are typically applied in cases where multiple records (in or across Sources) with the same or similar attribute values. When the Survivorship Profile is executed the Cleansed Source Data, Matching Results data, and the current Golden Records are used to create and merge Master Golden Records. Source records that are deemed to be Duplicates do not survive.

Survivorship profiles are added as part of OOB CI package to merge (Create/Update) the matched records. This process creates or merges Golden Customer records as per the results of matching.

- A survivorship profile per source per match profile is provided.
- Survivorship profiles are not created for reconciled match profile as the output of reconcile match profile is given as input to a normal match profile.
- The Row level trust is defined for all Sources to provide priority to data coming in from specific sources.

The [Figure 7](#) displays the profiles available on the Survivorship Workbench UI. You can select any existing profile and execute and view execution details, modify the profile details, view new records and view match records and delete the profile. You can also create new survivorship profiles as per specific needs. For details on creating profiles and other activities that can be performed on Survivorship Workbench UI, *refer to Master Data Management Server Guide*.

Figure 7: Survivorship Workbench



The screenshot shows the 'Survivorship Profiles' interface. At the top, there are buttons for 'IMPORT', 'EXPORT', and 'ACTIONS'. Below is a table with four columns: 'Name', 'Description', 'Refresh status', and 'Last modified date'. The table lists four profiles: 'EmailSurvivorship', 'StreetSurvivorship', 'PhoneSurvivorship', and 'GoldenCustomerSurvivorship'. Each row has a checkbox on the left and a three-dot menu on the right. At the bottom right, there is a pagination control showing 'Items per page: 25' and '1 - 4 of 4'.

<input type="checkbox"/>	Name	Description	Refresh status	Last modified date ↓	
<input type="checkbox"/>	EmailSurvivorship	Email Survivorship	Refreshed	Jan 18, 2022, 2:02:01 PM	⋮
<input type="checkbox"/>	StreetSurvivorship	Street Survivorship	Refreshed	Jan 11, 2022, 6:11:22 PM	⋮
<input type="checkbox"/>	PhoneSurvivorship	Phone Survivorship	Refreshed	Jan 11, 2022, 6:11:21 PM	⋮
<input type="checkbox"/>	GoldenCustomerSurvivorship	Golden Customer Survivorship	Refreshed	Jan 11, 2022, 6:11:20 PM	⋮

Items per page: 25 1 - 4 of 4

Out of the Box Survivorship Profile Description

- **Profile Name:** StreetSurvivorship
Description: This profile reviews the matching results (scores/weights) of StreetMatching and performs the process of merging records, duplicate resolution, creating cross reference and creating new records and populates the data in MST_ADDR and MST_STREET_ADDR tables.
- **Profile Name:** GoldenCustomerSurvivorship
Description: This profile reviews the matching results (scores/weights) of NameMatching and performs the process of merging records, duplicate resolution, creating cross reference and creating new records and populates the data in MST_CUST, MST_INDIV_CUST and MST_ORG_CUST tables.
- **Profile Name:** EmailSurvivorship
Description: This profile reviews the matching results (scores/weights) of EmailMatching and performs the process of merging records, duplicate resolution, creating cross reference and creating new records and populates the data in MST_ELCTRNC_ADDR table.
- **Profile Name:** PhoneSurvivorship
Description: This profile reviews the matching results (scores/weights) of PhoneMatching and performs the process of merging records, duplicate resolution, creating cross reference and creating new records and populates the data in MST_TLPHN_NUM table.

Child Data Population

The master tables of CI data model are populated by matching and survivorship process. In addition to this there are some additional master/transaction tables which are populated by a set of stored procedure based on the golden ID and cross reference created for each customer.

The execution of stored procedure (CI_CHILD_TABLE_POPULATION) populates the following tables:

- MST_CUST_ADDR
- MST_CUST_PRSNA
- MST_LYLTY_PRG
- MST_LYLTY_LVL
- MST_LYLTY_ACCT
- MST_PMT_ACCT
- MST_CUST_PMT_ACCT
- MST_DGTL_IDNTFTN
- MST_GOVN_IDNTFTN
- MST_CUST_TRAT_VAL

- MST_CNTCT_PREFRC
- MST_CUST_UNVRSL_AUTHZTN
- MST_INSIGHTS_SUMMARY
- MST_CUST_MKTG_PRG

The execution of stored procedure (CI_POPULATE_HOUSEHOLD) populates the following tables:

- MST_SOC_GRP_CUST
- MST_CUST_MBRSHIP

The execution of stored procedure (CI_POPULATE_TXN) populates the following tables:

- MST_PURCHASE_SUMMARY
- MST_CAMPAIGN_DETAIL
- MST_CAMPAIGN_DETAIL_CONFIRMATION
- MST_PROMOTN_CUST

Final Archive and Clean-Up

At the end of data load process, when the data is moved to master tables, data from source tables are archived to archive tables and staging table data is cleaned up for next load.

The execution of stored procedure (CI_POPULATE_ARCHIVE) populates the following tables:

- MST_ELCTRNC_ADDR_RQST
- MST_ELCTRNC_ADDR_RESULT
- MST_PRSNA
- MST_PRSNA_ACCT
- MST_PRSNA_ADDR
- MST_PRSNA_CNTCT_PREFRC_TM_PRD
- MST_PRSNA_DGTL_IDNTFTN
- MST_PRSNA_GOVN_IDNTFTN
- MST_PRSNA_LYLTY_ACCT
- MST_PRSNA_PET
- MST_PRSNA_PET_TRAT
- MST_PRSNA_PMT_ACCT
- MST_PRSNA_PROD
- MST_PRSNA_RLTNSHP
- MST_PRSNA_TRAT
- MST_PRSNA_XREF
- MST_STREET_ADDR_RQST

- MST_STREET_ADDR_RESULT
- MST_TLPHN_NUM_RQST
- MST_TLPHN_NUM_RESULT

Audit Balance and Check Groups

The Audit Balance & Check (ABC) model in MDM captures all execution log of Cleansing and Standardization, Matching, Survivorship and any other batch execution performed from MDM server. These groups are created under ABC_GROUP during the MDM installation.

When a C & S, matching or survivorship profile is created either from UI or through Profile Migration Upload process, an ABC job is created in ABC_JOB with Job name same as the profile name and will be linked to the corresponding default group of that module as in the below list.

- DEFAULT_CS—the default ABC group created for C & S profiles.
- DEFAULT_MATCHING—the default ABC group created for Matching profiles.
- DEFAULT_SURVIVORSHIP—the default ABC group created for Survivorship profiles.

When any of the C&S or Matching or Survivorship profile is executed, the execution logs will be captured in ABC_JOB_LOG.

In CI package, the CI solution uses ABC group concept to execute C&S or Matching or Survivorship profiles. The order of execution of these jobs are defined in ABC_GROUP table. As part of the CI Implementation, a list of groups are provided as static data and the order of execution of the jobs within that groups are defined in a mapping table.

The group names provide as part of workflow execution are listed below:

- LEGACY_ALL
- CCENTER_ALL
- WEB1_ALL
- WEB2_ALL

For more details on the ABC Framework implementation, refer to Appendix B ABC Framework in Master Data Management 4.3.0 Server Guide.

New Source Registration

You can register for new Sources to contribute to reference data. Typical examples of sources are Salesforce, WideOrbit, BridgeFiles, TIM etc. RDM Cleansing Database used Source Data to attain the Golden Master table after the RDM specific process flow initiates.

For creating the new source, provide:

- **Source Name**
- Source Intake Id

- Source Description

Figure 8: Registration of a New Source

Create new source

Source Details:

Source name *

Source Intake id *

Source description

Assigned jobs:

+ ADD JOB GROUP

No assigned jobs.

CANCEL

SAVE

Creation of Audit Balance and Check Groups

Once a source is created, we can add multiple Job Groups to it and further configure this group by adding different ABC Jobs to it. These groups are created under ABC_GROUP

Figure 9: Configure Job Group

Configure Job Group:

1 Job group details

2 Configure jobs

3 Sequencer

Job Group Details:

Group name

Web2_All

Group description

Source Intake Id

4

Notification email

CANCEL

SAVE

When a C & S, matching or survivorship profile is created either from UI or through import profile process, an ABC job is created in ABC_JOB with Job name same as the profile name and will be linked to the corresponding default group of that module as in the below list.

- DEFAULT_CS— the default ABC group created for C & S profiles.
- DEFAULT_MATCHING—the default ABC group created for Matching profiles.
- DEFAULT_SURVIVORSHIP—the default ABC group created for Survivorship profiles

These jobs can be added to different ABC groups via the configure jobs tab by selecting the required jobs from the list of jobs shown.

Figure 10: Configure Job Groups

Configure Job Group:

1 Job group details

2 Configure jobs

3 Sequencer

Assign Jobs to be executed:

Cleansing

Matching

Survivorship

Additional

Custom

Select Profiles

<input checked="" type="checkbox"/>	Job id	Job Name	Job Type	Last modified date ↓	
<input checked="" type="checkbox"/>	103	PhoneCleansingStandardization	Cleansing Profile	Jan 11, 2022, 9:14:04 AM	⋮
<input checked="" type="checkbox"/>	104	StreetCleansingStandardization	Cleansing Profile	Jan 11, 2022, 9:14:04 AM	⋮
<input checked="" type="checkbox"/>	102	PersonaCleansingStandardization	Cleansing Profile	Jan 11, 2022, 9:14:04 AM	⋮
<input checked="" type="checkbox"/>	101	EmailCleansingStandardization	Cleansing Profile	Jan 11, 2022, 9:14:04 AM	⋮

Items per page: 25 1 – 4 of 4 < >

CANCEL

SAVE

In CI package, the CI solution uses ABC group concept to execute C&S or Matching or Survivorship profiles. The order of execution of these selected jobs can be defined in the sequencer tab.

Figure 11: Sequence Execution of Job Group

1 Job group details

2 Configure jobs

3 Sequencer

Define sequence of execution:

≡

EmailCleansingStandardization

Cleansing Profile

⌚ 01

≡

StreetCleansingStandardization

Cleansing Profile

⌚ 02

≡

PhoneCleansingStandardization

Cleansing Profile

⌚ 03

≡

PersonaCleansingStandardization

Cleansing Profile

⌚ 04

≡

EmailMatching

Matching Profile

⌚ 05

≡

PhoneMatching

Matching Profile

⌚ 06

≡

StreetMatching

Matching Profile

⌚ 07

As part of the CI Implementation, a list of groups are provided as static data and the order of execution of the jobs within that groups are defined in a mapping table.

The group names provide as part of workflow execution are listed below:

- LEGACY_ALL
- CALLCENTER_ALL
- WEB1_ALL

- WEB2_ALL

ABC Logs

The integration of Audit Balance & Check (ABC) model in MDM provides an enterprise view of logging mechanism for any of MDM OOTB processes and/or custom processes. It increases traceability, data quality, data lineage and exception handling.

The Audit Balance & Check (ABC) model in MDM captures all execution log of Cleansing and Standardization, Matching, Survivorship and any other batch execution performed from MDM server. When any of the C&S or Matching or Survivorship profile is executed, the execution logs will be captured in ABC_JOB_LOG. And when a group is executed, the execution logs will be captured in ABC_GROUP_JOB_LOG. These logs of each group execution can be viewed through Logs UI under Sources.

Figure 12: ABC: Group Execution Details

Audit balance control : Group execution details						
	Group name	Source type name	Workflow instance id	Job group run id	Start Time	End time
✓ 01/11/2022 at 6:13 PM Executed in 2 minutes	Web2_All	WEB2	211301641875852800	2022011044321-0104-00001	Jan 11, 2022, 6:13:21 PM	Jan 11, 2022, 6:13:21 PM
✓ 01/11/2022 at 6:11 PM Executed in 2 minutes	Web1_All	WEB1	211301641875852800	2022011044124-0103-00001	Jan 11, 2022, 6:11:24 PM	Jan 11, 2022, 6:11:24 PM
✓ 01/11/2022 at 6:09 PM Executed in 2 minutes	CallCenter_All	CCENTER	211301641875852800	2022011043929-0102-00001	Jan 11, 2022, 6:09:29 PM	Jan 11, 2022, 6:09:29 PM
✓ 01/11/2022 at 6:07 PM Executed in 2 minutes	Legacy_All	LEGACY	211301641875852800	2022011043735-0101-00001	Jan 11, 2022, 6:07:35 PM	Jan 11, 2022, 6:07:35 PM

Upon clicking any particular group name, it will display the executed logs of each job defined in that group.

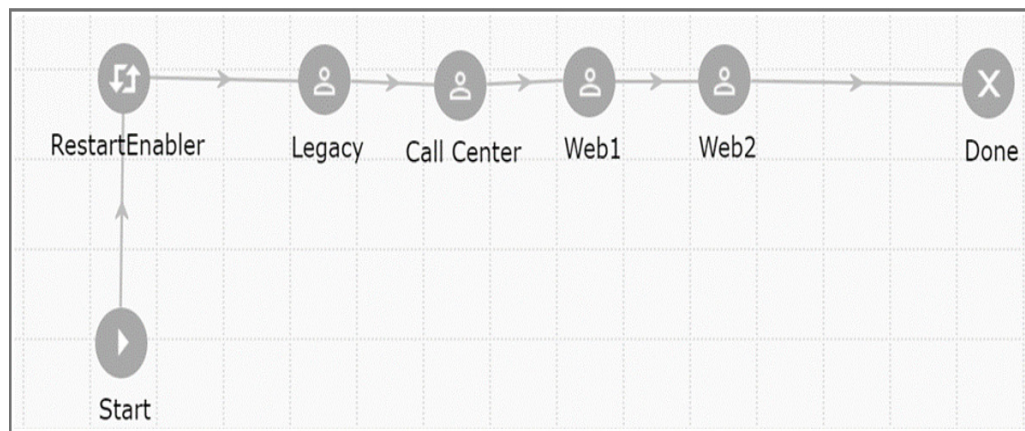
Figure 13: Executed Logs of ABC Group

Web2_All:2022011044321-0104-00001							
	Job name	Source type name	Source record count	Target record count	Error record count	Duplicate record count	Deleted record count
✓ 01/11/2022 at 6:15 PM Executed in a few seconds	ArchiveTablePopulati	WEB2	0	0	0	0	0
✓ 01/11/2022 at 6:14 PM Executed in a few seconds	ChildTablePopulation	WEB2	0	0	0	0	0
✓ 01/11/2022 at 6:14 PM Executed in a few seconds	GoldenCustomerSun	WEB2	993	8634	0	0	0
✓ 01/11/2022 at 6:14 PM Executed in a few seconds	EmailSurvivorship	WEB2	1011	8548	0	0	0
✓ 01/11/2022 at 6:14 PM Executed in a few seconds	StreetSurvivorship	WEB2	916	8836	0	0	0

CI Process Orchestration

A runtime workflow (CIloadAllSources) is defined to orchestrate the entire CI end to end process. The [Figure 14](#) displays the CI process orchestration workflow. The CI process workflow is added to simplify the Customer data load process from staging to master area. Each of the workflow nodes: LEGACY, CCENTER, WEB1, WEB2 within this workflow internally calls a request to execute Cleansing & Standardization, Matching, Survivorship and child table population Stored Procedures associated with that source. At the end of this workflow execution, data from each source tables will be merged to master and Golden customers will be created or merged. This workflow can be executed for initial load or any incremental loads. The workflow can be scheduled as per convenience.

Figure 14: CI Process Orchestration Workflow



Restart Enabler helps to restart the workflow from the fail step.

User Security and Reference Codes

Once the CI installation is completed, it provides user security related static data that handles CI specific user interface authentication. You can login to connected Identity UI with username as “ciuser” and password as “ciuser” and change the password to “ciuser”. By default, the “ciuser” will be assigned to the CI_ROLE and will have the CI related activities added to the role as in [Figure 15](#).

Figure 15: CI User Role Activities

Role Details

* denotes required field

Role Name *

Connected Identity Role

Role Id *

CI_ROLE

☒ Enable Edit Access

Landing Page *

Default Home Page

CANCEL

SAVE AS NEW

UPDATE

WORKFLOW ROLE DETAILS

TABLE ROLE DETAILS

User Activities: Page 1 of 1

☐ Activity Name

×

Q

☐ ci-WebAnalyticsSummaryPage

☐ ci-WebAnalyticsDetailPage

☐ ci-SearchCustomersPage

→

1 - 6 of 6

I<

<

>

>I

REMOVE ACTIVITIES

ADD ACTIVITIES

The CI installation also provides static reference data for different code sets used in the data model as in [Figure 16](#).

Figure 16: CI Related Code Sets

Manage Lookup Data

Groups
Select...

IMPORT

ACTIONS

Search Results: Page 1 of 6

Code Set Id #	Name #	Description #	Associated Tables	Enterprise Records #	Mapped So
<div>=</div> <div>▼</div> <div>X</div> <div>Q</div>	<div>X</div> <div>Q</div>	<div>X</div> <div>Q</div>			
1	Source Systems	Source Systems		1	0
259	Digital Identification Source Cd	Digital Identif...		11	0
331	Currency	Currency		169	0
253	Pet Breed CD	Pet Breed CD		7	0
276	Customer Type Cd	Customer Type C...		8	0
312	Trait Value Type Format Cd	Trait Value Typ...		5	0
291	Reference Domain	Reference Domai...		0	0
290	Persona Role Cd	Persona Role Cd		5	0

→

1 - 20 of 104

I<

<

>

>I

CHAPTER 4 Customer 360 UI and Dashboard

What's In This Chapter

This chapter provides information on out of box CI profiles.

Topics include:

- [Introduction](#)
- [Connected Identity Dashboard](#)
- [Data Ingestion Dashboard](#)
- [Search Customer](#)

Introduction

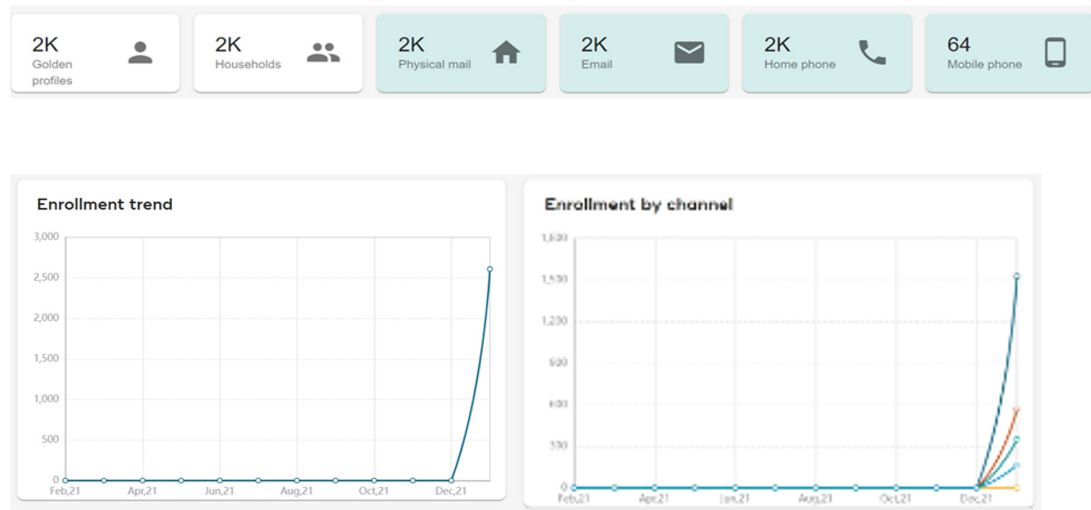
The Connected Identity (CI) dashboard provide end to end display of customer portfolio. It provides a detailed summary of customer statistics, enrollment trends, customer consents and demographics details like persona details, contact information, identification details, traits, brands opted by customers and social media details. It also provides channel wise enrollment trend of customers and volume of customers enrolled over a period of time. The CI also provides an UI to search customer based on various search criteria like ID, Name, Address, City, State or Email. On the search results, click on the row of interested customer to navigate to the Customer 360 view UI. The customer 360 view UI show all aspects of the customer details as listed above.

Connected Identity Dashboard

Connected Identity ingests data from web analytics and social media platforms to track user interactions using a cross reference of known and unknown identities. Connected identity provides the viewpoint on every interaction which a user might have with online and offline data.

The CI dashboard summarizes CI customer statistics and displays the following panels showcasing the different enrollment trends of the customer. By default, the dashboard displays the customer details spanned for a period of one year. You can use the Time frame option to display customer details either for year or 6months or 3 months.

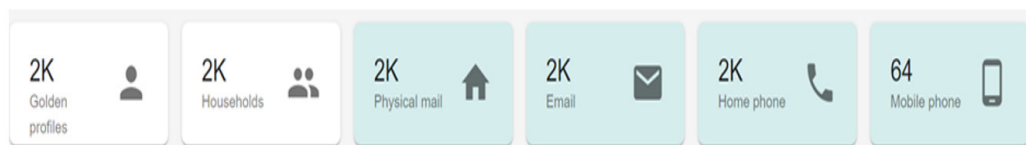
Figure 17: Connected Identity Dashboard



- The upper panel displays the records of golden master with details like count of golden master profiles, count of household, count of customer personal details like physical mail, emails, home phone and mobile phone.
- The Enrollment by Channel section displays the channel wise enrollment trend of customers.
- The Enrollment Trend section displays the overall volume of the customers enrolled over a period of time.
- The Demographics section displays the demographics details of the customer.
- The Consent section displays the customer consents.

The following section describes each of the CI dashboard panels:

Figure 18: Connected Identity Dashboard



The above section on the CI dashboard displays the details of the golden master records. The count 2k refers to any count between 2k to 3k. If the count is less than 1000 records, the actual number of count is displayed. For example 63 mobile phone.

- Master Profiles—shows the total number of golden customer profiles. You can click on Master Profiles to drill down to view golden customer record details as in [Figure 19](#). By default, the UI displays 10 number of customer records. You can customize to display 5 or 20 records per page.

Figure 19: Golden Customer Records

ID ↑	Name	Gender
20	Paul Jonas Schwarz	M
21	Lukas Robert Fischer	M
22	Chao Ru Chen	M
23	Sandra Leoni Schafer	F
24	Samantha Ava Wilson	F
25	Al-mu Kai-ying Tasi	F
26	Emma Elodie Bertrand	F

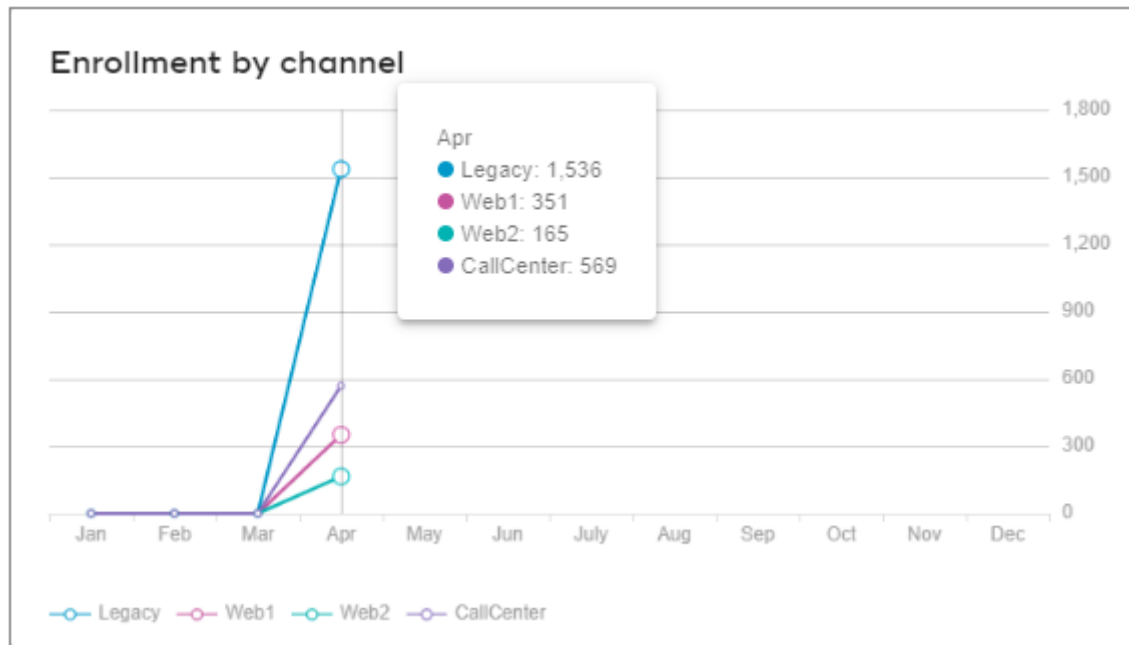
Items per page: 25 1 – 25 of 2621 < >

- **Households:** displays the total count of households of the golden customers.
- **Physical Mail:** displays the total number of physical mails received by the golden customers.
- **Email:** displays the total number of electronic mails received by the golden customers.
- **Home Phone:** displays the total number of home phone registered for the golden customers.
- **Mobile Phone:** displays the total number of mobile phone registered for the golden customers.

Enrollment by Channel

Enrollment by Channel UI displays the number of customers enrolled from each different sources (legacy, Web1, Web2 and callcenter) over a period of time in a graphical format as in [Figure 20](#). The x-axis displays the time period in months and y-axis displays the numbers of customers. The different sources are represented by different colours.

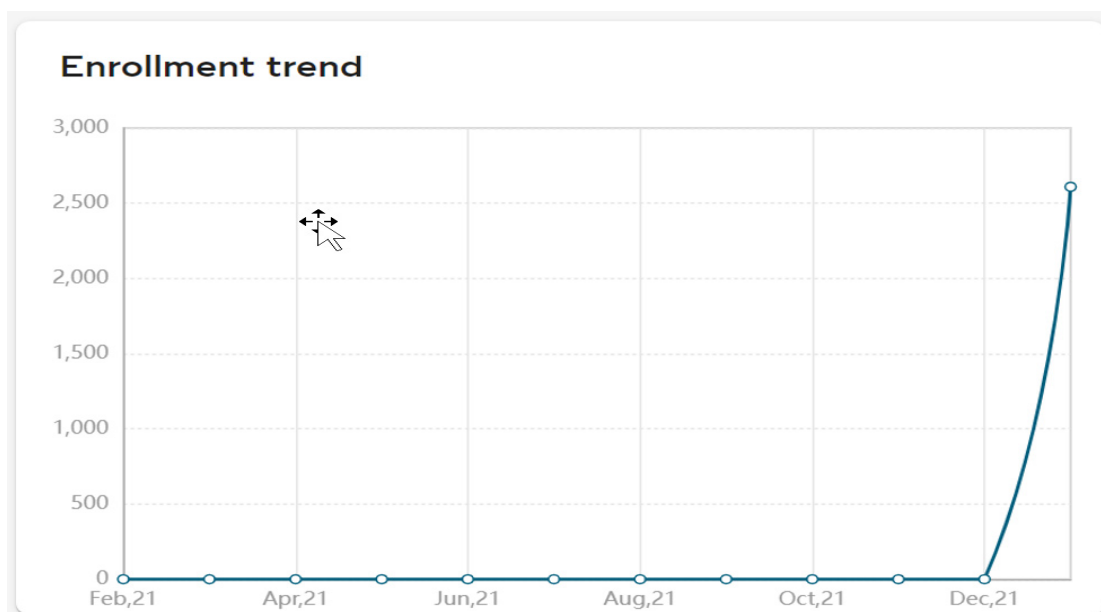
Figure 20: Enrollment by Channel



Enrollment by Trend

Enrollment by Channel UI displays the total number of customers enrolled from all the different sources (legacy, Web1, Web2 and callcenter) at a particular period of time as in [Figure 21](#). The x-axis displays the time period in months and y-axis displays the numbers of customers.

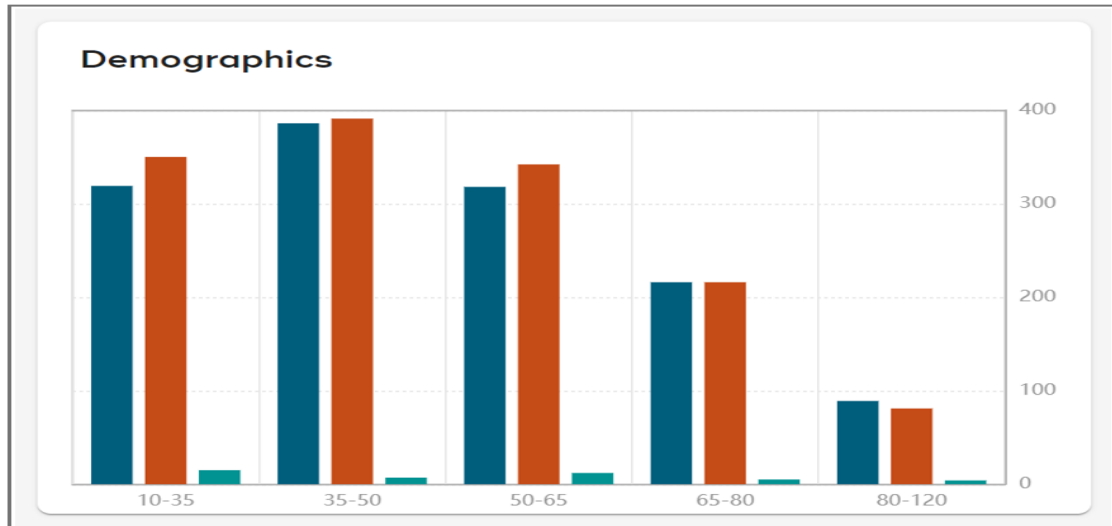
Figure 21: Enrollment Trend



Demographics

The Demographics UI displays in bar chart graphical representation of the number of customers based on their age group and on the gender type as in [Figure 22](#). The x-axis represents the age group and y-axis represents the number of customers. The different colors on the bar chart represents the gender type (Male, Female and Unknown).

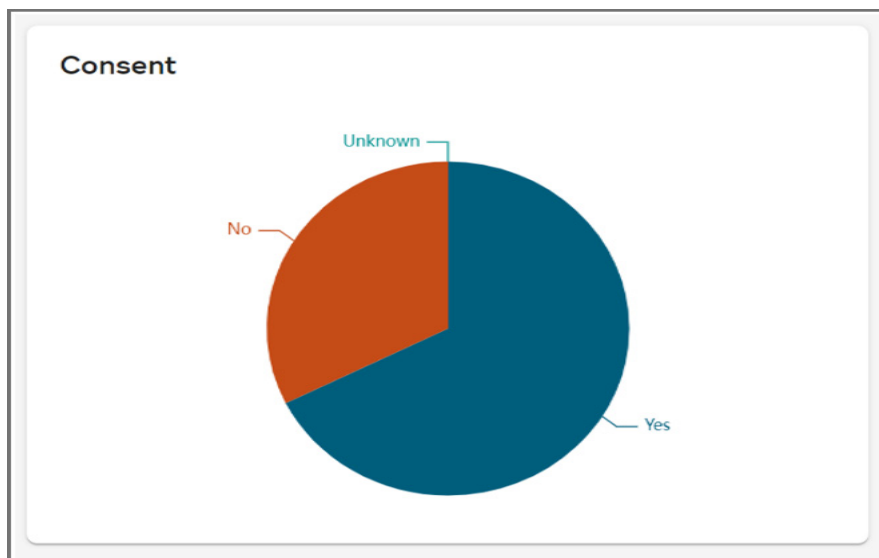
Figure 22: Demographics



Consent

The consent chart displays the different consent taken from the customers. The different parameters of consent include the following: Yes, No and Unknown represented by different colors as [Figure 23](#).

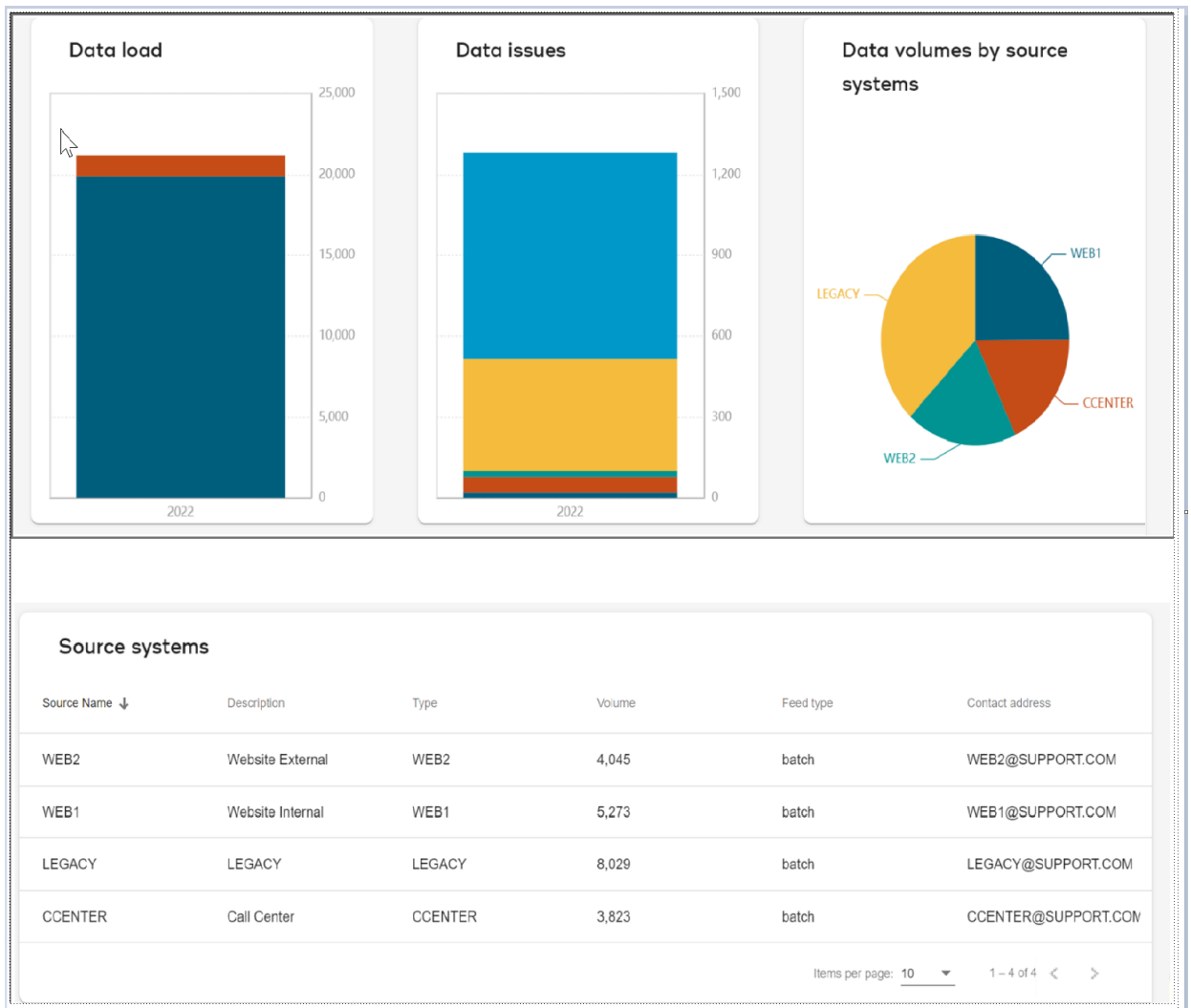
Figure 23: Consent



Data Ingestion Dashboard

The Data Ingestion Dashboard displays statistics of loading of consolidated Customer data from different source systems. It also provides details around different categories of Data issues encountered during data load process. The Dashboard shows details of all sources involved in the data load process. On the Data Ingestion Dashboard, you can view volume of data loaded for different time frame (past 6 months, 3 months, 1 month, 2 weeks, 1 week and 24 hours) and by different sources (Web1, Web2 Legacy and Ccenter). You can select the source system and time frame from the corresponding drop downs and refresh the screen to view the filtered data. By default, the data is displayed for all source systems and for 1 week time frame as in [Figure 24](#).

Figure 24: Data Ingestion Dashboard

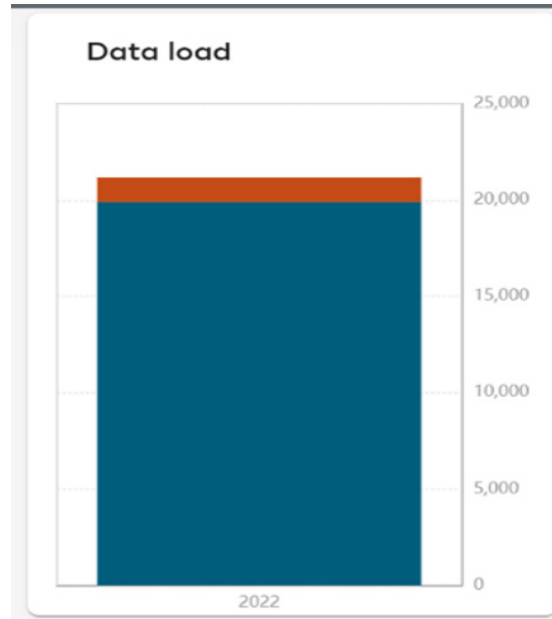


- **Data Load**

The data load graph displays the total number of customer records loaded from different source systems into the staging areas. It shows both the cleansed data and error records in

different colors. The C & S validation rules are applied on the data in the staging area and the total count of cleansed data are displayed as successful records and the error records are displayed as Error as in [Figure 25](#).

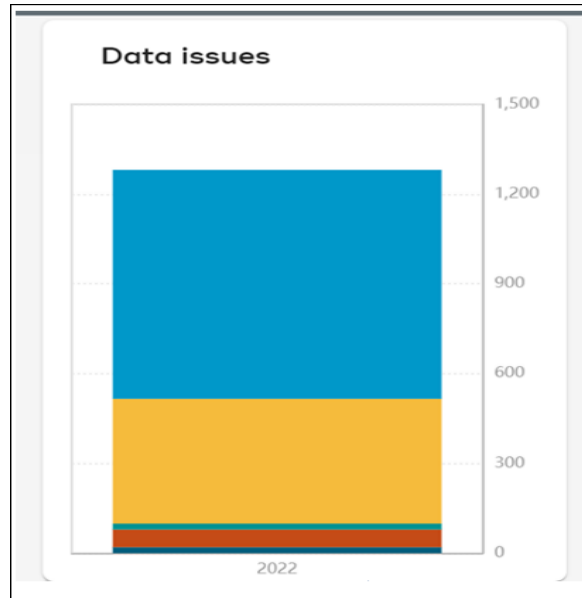
Figure 25: Data Load



- **Data Issues**

The Data Issues graph displays the data errors that occurred during the data load process. It shows the count of errors that occurred against each of the validation parameters (Zip code, Email address, Phone number, Gender and Birthdate) defined in the validation rules as in [Figure 26](#).

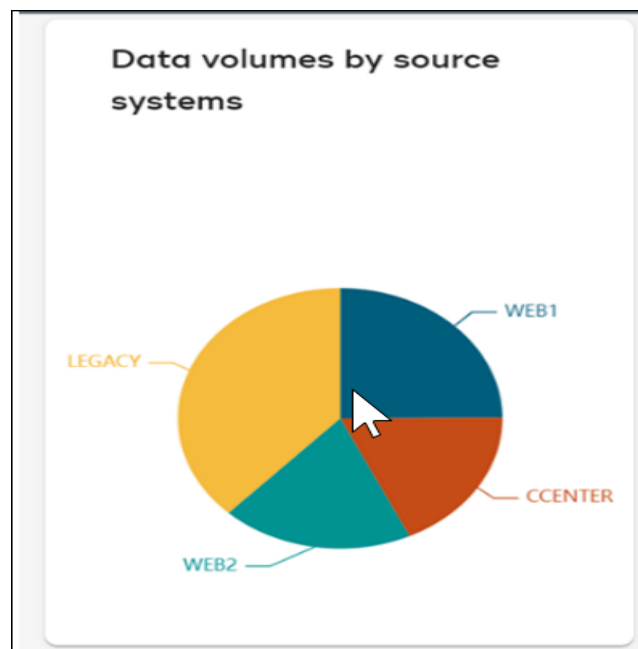
Figure 26: Data Load



- **Data Volumes by Source Systems**

The Data Volume by Source Systems graph displays the total number of records (both the successful records and error records) loaded by each source system as in [Figure 27](#).

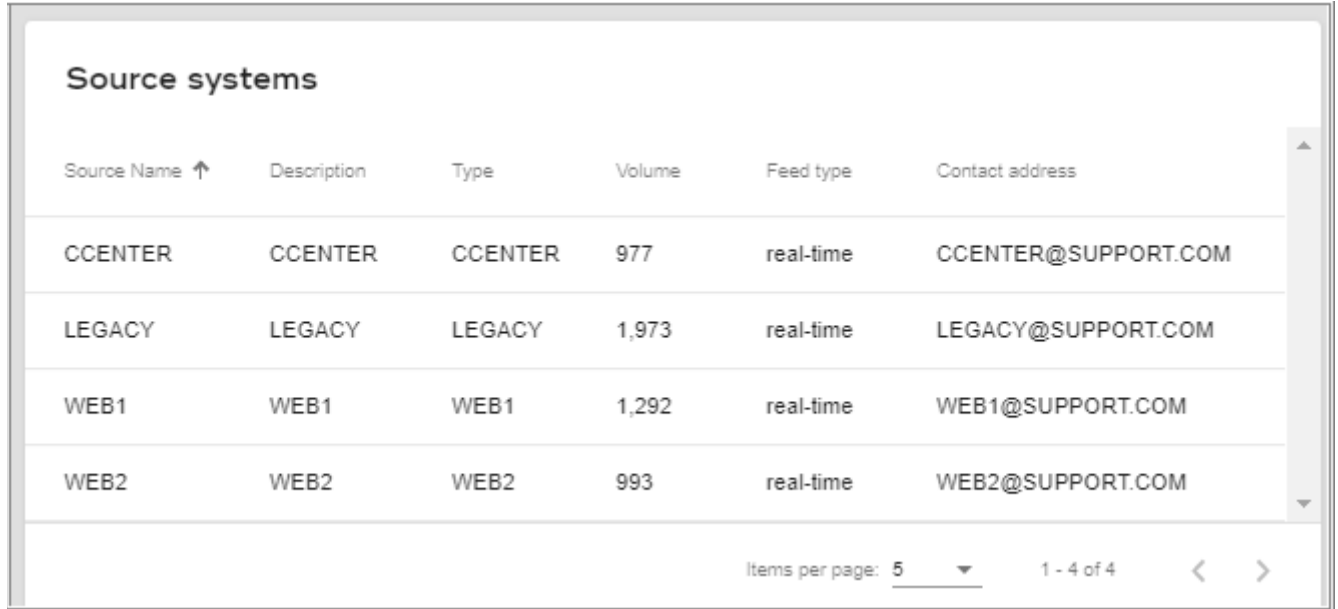
Figure 27: Data Volume by Source System



- **Source Systems**

The Source Systems UI section displays the different source system details like, Source Name, description, type, volume (total number of records), feed type and contact address as in [Figure 28](#).

Figure 28: Source Systems



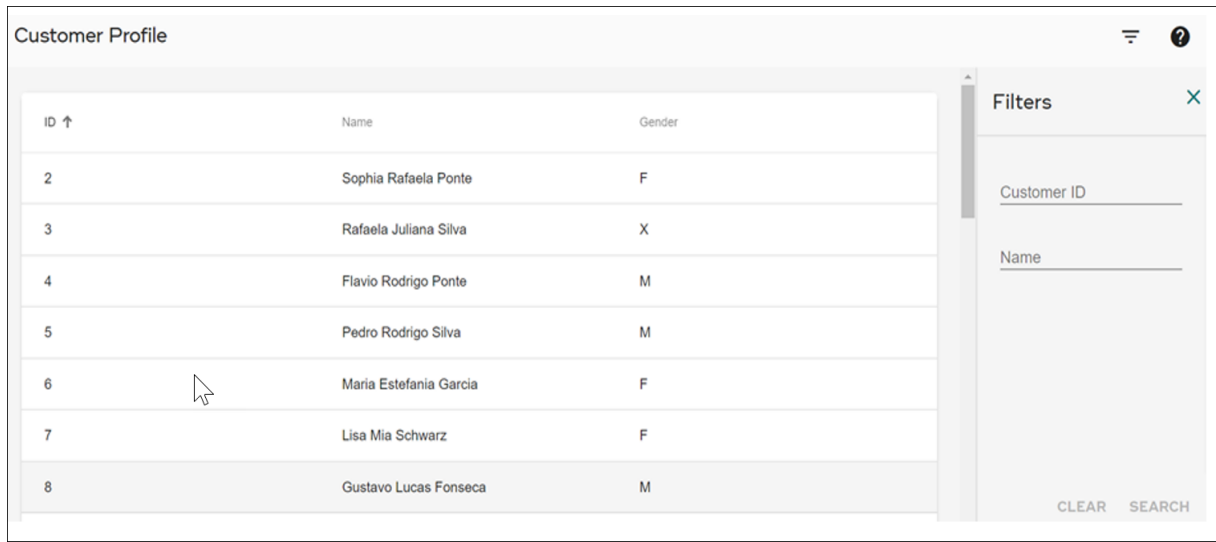
Source Name ↑	Description	Type	Volume	Feed type	Contact address
CCENTER	CCENTER	CCENTER	977	real-time	CCENTER@SUPPORT.COM
LEGACY	LEGACY	LEGACY	1,973	real-time	LEGACY@SUPPORT.COM
WEB1	WEB1	WEB1	1,292	real-time	WEB1@SUPPORT.COM
WEB2	WEB2	WEB2	993	real-time	WEB2@SUPPORT.COM

Items per page: 5 1 - 4 of 4 < >

Search Customer

The Search Customers UI displays the list of customers. The Customer 360 view of any customer starts with a search of customers you are interested in. Customers can be searched using the filters Customer ID, Name or Email as in [Figure 29](#).

Figure 29: Customer Profile

A screenshot of the 'Customer Profile' UI. It features a table with columns 'ID ↑', 'Name', and 'Gender'. The table contains 8 rows of customer data. To the right of the table is a 'Filters' sidebar with input fields for 'Customer ID' and 'Name', and 'CLEAR' and 'SEARCH' buttons at the bottom. The table has a light gray header and alternating row colors. A mouse cursor is pointing at the row with ID 6.

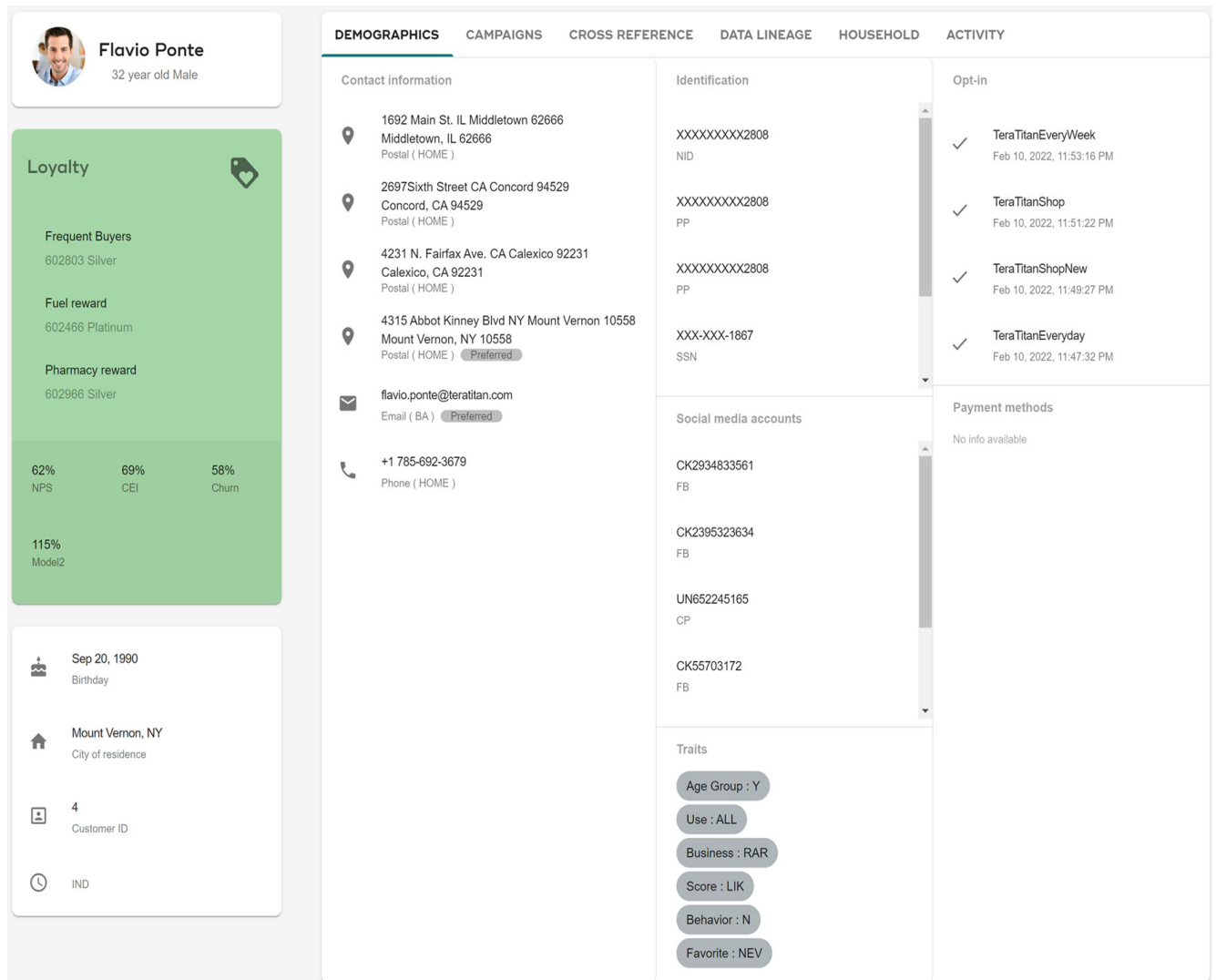
ID ↑	Name	Gender
2	Sophia Rafaela Ponte	F
3	Rafaela Juliana Silva	X
4	Flavio Rodrigo Ponte	M
5	Pedro Rodrigo Silva	M
6	Maria Estefania Garcia	F
7	Lisa Mia Schwarz	F
8	Gustavo Lucas Fonseca	M

Filters

CLEAR SEARCH

On the Search Customer UI, click on the any row of interested customer data will take you to the Customer 360 view UI. This UI displays all aspects of the customer details like profile details, loyalty details and other details of customer under different tabs like Demographics, Campaigns, Cross-reference, Household, Interactions and Purchases as in [Figure 30](#).

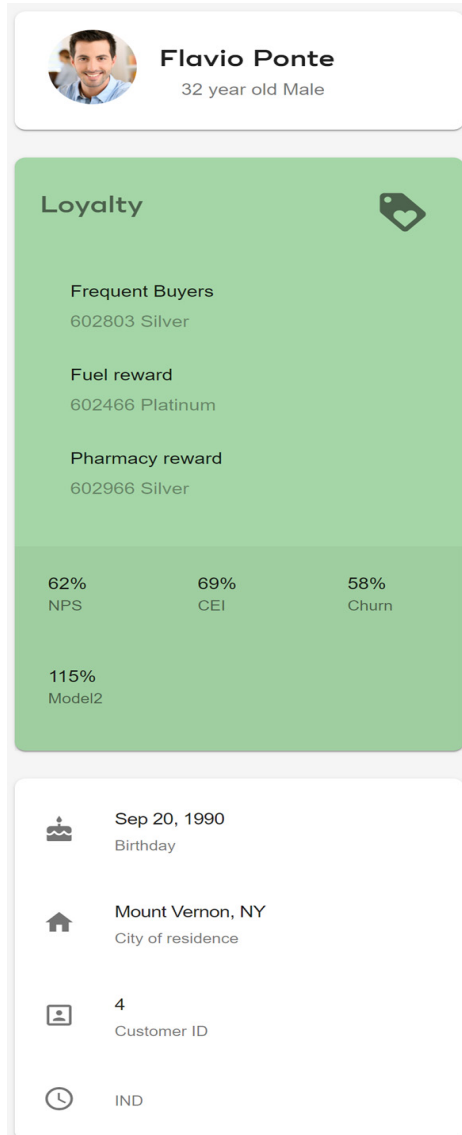
Figure 30: Customer 360 View



- **Profile Description**

The Profile Description section displays the top level information about the customer like customer name, age and gender. It also displays the profile picture of the customer, Birth date of the customer in the format month, date and year, city of residence, customer Id and type of customer (individual customer or Organization customer) as in [Figure 31](#).

Figure 31: Customer 360 View—Profile Details



- **Loyalty**

The Loyalty section displays the loyalty customer count and loyalty points in the form of fuel rewards. The loyalty of customer is the customer's satisfaction as happy customers and the customers devotion to company's products or services. The company in turn provides the customers with loyalty points, discounts and fuel reward points.

- **Demographics**

The following different personal details of the customer are displayed under Demographics tab as in [Figure 32](#):

- Contact Information section displays the customer current home address, email Id and contact phone number.
- Work section displays the customer work address.
- Identification section displays all identification types and numbers of the customer.

- Social Media Accounts section displays customer’s social media types and Id.
- Traits section displays the additional details of the customer that were not captured as part standard data model details.
- Opt-in section displays the various brands opted by the customer. It also displays the date and time as when the option was opted by the customer.
- Payment Methods section displays the different methods of payments requested by the customer. It displays the card number along with the method opted.
- Touchpoints section displays the different devices used by the customer for browsing and selecting the options.

Figure 32: Customer 360 View—Demographics

DEMOGRAPHICS	CAMPAIGNS	CROSS REFERENCE	DATA LINEAGE	HOUSEHOLD	ACTIVITY
Contact information <div> <div>3297 City Blvd W MT Sumatra 59083 Sumatra, MT 59083 Postal (HOME)</div> <div>3339 Abbot Kinney OK Snow 74567 Snow, OK 74567 Postal (HOME) Preferred</div> <div>85 Barley Mill Road KS Scandia 66969 Scandia, KS 66969 Postal (HOME)</div> <div>sophia.ponte@teratitan.com Email (BA) Preferred</div> <div>+1 865-388-0869 Phone (HOME)</div> </div>		Identification <div> <div>XXXXXXXX9168 NID</div> <div>XXXXXXXX9168 NID</div> <div>XXX-XXX-1352 SSN</div> <div>XXXXXXXX8080 PP</div> </div>		Opt-in <div> <div>✓ TeraTitanEveryWeek Jan 11, 2022, 6:15:02 PM</div> <div>✓ TeraTitanShop Jan 11, 2022, 6:13:09 PM</div> <div>✓ TeraTitanShopNew Jan 11, 2022, 6:11:13 PM</div> <div>✓ TeraTitanEveryday Jan 11, 2022, 6:09:15 PM</div> </div>	
		Social media accounts		Payment methods No info available	

- **Campaigns**

The Campaigns tab displays the various marketing campaigns offered to the customers by company as in [Figure 33](#). Under the Campaigns tab, the Interaction Details section displays the details of the first campaign offered to the customer. The right panel displays the five most recent campaigns offered to the customer. It displays the campaign details like campaign name, date on which the campaign was offered, the status of the campaign (delivered or not delivered) and the reason for the campaign.

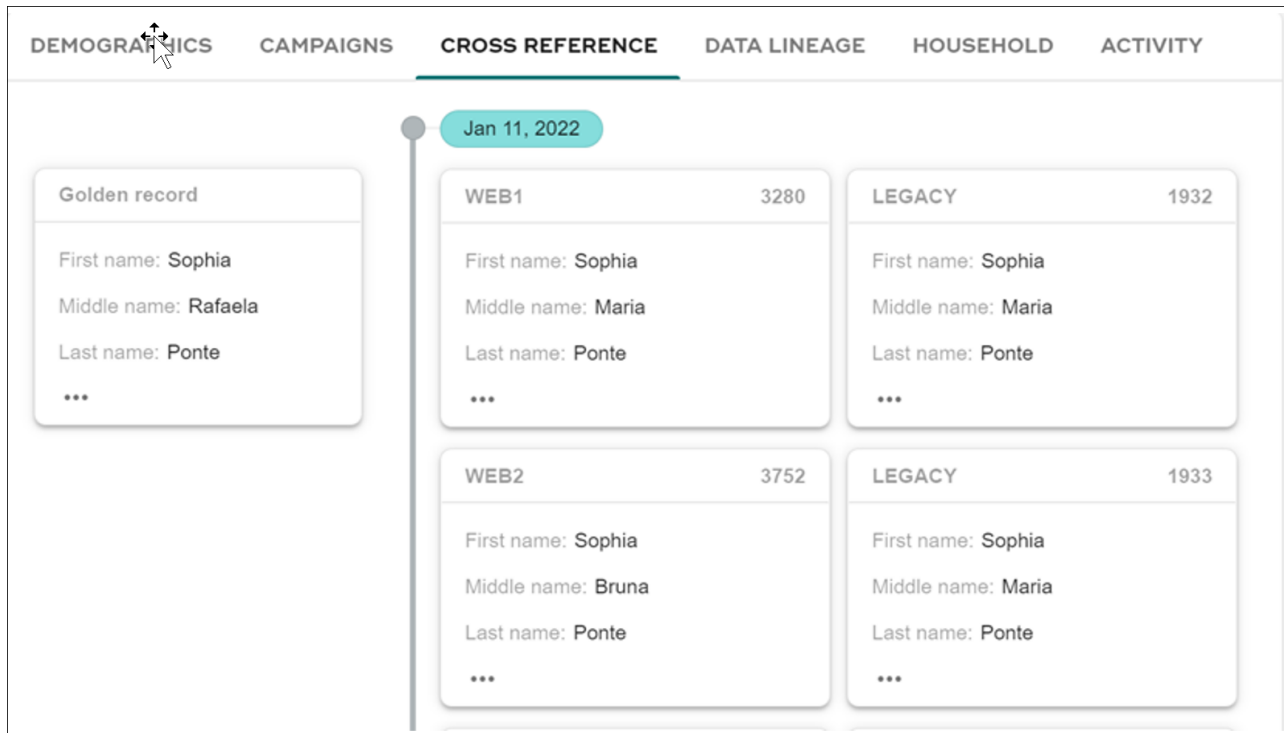
Figure 33: Customer 360 View—Campaigns

DEMOGRAPHICS	CAMPAIGNS	CROSS REFERENCE	DATA LINEAGE	HOUSEHOLD	ACTIVITY
Interaction details Entered:N/A - Bookmark or Unlisted Campaign First campaign sent:Feb 4, 2019, 1:30:00 PM Valid authorization:YES Opt in:Email Address		5 most recent marketing campaigns			
		Campaign name	Date ↓	Status	Reason
		Public relations or publicity Campaign	February 4, 2019	NotDeliver	Incorrect Door Number
		N/A - Bookmark or Unlisted Campaign	February 4, 2019	Delivered	Delivered
		Advertising or Unlisted Campaign	February 4, 2019	NotDeliver	Invalid Postal Coder
		Vouchers and Coupons	February 4, 2019	Delivered	Delivered

- **Cross-Reference**

The cross reference section displays the Data lineage of customer records. It displays the golden master records that were matched and merged from various source systems as in [Figure 34](#). Different attributes like customer Id, first name, last name, address, city, postal code, state, email, email type, country code and local number are used in matching and merging process to create the golden master records.

Figure 34: Customer 360 View—Cross-Reference



- **Data Lineage**

The Data Lineage section displays the various changes in data of the customer that have occurred over a period of time. Currently the UI provides lineage of the different types of addresses of any customer namely street address, email address and telephone numbers. Each type of address has its own subtypes (for example, street address can be of the type home, office etc.). The lineage is displayed as per the type of address and its subtype selected. Different information like customer Id, creation date, end date, subtype, telephone number, email, address, city, postal code, state etc. are displayed as per the selection.

Under the Data Lineage tab, by default it displays the street address of the subtype that occurs first in the dropdown. The subtype dropdown is dynamic in nature. It loads the different subtypes available for the selected address type. Lineage of any address can be loaded by changing the address type and its subtype dropdowns. If a customer does not have a lineage for any particular type or subtype, "No info available" message is displayed.

The UI displays the changes that occurred on the selected type and subtype based on the creation date and end date of the data as in [Figure 35](#). On the UI, you can see and compare the data changes.

Figure 35: Customer 360 View—Data Lineage

DEMOGRAPHICS		CAMPAIGNS		CROSS REFERENCE		DATA LINEAGE		HOUSEHOLD		ACTIVITY	
Data lineage						Street address ▾		HOME ▾			
Customer ID	Creation date ▾	End date	Address type	Address line	City	State		Country		Zip code	
2	Jan 31, 2019, 12:00:00 AM	Jan 11, 2022, 12:00:00 AM	HOME	85 Barley Mill Road KS Scandia 66969	Scandia	KS				66969	
2	Jan 31, 2019, 12:00:00 AM		HOME	3339 Abbot Kinney Snow 74567	Snow	OK				74567	

The below Views are created to populate the lineage data for each of the address types.

REF_STREET_ADDR_V - for Street Address

REF_ELCTRNC_ADDR_V - for Email

REF_TLPHN_NUM_V - for Telephone

The above view names are used directly in the UI to load the options in the address type dropdown. If a new address type to be added, a new view should be created to populate the data and the same should be added in the dropdown list. As the subtype dropdown is dynamic in nature, it will automatically pick the subtypes even if a new subtype gets added for a particular address type.

- **Household**

The Household section displays the household details of the selected customer. At any point of time, customers having common address (Email, Telephone, Street) will be considered as belonging to same household. All members of the household will have a common household Id as in [Figure 36](#). From the customer ID, first name or last name, you can drill down to the Customer Loyalty page.

Figure 36: Customer 360 View—Household

DEMOGRAPHICS	CAMPAIGNS	CROSS REFERENCE	DATA LINEAGE	HOUSEHOLD	ACTIVITY
Household ID ↑	Customer ID	Name	Gender	Head of household	
3145	2	Sophia Ponte	F	No	
3145	1421	Arthur Ponte	M	Yes	
<div>Items per page: 10</div> <div>1 – 2 of 2</div>					

- **Activity**

The Activity section displays the details of various purchases made by the customer. It displays the details like Purchase Id, type of purchase (Retail purchase or Web purchase), purchase description, status (shipped, delivered, returned or canceled), purchase creation date and updated date.

CHAPTER 5 Connected Identity Web Services

What's In This Chapter

This chapter provides information on connected identity web services.

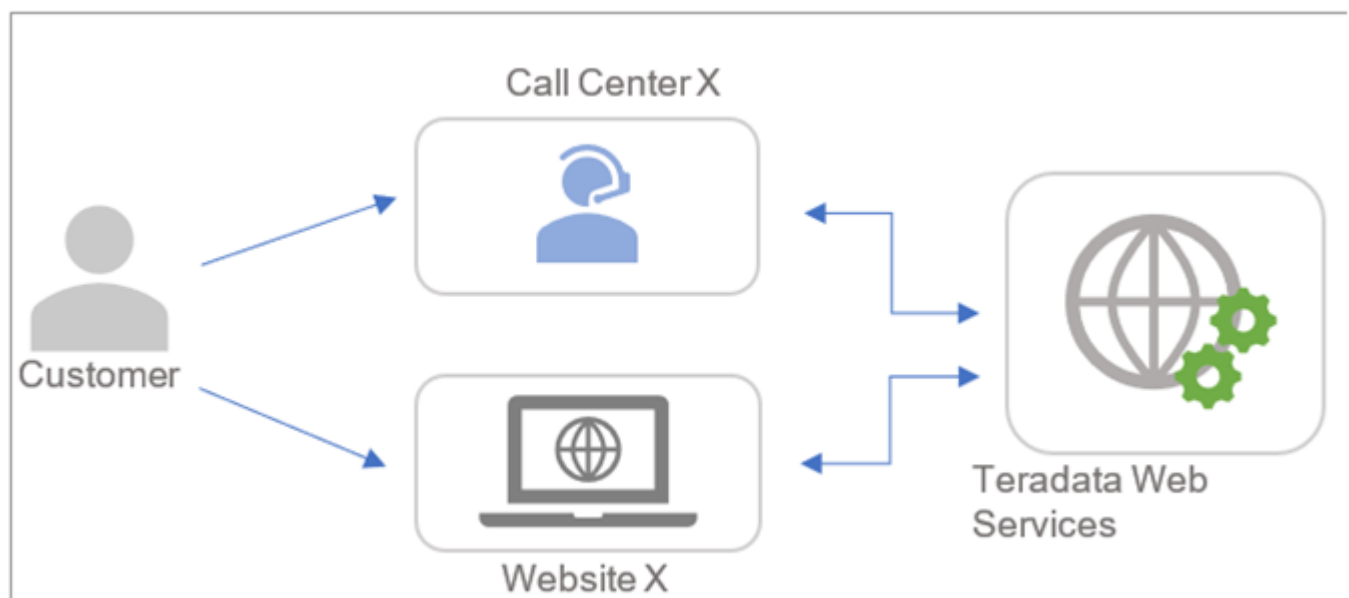
Topics include:

- [Introduction](#)
- [Web Service Implementation Details](#)
- [Search Web Services](#)
- [Admin Web Services](#)
- [Update Web Services](#)

Introduction

This chapter describes the processing of the data interface web service either external or internal and provides a detailed layout for the web service methods, values, and rules used in processing. A web service is a collection of codes, module or a piece of software which can be accessed by different applications. It is mostly used for exchanging data and are built on open standards such as HTTP and XML-based protocols. They can be available over the internet or private networks (intranet).

Figure 37: CI Web Services

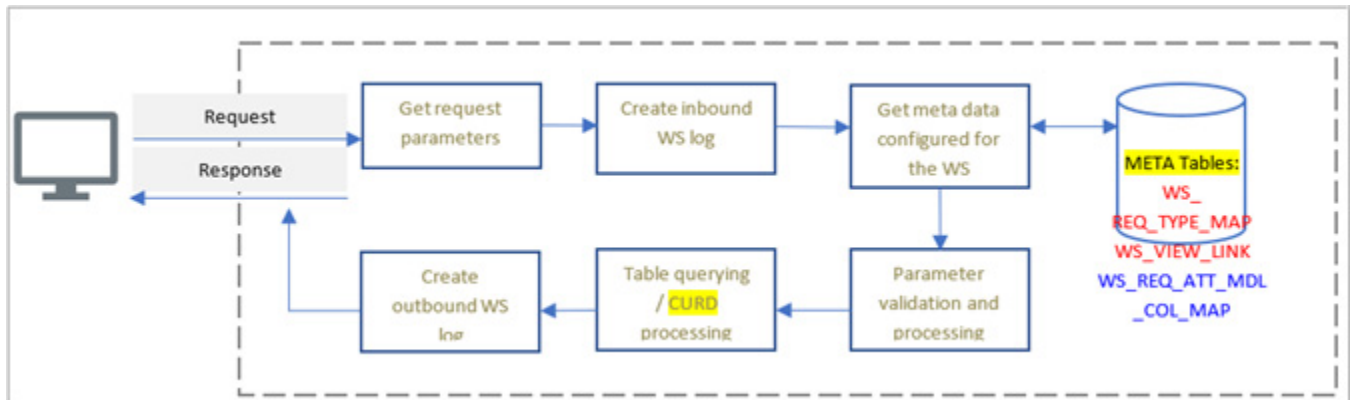


The CI Web services are based on MDM framework rules. The solution can be applied to the use cases as in [Figure 37](#). As seen in [Figure 37](#), a customer can either enroll himself in a new program by calling a call center or registering online thru a website. During enrollment, the agent or the website needs to verify if the customer is already existing or not. The application will pass a request with type SEARCH to the Teradata web service. Teradata web service will then send a response of the result of the search and other data. Other applicable web services which can be used by website admins or call center admins are Admin and Update wherein administrative tables can have data inserted, updated or deleted thru accessing of the Teradata web services.

Web Service Implementation Details

When a request is made from an application, the parameters that are passed will be processed by the Teradata web service API. [Figure 38](#) displays the simplified process diagram of the Web service API. It is metadata driven and is dependent on the data configured in the tables. Parameter validation and processing includes - required column checking, primary / foreign key validations, internal system default value processing, where clause generation (for search), etc.

Figure 38: Teradata Web Service APIs



Request Types

Admin and Update Web Services includes the following request types - Add, Update, Delete, and View. Search Web Services have different request types. Parameters expected for input and output are mostly configured and are based on the metadata tables. All web services use a common framework to process the above request types.

Add, update and delete request types use the below two web service metadata tables:

- WS_MDL_MAP

This metadata table mainly stores the request types applicable in a web service. The framework reads this table first to check if the web service can process the request type passed in the request. [Table 1](#) describes the WS_MDL_MAP column descriptions.

Table 1: WS_MDL_MAP API Column Description

Column Name	Type	Value	Description
WEB_SERVICE_NAME	String	-----	Web Service Name
REQUEST_TYPE	String	Add, Update, Delete, View	-----
MODEL_NAME	String	-----	Name of models associated with the Web Service
SEQ	Integer	-----	Order on how data will be saved in the model
REC_EXIST	String	CONTINUE, EXIT, <null>	Can be null; used mainly during ADD
ARCHIVE_FLG	String	Y, <null>	Used for main/sub tables (non-archive tables) Set to 'Y' if the data needs to be archived, else null

- WS_REQ_ATT_MDL_COL_MAP

This metadata table stores the parameters per request type which will be passed in the request, or is needed internally in the system. The parameters configured are mapped to their respective models, logical column names, constraints etc. (if available). If the parameter column is not defined in this table, the column of that table will not have data inserted or updated in the system. [Table 2](#) describes the WS_REQ_ATT_MDL_COL_MAP column descriptions.

Table 2: WS_REQ_ATT_MDL_COL_MAP API Column Description

Column Name	Type	Value	Description
WEB_SERVICE_NAME	String	-----	Web Service Name
REQUEST_TYPE	String	Add, Update, Delete, View, Archive	-----
REQUEST_PARAM	String	Request parameter name	Request parameter name
REQUIRED	String	Y, N, <null>	
MODEL	String	Model Name, <null>	Model name mapped to the request parameter
MODEL_COLUMN	String	Model column name, <null>	Corresponding model column name of the request parameter

Table 2: WS_REQ_ATT_MDL_COL_MAP API Column Description

Column Name	Type	Value	Description
COLUMN_OPERATION	String	INPUT	Flag on how the parameter will behave. Fields coming in the request which are not primary keys or foreign keys.
		INPUT_PK,	Primary key field in the request which is user generated
		INPUT_PK_SEQ,	Primary key field, which is not passed in the request, is an INTEGER type and is generated internally in the system.
		INPUT_PKFK,	Primary key field in the request but is a foreign key in another table and needs to be validated if existing.
		INPUT_PKFK_MAP,	Primary key field of a subtable, which is not passed in the request but is a PK coming from the main/previous tables which have been processed.
		INPUT_FK,	Foreign key field in the request, which is user generated and needs to be validated if existing in the reference table.
		INPUT_FK_MAP,	Foreign key field of a sub table, which is not passed in the request but is a PK coming from the main/previous tables which have been processed.
		SYS_INTERNAL,	Field in the model whose value is expected to be generated internally in the system or has a default value defined.
		RETAIN_VALUE,	This operation is used when the parameter is passed in the request and is expected to be displayed in the RESPONSE.
		DELETE#<ENDDATE ENTITYSTATE>,	Used for columns with 'DELETE' request type. Framework deletion is updating the effective end date or the entity_state of the table. Configure the effective end date column and indicate the column operation = 'DELETE#ENDDATE' If the effective end date is not available in the table then use column operation = 'DELETE#ENTITYSTATE'
COLUMN_DATA_TYPE	String	-----	Data type of the column in the table definition

Table 2: WS_REQ_ATT_MDL_COL_MAP API Column Description

Column Name	Type	Value	Description
COLUMN_CONSTRAINT	String	PK, PKFK, FK, <null>	----
COL_OBJECT_MAP	String	----	Logical Table name with logical column name corresponding to the request parameter if the value of COLUMN_CONSTRAINT is PKFK or FK. Format: <logical table name>.<logical column name> Ex: CHNL_TYPE.ChannelTypeCd
OBJECT_MAP_TYPE	String	----	-----
COLUMN_DEF_VAL	String	----	Used along COLUMN_OPERATION = 'SYS_INTERNAL'. This column is for fields, which are not expected to be passed during request and is generated from the system. Define the default value of the field in this column. For ARCHIVE request type: If the field is not passed in the request, then the default value will be considered.
COLUMN_API_VAL	String	----	Used along COLUMN_OPERATION = 'SYS_INTERNAL' Define the API name to be called in the rule file to generate the value of the field. Logic: If COLUMN_API_VAL is null, then system will get the value defined in COLUMN_DEF_VAL column

Search Web Services use the below two meta tables:

- WS_REQ_TYPE_MAP

This metadata table stores the parameters per request type which will be passed during API call. The framework reads this table first to check if the web service can process the RequestType passed in the inbound request. [Table 3](#) describes the WS_REQ_TYPE_MAP column descriptions.

Table 3: WS_REQ_TYPE_MAP API Column Description

Column Name	Type	Value	Description
WEB_SERVICE_NAME	String	----	Web Service Name.
REQUEST_TYPE	String	----	----

Table 3: WS_REQ_TYPE_MAP API Column Description

Column Name	Type	Value	Description
VIEW_NAME	String	----	View that will used by the WS associated with the request type.
COLUMN_WEB	Integer	----	Request/Response parameter name.
COLUMN_VIEW	String	----	View column name corresponding to the parameter name.
COLUMN_DATA_TYPE	String	-----	Parameter data type.
COLUMN_OPERATION	String	RETAIN_VALUE, VIEW_RESULT, <null>	Will define how the parameter will behave. RETAIN_VALUE will be part of the Response even if not part of the result of the view. VIEW_RESULT will extract the data from the view.
COLUMN_STATE	String	ACTIVE, INACTIVE	----
SEARCH_FLAG	String	Y, N or <null>	Will be used to define if a parameter can be used as a search criteria.
REQUIRED_FLAG	String	Y, N or <null>	Will be used to define if a parameter is required to have data. System will return error if set to Y and no value has been provided.
START_POINT	String	----	A flag used to identify that a parameter is a part of a sub-group of the result.
END_POINT	String	----	A flag used to identify that the parameter is a sub-group header.

- WS_VIEW_LINK

Every request type is mapped to a view. There are cases that a request type is mapped to multiple views. Some properties per request type are stored also in this metastable.

[Table 4](#) describes the WS_VIEW_LINK column descriptions.

Table 4: WS_VIEW_LINK API Column Description

Column Name	Type	Value	Description
WEB_SERVICE_NAME	String	----	Web Service Name. Not null
REQUEST_TYPE	String	----	Request type, not null.
POINT_NAME	String	----	FLAG name defined in WS_REQ_TYPE_META for a request type having multiple views. Required if request type has multiple views.

Table 4: WS_VIEW_LINK API Column Description

Column Name	Type	Value	Description
COLUMN_ORDER	Integer	-----	-----
VIEW1	String	VIEW_NAME	View name associated with the request type. Not null.
VIEW2	String	VIEW_NAME, <null>	Second View associated with the request type. Required if request type has multiple views.
COLUMN1	String	-----	View column name that will be used to join to the 2nd view.
COLUMN2	String	-----	View column name that will be used to join to the 1st view. Required if request type has multiple views.
LINK_STATUS	String	ACTIVE, INACTIVE, <null>	-----
SORT_BY	String	-----	View column name that will be used for the sorting.
SORT_ORDER	String	Desc, Asc, <null>	Order on how the records will be sorted. This is required if the result is intended to be sorted.
TOP_RECORD	Int	-----	Number of top records to be extracted. Often used for extracting recent records. This is required if the result is intended to extract latest record.

Table 4: WS_VIEW_LINK API Column Description

Column Name	Type	Value	Description
COLUMN_OPERATION	String	INPUT	Flag on how the parameter will behave. Fields coming in the request, which are not primary keys or foreign keys.
		INPUT_PK,	Primary key field in the request, which is user generated.
		INPUT_PK_SEQ,	Primary key field, which is not passed in the request, is an INTEGER type and is generated internally in the system.
		INPUT_PKFK,	Primary key field in the request but is a foreign key in another table and needs to be validated if existing.
		INPUT_PKFK_MAP,	Primary key field of a sub table, which is not passed in the request but is a PK coming from the main/previous tables which have been processed.
		INPUT_FK,	Foreign key field in the request, which is user generated and needs to be validated if existing in the reference table
		INPUT_FK_MAP,	Foreign key field of a sub table, which is not passed in the request but is a PK coming from the main/previous tables which have been processed
		SYS_INTERNAL,	Field in the model whose value is expected to be generated internally in the system or has a default value defined.
		RETAIN_VALUE,	This operation is used when the parameter is passed in the request and is expected to be displayed in the RESPONSE.
		DELETE#<ENDDATE ENTITYSTATE>,	Used for columns with 'DELETE' request type. Framework deletion is updating the effective end date or the entity_state of the table. Configure the effective end date column and indicate the column operation = 'DELETE#ENDDATE' If effective end date is not available in the table then use column operation = 'DELETE#ENTITYSTATE'
COLUMN_DATA_TYPE	String	-----	Data type of the column in the table definition

Table 4: WS_VIEW_LINK API Column Description

Column Name	Type	Value	Description
COLUMN_CONSTRAINT	String	PK, PKFK, FK, <null>	----
COL_OBJECT_MAP	String	----	<p>Logical Table name with logical column name corresponding to the request parameter if the value of COLUMN_CONSTRAINT is PKFK or FK.</p> <p>Format: <logical table name>.<logical column name></p> <p>Ex: CHNL_TYPE.ChannelTypeCd</p>
OBJECT_MAP_TYPE	String	----	-----
COLUMN_DEF_VAL	String	----	<p>Used along COLUMN_OPERATION = 'SYS_INTERNAL'.</p> <p>This column is for fields which are not expected to be passed during request and is generated from the system.</p> <p>Define the default value of the field in this column.</p> <p>For ARCHIVE request type: If the field is not passed in the request, then the default value will be considered.</p>
COLUMN_API_VAL	String	----	<p>Used along COLUMN_OPERATION = 'SYS_INTERNAL'</p> <p>Define the API name to be called in the rule file to generate the value of the field.</p> <p>Logic: If COLUMN_API_VAL is null, then system will get the value defined in COLUMN_DEF_VAL column</p>

Search Web Services

Currently configured and implemented search web services includes the following:

- [SearchCustProfile](#)
- [GetCustomerAddress](#)
- [GetCustomerUAL](#)
- [CustManageLyltyAccount](#)
- [CustManagePymntAccount](#)
- [CustManageTrait](#)

SearchCustProfile

This web service allows you to get customer records. This Web service has the following request types:

- DETAIL
- ELECTRONIC
- TELEPHONE
- CUSTINFO
- EXTCUSTID
- LYLTYNUM
- ACCTNUM
- STADDR

Sample Rest Request/Response

- Resource:
`/mdm/xcorexml/services/CDHServices/rules SearchCustProfile?response=application/json`

- Request:

```
<root>
  <TransactionID Value="5002"/>
  <RequestType Value="DETAIL"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SearchCustID Value="430"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TotalSearchCount": 1,
    "CustomerProfileView": {
      "BrandCD": "KNOWN",
      "SourceProviderCD": "AIMIA",
      "TransactionID": 5002,
      "MarketingProgramID": 23,
      "LoyaltyAccounts": [
        {
          "LoyaltyAccountNum": 800430,
          "LoyaltyProgramID": 800430,
          "LoyaltyLvlCD": 1,
          "LA_FirstName": "Ludwig",
          "LA_MiddleName": "",
          "LA_LastName": "Koch",
          "LA_Gender": "M",
          "LA_DOB": ""
        }
      ]
    }
  }
}
```

```

],
"Electronic_Addresses":          [
    {
        "ElctrncAddrRqstId": "",
        "ElctrncAddrDmnName": "",
        "ElctrncAddrTxt": "Craig.Hackett@teratitan.com",
        "ElctrncAddrSbTypeCd": "HOME",
        "DmnRootCd": "",
        "EA_CREATED_DATE_TIME": "",
        "EA_SOURCE": "BackEnd"
    },
    {
        "ElctrncAddrRqstId": "",
        "ElctrncAddrDmnName": "",
        "ElctrncAddrTxt": "Craig.Hackett@teratitan.com",
        "ElctrncAddrSbTypeCd": "HOME",
        "DmnRootCd": "",
        "EA_CREATED_DATE_TIME": "",
        "EA_SOURCE": "BackEnd"
    }
],
"Street_Addresses":              [
    {
        "AddressSequenceID": "",
        "StreetAddrReqId": "",
        "AddressTypeCD": "Street",
        "AddressInvldFlg": "",
        "AddressIgnoreFlg": "",
        "AddressAbusedFlg": "",
        "AddrLine1Txt": "5518 Auburn Mill Road",
        "AddrLine2Txt": "",
        "AddrLine3Txt": "",
        "AddrLine4Txt": "",
        "StOrProvCd": "Texas",
        "CityName": "Roosevelt",
        "PostlCd": 76874,
        "LatMeas": 33.5815,
        "LngtdMeas": -101.6743,
        "CtryId": 840,
        "GeoAcrcyCd": "",
        "OrgCd": "",
        "BldgId": "",
        "SubBldgId": "",
        "Premscd": "",
        "ThoroughfareCd": "",
        "SA_CREATED_DATE_TIME": "01/04/2019 00:00:00",
        "SA_SOURCE":
"MDM420_VM_CDI_STAGE.WRK_STREET_ADDR_RESULT"
    }
],
"Accounts":                      [
    {
        "UserID": 43001,
        "UserAccountStartDt": "01/01/2019 00:00:00",
        "UserAccountRoleCD": "B",
        "CustAcctNum": 43001,
        "CustAcctName": "LUD",
        "CustAcctType": "IND"
    }
]

```

```

    },
    {
        "UserID": 43002,
        "UserAccountStartDt": "02/01/2019 00:00:00",
        "UserAccountRoleCD": "B",
        "CustAcctNum": 43002,
        "CustAcctName": "LK",
        "CustAcctType": "IND"
    }
],
"SearchCustID": 430,
"Cust_FirstName": "Ludwig",
"Cust_MiddleName": "",
"Cust_LastName": "Koch",
"Cust_FullName": "Ludwig Koch",
"OrganizationName": "",
"BirthDate": "07/09/1944 00:00:00",
"DeathDate": "",
"GenderTypeCD": "MALE",
"EthnicityTypeCD": "",
"NationalityCD": "USA",
"BirthMth": "",
"BirthYr": "",
"BirthDy": "",
"PetBreedCd": "",
"PetTypeCd": "",
"DcsdFlg": "",
"NickName": "",
"UnvrslAuthztnLangCd": "B",
"UnvrslAuthztnStrtDt": "",
"SocGrpName": "",
"ProfanityFlg": ""
}
}
}

```

Web Service Value Notes

Value notes are as per current metadata configured in the table WS_REQ_TYPE_MAP.

GetCustomerAddress

This web service allows you to get customer's address record. This Web service has the following request types:

- ALL
- STREET
- TELEPHONE
- ELECTRONIC
- LATEST

Sample Rest Request/Response

- Resource:
/mdm/xcorexml/services/CDHServices/rules/
GetCustomerAddress?response=application/json

- Request:

```
<root>
  <TransactionID Value="02071242"/>
  <RequestType Value="ALL"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <CustID Value="5700"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TotalSearchCount": 1,
    "CustomerProfileView": {
      "TransactionID": "02071242",
      "Telephone_Addresses": {
        "TN_AddrSeqId": "",
        "TelephoneNumRqstId": "",
        "TelephoneNumSbtypeCd": "MOBILE",
        "TelephoneSrcNum": "1 (425) 934-0975",
        "TelephoneLnNum": "+1 425-934-0975",
        "TelephoneE164Num": "",
        "TelephoneExtnNum": "",
        "TelephoneExchNum": "",
        "TelephoneAreaCdNum": "",
        "TelephoneCtryCdNum": "+1",
        "TN_CREATED_DATE_TIME": "",
        "TN_SOURCE": "BackEnd"
      },
      "Electronic_Addresses": {
        "ElctrncAddrRqstId": "",
        "ElctrncAddrDmnName": "",
        "ElctrncAddrTxt": "rodrigo.sousa@teratitan.com",
        "ElctrncAddrSbTypeCd": "HOME",
        "DmnRootCd": "",
        "EA_CREATED_DATE_TIME": "",
        "EA_SOURCE": "BackEnd"
      }
    }
  }
}}
```

Web Service Value Notes

Value notes are as per current metadata configured in the table WS_REQ_TYPE_MAP.

GetCustomerUAL

This web service allows the user to search different universal authorization associated with a customer. This Web service has the following request types:

- ALL or Current

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/GetCustomerUAL?response=application/json

- Request:

```
<root>
  <TransactionID Value="02071346"/>
  <RequestType Value="ALL"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <CustID Value="5001"/>
  <SourceProviderCD Value="ABC"/>
  <MarketingProgramID Value="5001"/>
  <BrandCD Value="MAIN"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TotalSearchCount": 2,
    "CustomerProfileView": [
      {
        "TransactionID": "02071346",
        "CustomerUniversalAuthID": 5001,
        "CustomerUniversalChnlCD": 222,
        "CustomerUniversalChnlStrtDt": "",
        "CustomerUniversalChnlEndDt": "",
        "CustomerUniversalLangCD": "UA_5001"
      },
      {
        "TransactionID": "02071346",
        "CustomerUniversalAuthID": 5001,
        "CustomerUniversalChnlCD": 111,
        "CustomerUniversalChnlStrtDt": "12/11/2018 00:00:00",
        "CustomerUniversalChnlEndDt": "12/31/9999 00:00:00",
        "CustomerUniversalLangCD": "UA_5001"
      }
    ]
  }
}
```

Web Service Value Notes

Value notes are as per current metadata configured in the table WS_REQ_TYPE_MAP.

CustManageLyltyAccount

This web service manages loyalty account associated with a customer. This Web service has the following request types:

- VIEWALL or VIEWLATEST

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
CustManageLyltyAccount?response=application/json

- Request:

```
<root>
  <TransactionID Value="02071401"/>
  <RequestType Value="VIEWALL"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <CustID Value="5001"/>
  <SourceProviderCD Value="ABC"/>
  <MarketingProgramID Value="5001"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TotalSearchCount": 2,
    "CustomerProfileView": [
      {
        "CustID": 5001,
        "TransactionID": "02071401",
        "LoyaltyAccountNum": 5001,
        "LoyaltyProgramId": 5001,
        "FirstName": "Brandoh",
        "MiddleName": "",
        "LastName": "Will",
        "Gender": "M",
        "DOB": "12/12/2018 00:00:00"
      },
      {
        "CustID": 5001,
        "TransactionID": "02071401",
        "LoyaltyAccountNum": 5011,
        "LoyaltyProgramId": 5011,
        "FirstName": "Andoh",
        "MiddleName": "",
        "LastName": "William",
        "Gender": "M",
        "DOB": "02/01/2018 00:00:00"
      }
    ]
  }
}
```

Web Service Value Notes

Value notes are as per current metadata configured in the table WS_REQ_TYPE_MAP.

CustManagePymntAccount

This web service manages payment account associated with a customer. This Web service has the following request types:

- VIEWALL or VIEWLATEST

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
CustManagePymntAccount?response=application/json

- Request:

```
<root>
  <TransactionID Value="02071409"/>
  <RequestType Value="VIEWALL"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <CustID Value="5001"/>
  <SourceProviderCD Value="ABC"/>
  <MarketingProgramID Value="5001"/>
  <BrandCD Value="MAIN"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TotalSearchCount": 7,
    "CustomerProfileView": [
      {
        "TransactionID": "02071409",
        "CustID": 5001,
        "PymntAccountNum": 7001,
        "PymntAccountTypeCD": "PT_IND",
        "PymntAccountSubTypeCD": "PST_IND_S",
        "PymntAcctCurrCD": "USD",
        "PymntAcctCOAID": 12345,
        "CreditRatingCD": "CR_501",
        "PymntAcctIssueDt": "01/01/2019 00:00:00",
        "PymntAcctExprDt": "01/01/2030 00:00:00",
        "SetOfBooksCD": "SB_S",
        "CustPymntAcctStrtDT": "01/01/2019 00:00:00",
        "CustPymntAcctUsageStrtDT": "PAY"
      },
      {
        "TransactionID": "02071409",
        "CustID": 5001,
        "TransactionID": "02071409",
        "PymntAccountNum": 2003,
        "PymntAccountTypeCD": "PT_IND",
        "PymntAccountSubTypeCD": "PST_IND_S",
        "PymntAcctCurrCD": "USD",
        "PymntAcctCOAID": 12345,
        "CreditRatingCD": "CR_501",
        "PymntAcctIssueDt": "01/01/2019 00:00:00",
        "PymntAcctExprDt": "01/01/2030 00:00:00",
        "SetOfBooksCD": "SB_S",
        "CustPymntAcctStrtDT": "01/01/2019 00:00:00",
        "CustPymntAcctUsageStrtDT": "PAY"
      }
    ]
  }
}
```

```
    }
  }
}
```

Web Service Value Notes

Value notes are as per current metadata configured in the table WS_REQ_TYPE_MAP.

CustManageTrait

This web service manages the traits associated with a customer. This Web service has the following request types:

- VIEWALL or VIEWLATEST

Sample Rest Request/Response

- Resource:

```
/mdm/xcorexml/services/CDHServices/rules/CustManageTrait?response=application/
json
```

- Request:

```
<root>
  <TransactionID Value="02071518"/>
  <RequestType Value="VIEWALL"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <CustID Value="430"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TotalSearchCount": 2,
    "CustomerProfileView": [
      {
        "TransactionID": "02071518",
        "CustID": 430,
        "TraitCD": "TRAIT_TRAVEL",
        "TraitVal": "DAILY",
        "TraitValDt": "",
        "TraitStartDt": "",
        "TraitEndDt": "",
        "ChannelId": "",
        "MktgProgramIDAsstn": 0
      },
      {
        "TransactionID": "02071518",
        "CustID": 430,
        "TraitCD": "TRAIT_TRANSPORT",
        "TraitVal": "CAR",
        "TraitValDt": "",
        "TraitStartDt": "",
        "TraitEndDt": ""
      }
    ]
  }
}
```

```

        "ChannelId": "",
        "MktgProgramIDAsstn": 1
    }
]
}
}}
```

Web Service Value Notes

Value notes are as per current metadata configured in the table WS_REQ_TYPE_MAP.

Admin Web Services

Currently configured and implemented admin web services includes the following:

- [MktgPrgMaintenanceRequest](#)
- [ChannelMaintenanceRequest](#)
- [PromotionMaintenanceRequest](#)
- [CampaignMaintenanceRequest](#)
- [UALMaintenanceRequest](#)
- [TraitMaintenanceRequest](#)

MktgPrgMaintenanceRequest

This web service allows user to perform request types in Marketing Program table. This Web service has the following request types:

- **ADD**

This request type is invoked every time a user needs to modify/update data in the table 'Marketing Program' (MKTG_PRG). User is required to pass the primary key of the table to update a record successfully.

Sample Rest Request/Response

- Resource:

```

/mdm/xcorexml/services/CDHServices/rules/
MktgPrgMaintenanceRequest?response=application/json
```

- Request:

```

<root>
  <TransactionID Value="12345"/>
  <RequestType Value="ADD"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="SPCD"/>
  <BrandCD Value="KNOWN"/>
  <MktgPrgName Value="Healthy Food"/>
  <MktgPrgDesc Value="Living Healthy"/>
  <MktgPrgTypeCD Value="PRGHLTH"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 12345,
    "EE_Code": 10,
    "EE_Message": "Webservice MktgPrgMaintenanceRequest for Request
Type ADD successful!",
    "TotalResultCount": 1
  }
}}
```

- **UPDATE**

This type is invoked every time a user needs to add data in the table 'Marketing Program' (MKTG_PRG). User must pass the required parameters and column field data to have a successful insert. The marketing program id is a generated sequence in the system which does not need to be included during the request.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
MktgPrgMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="112233"/>
  <RequestType Value="UPDATE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="SPCD"/>
  <BrandCD Value="KNOWN"/>
  <MarketingProgramID Value="3"/>
  <MktgPrgName Value="Living Healthy"/>
  <MktgPrgDesc Value="Living healthy and happy"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 112233,
    "EE_Code": 10,
    "EE_Message": "Webservice MktgPrgMaintenanceRequest for Request
Type UPDATE successful!",
    "TotalResultCount": 1
  }
}}
```

- **DELETE**

This request type is invoked every time a user needs to delete data in the table 'Marketing Program' (MKTG_PRG). Deletion in this table is updating the Marketing program end date to the current system date. User is required to pass the primary key to delete the record successfully.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
MktgPrgMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="123456"/>
  <RequestType Value="DELETE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="SPCD"/>
  <BrandCD Value="KNOWN"/>
  <MarketingProgramID Value="3"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 123456,
    "EE_Code": 10,
    "EE_Message": "Webservice MktgPrgMaintenanceRequest for Request
Type DELETE successful!",
    "TotalResultCount": 1
  }
}}
```

- **VIEW**

This request type is invoked every time a user needs to view data in the table 'Marketing Program' (MKTG_PRG). User can use the field columns of the Marketing Program table during search by indicating the value in the request.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
MktgPrgMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="321"/>
  <RequestType Value="VIEW"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SearchCustID Value="430"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="3"/>
  <BrandCD Value="KNOWN"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
```

```

        "Status": "Success",
        "TotalSearchCount": 1,
        "CustomerProfileView": {
            "TransactionID": 321,
            "MarketingProgramID": 3,
            "MktgPrgName": "Living Healthy",
            "MktgPrgDesc": "Living healthy and happy",
            "MktgPrgOpenDttm": "",
            "MktgPrgCloseDttm": "02/06/2019 12:57:01",
            "MktgPrgCmmtInd": "",
            "MktgPrgTypeCD": "PRGHLTH"
        }
    }
}

```

Web Service Value Notes

Value notes are as per current metadata configured in the tables

WS_REQ_ATT_MDL_COL_MAP (Add, Update, Delete) and WS_REQ_TYPE_MAP (View).

ChannelMaintenanceRequest

This web service allows the user to perform request types in the Channel table. This Web service has the following request types:

- **ADD**

This request type is invoked every time a user needs to add data in the table 'Channel' (CHNL). User must pass the required parameters and the column field data to have a successful insert. The channel id is a generated sequence in the system, which is not needed to be included during the request.

Sample Rest Request/Response

- **Resource:**

/mdm/xcorexml/services/CDHServices/rules/
ChannelMaintenanceRequest?response=application/json

- **Request:**

```

<root>
  <TransactionID Value="5002"/>
  <RequestType Value="ADD"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <ChnlName Value="Channel C"/>
  <ChnlTypeCD Value="CH_D"/>
  <ChnlCpctyQty Value="10"/>
  <ChnlStrtDt Value="01/20/2019"/>
  <ChnlEndDt Value="06/20/2019"/>
</root>

```

- **Response**

```

{"RESPONSES": {

```

```

    "Status": "Success",
    "RESPONSE": {
      "Status": "Success",
      "TransactionID": 5002,
      "EE_Code": 10,
      "EE_Message": "Webservice ChannelMaintenanceRequest for Request
Type ADD successful!",
      "TotalResultCount": 1
    }
  }
}

```

- **UPDATE**

This request type is invoked every time a user needs to modify/update data in the table 'Channel' (CHNL). User is required to pass the primary key of the table to update a record successfully.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
ChannelMaintenanceRequest?response=application/json

- Request:

```

<root>
  <TransactionID Value="5002"/>
  <RequestType Value="UPDATE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <ChnlID Value="2"/>
  <ChnlName Value="Channel B Update"/>
  <ChnlTypeCD Value="CH_B"/>
  <ChnlCpctyQty Value="50"/>
  <ChnlStrtDt Value="01/20/2019"/>
  <ChnlEndDt Value="06/20/2019"/>
</root>

```

- Response

```

{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 5002,
    "EE_Code": 10,
    "EE_Message": "Webservice ChannelMaintenanceRequest for Request
Type UPDATE successful!",
    "TotalResultCount": 1
  }
}
}

```

- **DELETE**

This request type is invoked every time a user needs to delete data in the table 'Channel' (CHNL). Deletion in this table is updating the Channel End Date to the current system date. User is required to pass the primary key to delete the record successfully.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
ChannelMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="5002"/>
  <RequestType Value="DELETE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <ChnlID Value="4"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 5002,
    "EE_Code": 10,
    "EE_Message": "Webservice ChannelMaintenanceRequest for Request
Type DELETE successful!",
    "TotalResultCount": 1
  }
}}
```

- VIEW**

This request type is invoked every time a user needs to view data in the table 'Channel' (CHNL). User can use the field columns of the Channel table during search by indicating the value in the request.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
ChannelMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="5002"/>
  <RequestType Value="VIEW"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <ChnlID Value="3"/>
  <!--ChnlName Value="Channel"/-->
  <ChnlEndDt Value="06/20/2019"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TotalSearchCount": 1,
    "CustomerProfileView": {
      "TransactionID": 5002,
      "MarketingProgramID": 23,
      "ChnlID": 3,
      "ChnlName": "Channel C",
      "ChnlTypeCD": "CH_C",
      "ChnlStrtDt": "01/20/2019 00:00:00",
      "ChnlEndDt": "06/20/2019 00:00:00",
      "ChnlCpctyQty": "10.0000",
      "UOMCd": ""
    }
  }
}}
```

Web Service Value Notes

Value notes are as per current metadata configured in the tables

WS_REQ_ATT_MDL_COL_MAP (Add, Update, Delete) and WS_REQ_TYPE_MAP (View).

PromotionMaintenanceRequest

This web service allows the user to perform request types in the Promotion table. This Web service has the following request types:

- **ADD**

This request type is invoked every time a user needs to add data in the table 'Promotion' (PROMOTN). When adding data, the Campaign ID from Campaign table should be existing and must be passed in the request. The promotion id is a generated sequence in the system which is not required to be included during the request.

Sample Rest Request/Response

- **Resource:**

/mdm/xcorexml/services/CDHServices/rules/
PromotionMaintenanceRequest?response=application/json

- **Request:**

```
<root>
  <TransactionID Value="5000"/>
  <RequestType Value="ADD"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <CampaignID Value="1"/>
  <PrmtnDsc Value="Prom FF"/>
  <PrmtnStrtDt Value="01/20/2019"/>
  <PrmtnEndDt Value="12/20/2020"/>
  <PrmtnGlamnt Value="20"/>
```

```

    <AmmntCrncyCD Value="PHP"/>
    <PrmtnActUnitCnt Value="2"/>
    <PrmtnTypeCD Value="Prom_FF"/>
  </root>

```

- **Response**

```

{"RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 5000,
    "EE_Code": 10,
    "EE_Message": "Webservice PromotionMaintenanceRequest for
Request Type ADD successful!",
    "TotalResultCount": 1
  }
}}

```

- **UPDATE**

This request type is invoked every time a user needs to modify/update data in the table 'Promotion' (PROMOTN). User is required to pass the primary key of the table to update a record successfully.

Sample Rest Request/Response

- **Resource:**

```

/mdm/xcorexml/services/CDHServices/rules/
PromotionMaintenanceRequest?response=application/json

```

- **Request:**

```

<root>
  <TransactionID Value="5000"/>
  <RequestType Value="UPDATE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <CampaignID Value="1"/>
  <PromotionID Value="7"/>
  <PrmtnDsc Value="Promotion 7 Edit"/>
  <AmmntCrncyCD Value="KRW"/>
</root>

```

- **Response**

```

{"RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 5000,
    "EE_Code": 10,
    "EE_Message": "Webservice PromotionMaintenanceRequest for
Request Type UPDATE successful!",
    "TotalResultCount": 1
  }
}}

```

- **DELETE**

This request type is invoked every time a user needs to delete data in the table 'Promotion' (PROMOTN). Deletion in this table is updating the Promotion End Date to the current system date. User is required to pass the primary key to delete the record successfully.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
PromotionMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="5000"/>
  <RequestType Value="DELETE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <PromotionID Value="5"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 5000,
    "EE_Code": 10,
    "EE_Message": "Webservice PromotionMaintenanceRequest for
Request Type DELETE successful!",
    "TotalResultCount": 1
  }
}}
```

- **VIEW**

This request type is invoked every time a user needs to view data in the table 'Promotion' (PROMOTN). User can use the field columns of the Promotion table during search by indicating the value in the request.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
PromotionMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="5000"/>
  <RequestType Value="VIEW"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
```

```
<CampaignID Value="10"/>
<PromotionID Value="4"/>
</root>
```

- **Response**

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TotalSearchCount": 1,
    "CustomerProfileView": {
      "MarketingProgramID": 23,
      "TransactionID": 5000,
      "PromotionID": 4,
      "CampaignID": 10,
      "PrmtnDsc": "Prom C",
      "PrmtnStrtDt": "05/20/2019 00:00:00",
      "PrmtnEndDt": "11/20/2019 00:00:00",
      "PrmtnGlamnt": "20.0000",
      "AmmntCrncyCD": "JPY",
      "PrmtnActUnitCnt": 3,
      "PrmtnTypeCD": "Prom_C"
    }
  }
}}
```

Web Service Value Notes

Value notes are as per current metadata configured in the tables

WS_REQ_ATT_MDL_COL_MAP (Add, Update, Delete) and WS_REQ_TYPE_MAP (View).

CampaignMaintenanceRequest

This web service allows the user to perform request types in the Campaign table. This Web service has the following request types:

- **ADD**

This request type is invoked every time a user needs to add data in the table 'Campaign' (CAMPN). The campaign id is a generated sequence in the system and is not required to be included during the request.

Sample Rest Request/Response

- **Resource:**

```
/mdm/xcorexml/services/CDHServices/rules/
CampaignMaintenanceRequest?response=application/json
```

- **Request:**

```
<root>
  <TransactionID Value="5002"/>
  <RequestType Value="ADD"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
```

```
<BrandCD Value="KNOWN"/>
<CmpgnDsc Value="Camp January"/>
<CmpgnStrtDt Value="01/20/2019"/>
<CmpgnEndDt Value="12/25/2020"/>
<CmpgnStrgyCD Value="CD01"/>
<CmpgnName Value="CampJan"/>
<CmpgnTypeCD Value="TypeA"/>
<CurrencyCD Value="PHP"/>
</root>
```

- **Response**

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 5002,
    "EE_Code": 10,
    "EE_Message": "Webservice CampaignMaintenanceRequest for
Request Type ADD successful!",
    "TotalResultCount": 1
  }
}}
```

- **UPDATE**

This request type is invoked every time a user needs to modify/update data in the table 'Campaign' (CAMPN). User is required to pass the primary key of the table to update a record successfully.

Sample Rest Request/Response

- **Resource:**

/mdm/xcorexml/services/CDHServices/rules/
CampaignMaintenanceRequest?response=application/json

- **Request:**

```
<root>
  <TransactionID Value="5002"/>
  <RequestType Value="UPDATE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <CampaignID Value="2"/>
  <CmpgnDsc Value="Campaign February"/>
  <CurrencyCD Value="KRW"/>
</root>
```

- **Response**

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 5002,
    "EE_Code": 10,
    "EE_Message": "Webservice CampaignMaintenanceRequest for
Request Type UPDATE successful!",
    "TotalResultCount": 1
  }
}}
```

```
    }
  }}
}
```

- **DELETE**

This request type is invoked every time a user needs to delete data in the table 'Campaign' (CAMPN). Deletion in this table is updating the Campaign End Date to the current system date. User is required to pass the primary key to delete the record successfully.

Sample Rest Request/Response

- Resource:

```
/mdm/xcorexml/services/CDHServices/rules/
CampaignMaintenanceRequest?response=application/json
```

- Request:

```
<root>
  <TransactionID Value="5002"/>
  <RequestType Value="DELETE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <CampaignID Value="1"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 5002,
    "EE_Code": 10,
    "EE_Message": "Webservice CampaignMaintenanceRequest for
Request Type DELETE successful!",
    "TotalResultCount": 1
  }
}}
```

- **VIEW**

This request type is invoked every time a user needs to view data in the table 'Campaign' (CAMPN). User can use the field columns of the Promotion table during search by indicating the value in the request.

Sample Rest Request/Response

- Resource:

```
/mdm/xcorexml/services/CDHServices/rules/
CampaignMaintenanceRequest?response=application/json
```

- Request:

```
<root>
  <TransactionID Value="5002"/>
  <RequestType Value="VIEW"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
```

```
<SourceProviderCD Value="AIMIA"/>
<MarketingProgramID Value="23"/>
<BrandCD Value="KNOWN"/>
<CampaignID Value="2"/>
</root>
```

- **Response**

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TotalSearchCount": 1,
    "CustomerProfileView": {
      "MarketingProgramID": 23,
      "TransactionID": 5002,
      "CampaignID": 2,
      "CmpgnDsc": "Campaign February",
      "CmpgnStrtDt": "02/20/2019 00:00:00",
      "CmpgnEndDt": "08/10/2019 00:00:00",
      "CmpgnStrgyCD": "CD01",
      "CmpgnName": "CampB",
      "CmpgnTypeCD": "TypeA",
      "CurrencyCD": "KRW"
    }
  }
}}
```

Web Service Value Notes

Value notes are as per current metadata configured in the tables

WS_REQ_ATT_MDL_COL_MAP (Add, Update, Delete) and WS_REQ_TYPE_MAP (View).

UALMaintenanceRequest

This web service allows the user to perform request types in the Universal Authorization table. This Web service has the following request types:

- **ADD**

This request type is invoked every time a user needs to add data in the tables 'Universal Authorization' (UNVRSL_AUTHZTN), 'Universal Authorization Language Text' (UNVRSL_AUTHZTN_LANG_TXT) or 'Universal Authorization Application' (UNVRSL_AUTHZTN_APLCTN). It is not mandatory to add data in all three tables. Not providing the data on the Language Text (or Application) will still end to a successful transaction. Universal Authorization parameters must be provided. If the data is already in the system, other data will be persisted in the corresponding tables.

Sample Rest Request/Response

- **Resource:**

```
/mdm/xcorexml/services/CDHServices/rules/
UALMaintenanceRequest?response=application/json
```

- **Request:**


```

<root>
  <TransactionID Value="02061328"/>
  <RequestType Value="ADD"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <UnivAuthCd Value="UA_CD_LV0206"/>
  <UAStrtdt Value="01/01/2019"/>
  <UAMacNum Value="MN_EN-0206"/>
  <MktgPrgTypeCD Value="TYPEA"/>
  <BrandCD Value="KNOWN"/>
  <UATxtLangCD Value="EN"/>
  <UATxt Value="English"/>
  <UALangTxtStrtdt Value="01/01/2019"/>
  <SourceProviderCD Value="SRC_A"/>
  <ChannelID Value="111"/>
  <MarketingProgramID Value="3"/>
  <UAApStrtdt Value="03/01/2019"/>
</root>

```

- **Response**

```

{"RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "EE_Code": 10,
    "EE_Message": "Webservice UALMaintenanceRequest for Request
Type ADD successful!",
    "TotalResultCount": 1
  }
}}

```

- **UPDATE**

This request type is invoked every time a user needs to modify/update data in the tables 'Universal Authorization' (UNVRSL_AUTHZTN), 'Universal Authorization Language Text' (UNVRSL_AUTHZTN_LANG_TXT) or 'Universal Authorization Application' (UNVRSL_AUTHZTN_APLCTN). User is required to pass the primary key of the table to update a record successfully. Some column names may be present in both the tables, the user must correctly identify the specific parameter to be used to update the correct table.

Sample Rest Request/Response

- **Resource:**

```

/mdm/xcorexml/services/CDHServices/rules/
UALMaintenanceRequest?response=application/json

```

- **Request:**

```

<root>
  <TransactionID Value="02061428"/>
  <RequestType Value="UPDATE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <UnivAuthCd Value="UA_CD_LV0206"/>
  <UAMacNum Value="MAC_NUM_EN-101"/>
  <UATxt_UA Value="English UK"/>
  <UATxtLangCD_UAL Value="EN"/>
  <UATxt_UAL Value="English UK"/>
</root>

```

- **Response**

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "EE_Code": 10,
    "EE_Message": "Webservice UALMaintenanceRequest for Request
Type UPDATE successful!",
    "TotalResultCount": 1
  }
}}
```

- **DELETE**

This request type is invoked every time a user needs to modify/update data in the tables 'Universal Authorization' (UNVRSL_AUTHZTN), 'Universal Authorization Language Text' (UNVRSL_AUTHZTN_LANG_TXT) or 'Universal Authorization Application'(UNVRSL_AUTHZTN_APLCTN). User is required to pass the primary key of the table to update a record successfully.

Sample Rest Request/Response

- **Resource:**

/mdm/xcorexml/services/CDHServices/rules/
UALMaintenanceRequest?response=application/json

- **Request:**

```
<root>
  <TransactionID Value="02051445"/>
  <RequestType Value="DELETE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <UnivAuthCd Value="UA_CD_LV0206"/>
</root>
```

- **Response**

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "EE_Code": 10,
    "EE_Message": "Webservice UALMaintenanceRequest for Request
Type DELETE successful!",
    "TotalResultCount": 1
  }
}}
```

- **VIEW**

This request type is invoked every time a user needs to view data in the tables 'Universal Authorization' (UNVRSL_AUTHZTN), 'Universal Authorization Language Text' (UNVRSL_AUTHZTN_LANG_TXT).

Sample Rest Request/Response

- **Resource:**

/mdm/xcorexml/services/CDHServices/rules/
UALMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="5001"/>
  <RequestType Value="VIEW"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <UnivAuthCd Value="UA_CD_LV0206"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TotalSearchCount": 1,
    "CustomerProfileView": {
      "TransactionID": 5001,
      "UniversalAuthorizationApp": {
        "UAppStrtDt": "03/01/2019 00:00:00",
        "UAppEndDt": "02/06/2019 00:00:00",
        "ChannelID": 111
      },
      "UnivAuthCd": "UA_CD_LV0206",
      "UniversalAuthorizationLang": [
        {
          "UATxtLangCD": "FR",
          "UALangTxtStrtDt": "01/01/2019 00:00:00",
          "UATxt": "French",
          "UALangBrandCD": "KNOWN"
        },
        {
          "UATxtLangCD": "EN",
          "UALangTxtStrtDt": "01/01/2019 00:00:00",
          "UATxt": "English UK",
          "UALangBrandCD": "KNOWN"
        }
      ],
      "UAStrtDt": "01/01/2019 00:00:00",
      "UAEndDt": "02/06/2019 14:46:09:003",
      "UMacNum": "MAC_NUM_EN-101",
      "UATxt": "English UK",
      "BrandCD": "KNOWN",
      "UALangBrandCD": "KNOWN"
    }
  }
}
```

Web Service Value Notes

Value notes are as per current metadata configured in the tables

WS_REQ_ATT_MDL_COL_MAP (Add, Update, Delete) and WS_REQ_TYPE_MAP (View).

TraitMaintenanceRequest

This web service allows the user to perform request types in the Trait table. There are four tables involved in this web service.

- Trait - this is the main table of Trait.
- Three sub tables (Trait Text, Trait Valid Value and Marketing Program Trait) that are not related to each other but is related to the main table.

This Web service has the following request types:

- **ADD**

This request type is invoked every time a user needs to add data in the main table 'Trait' (TRAT), or the sub tables 'Trait Text' (TRAT_TXT), 'Trait Valid Value' (TRAT_VLD_VAL) or 'Marketing Program Trait' (MKTG_PRG_TRAT). If the data in the main table is already existing, then this will not the data again. If there are other input fields passed in the request for the sub tables, then the request will process add for the next tables.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
TraitMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="5000"/>
  <RequestType Value="ADD"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <TraitCD Value="AAA_01"/>
  <TraitGrpCD Value="TG01"/>
  <TraitName Value="First_Trait"/>
  <TraitDesc Value="First_Trait"/>
  <TraitValueTypeCD Value="TV01"/>
  <TraitOwnerCD Value="TO01"/>
  <!-- trait text -->
  <TraitText Value="TTextA"/>
  <TraitTextStrtDT Value="01/20/2019"/>
  <TraitTextEndDT Value="11/30/2019"/>
  <TraitNativeLangTxt Value="EN"/>
  <TraitNativeLangCD Value="TNA"/>
  <!-- trait valid value -->
  <TraitRangeStrtVal Value="01/25/2019"/>
  <TraitRangeEndVal Value="12/31/2019"/>
  <TraitValueDt Value="02/02/2019"/>
  <TraitValueNum Value="10"/>
  <TraitValueCD Value="TVV01"/>
</root>
```

- Response

```
{"RESPONSES": {
```

```

    "Status": "Success",
    "RESPONSE": {
      "Status": "Success",
      "TransactionID": 5000,
      "EE_Code": 10,
      "EE_Message": "Webservice TraitMaintenanceRequest for Request
Type ADD successful!",
      "TotalResultCount": 1
    }
  }
}

```

- **UPDATE**

This request type is invoked every time a user needs to modify/update data in the table 'Trait' (TRAT) or its sub tables. User is required to pass the primary key of the main table or the sub table to update a record successfully.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
TraitMaintenanceRequest?response=application/json

- Request:

```

<root>
  <TransactionID Value="5000"/>
  <RequestType Value="UPDATE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <BrandCD Value="KNOWN"/>
  <TraitCD Value="AAA_01"/>
  <TraitGrpCD Value="TG01"/>
  <TraitName Value="First_Trait"/>
  <TraitDesc Value="First Updated Trait"/>
</root>

```

- Response

```

{"RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "EE_Code": 11,
    "EE_Message": "Record/s in table/s TRAT successfully updated. No
record/s updated in table/s TRAT_TXT,TRAT_VLD_VAL,MKTG_PRG_TRAT for
Webservice TraitMaintenanceRequest Request Type UPDATE.",
    "TotalResultCount": 1
  }
}
}

```

- **DELETE**

This request type is invoked every time a user needs to delete data in the table 'Trait' (TRAT). User is required to pass the primary key of the main table only to delete a record successfully. Sub table data will be deleted together if they are related to the main table.

If the table does not have an end date field, the ENTITY_STATE will be updated to 'DELETED'.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
TraitMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="5000"/>
  <RequestType Value="DELETE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <TraitCD Value="AAA_01"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "EE_Code": 11,
    "EE_Message": "Record/s in table/s TRAT,TRAT_TXT,TRAT_VLD_VAL
successfully deleted. No record/s deleted in table/s MKTG_PRG_TRAT for
Webservice TraitMaintenanceRequest Request Type DELETE.",
    "TotalResultCount": 1
  }
}}
```

- **VIEW**

This request type is invoked every time a user needs to view data in the table 'Trait' (TRAT). User can only use the search fields of the main table, but the response will display the sub tables related to it.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
TraitMaintenanceRequest?response=application/json

- Request:

```
<root>
  <TransactionID Value="5009"/>
  <RequestType Value="VIEW"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="23"/>
  <BrandCD Value="KNOWN"/>
  <TraitCD Value="AAA_01"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
```

```
"RESPONSE": {
  "Status": "Success",
  "TotalSearchCount": 1,
  "CustomerProfileView": {
    "TransactionID": 5009,
    "TraitValidValue": [
      {
        "TraitValueCD": "TVV01",
        "TraitRangeStrtVal": "01/25/2019",
        "TraitRangeEndVal": "12/31/2019",
        "TraitValueDt": "02/02/2019",
        "TraitValueNum": 10
      },
      {
        "TraitValueCD": "TVV02",
        "TraitRangeStrtVal": "01/25/2019",
        "TraitRangeEndVal": "12/31/2019",
        "TraitValueDt": "02/02/2019",
        "TraitValueNum": 10
      },
      {
        "TraitValueCD": "TVV03",
        "TraitRangeStrtVal": "01/25/2019",
        "TraitRangeEndVal": "12/31/2019",
        "TraitValueDt": "04/20/2019",
        "TraitValueNum": 10
      },
      {
        "TraitValueCD": "TVV04",
        "TraitRangeStrtVal": "01/25/2019",
        "TraitRangeEndVal": "12/31/2019",
        "TraitValueDt": "04/20/2019",
        "TraitValueNum": 10
      },
      {
        "TraitValueCD": "TVV05",
        "TraitRangeStrtVal": "01/25/2019",
        "TraitRangeEndVal": "12/31/2019",
        "TraitValueDt": "04/20/2019",
        "TraitValueNum": 10
      }
    ],
    "TraitTextData": [
      {
        "TraitText": "TTextB",
        "TraitTextStrtDT": "01/24/2019 00:00:00",
        "TraitTextEndDT": "02/04/2019 00:00:00",
        "TraitNativeLangTxt": "KR",
        "TraitNativeLangCD": "TNB"
      },
      {
        "TraitText": "TTextB",
        "TraitTextStrtDT": "01/21/2019 00:00:00",
        "TraitTextEndDT": "02/04/2019 00:00:00",
        "TraitNativeLangTxt": "KR",
        "TraitNativeLangCD": "TNB"
      },
      {
        "TraitText": "TTextB",

```

```

        "TraitTextStrtDT": "01/18/2019 00:00:00",
        "TraitTextEndDT": "02/04/2019 00:00:00",
        "TraitNativeLangTxt": "KR",
        "TraitNativeLangCD": "TNB"
    },
    {
        "TraitText": "TTextA",
        "TraitTextStrtDT": "01/17/2019 00:00:00",
        "TraitTextEndDT": "02/04/2019 00:00:00",
        "TraitNativeLangTxt": "EN",
        "TraitNativeLangCD": "TNA"
    }
],
"TraitCD": "AAA_01",
"MarketingProgramTrait": [
    {
        "TraitCd": "AAA_01",
        "MarketingProgramID": 2,
        "TraitGroupCdM": "MGroup2",
        "TraitNameM": "MTname2",
        "TraitDescM": "MTname2",
        "TraitValueTypeCdM": "TV01",
        "TraitOwnerCdM": "TO01"
    },
    {
        "TraitCd": "AAA_01",
        "MarketingProgramID": 1,
        "TraitGroupCdM": "MGroup1",
        "TraitNameM": "MTname1",
        "TraitDescM": "MTname1",
        "TraitValueTypeCdM": "TV01",
        "TraitOwnerCdM": "TO01"
    }
],
"TraitGrpCD": "TG01",
"TraitName": "First_Trait",
"TraitDesc": "First Updated Trait Feb",
"TraitValueTypeCD": "TV01",
"TraitOwnerCD": "TO01"
}
}
}}

```

Web Service Value Notes

Value notes are as per current metadata configured in the tables

WS_REQ_ATT_MDL_COL_MAP (Add, Update, Delete) and WS_REQ_TYPE_MAP (View).

Update Web Services

Currently configured and implemented update web services includes the following:

- [CustManageLyltyAccount](#)
- [CustManagePymntAccount](#)

- [CustManageTrait](#)

Archiving

Update web services use the archiving functionality of the system. Any add, update or delete request in the web service, a corresponding record is inserted in the PERSONA tables. Archiving is configured in the WS_MDL_MAP metadata table and can be turned off anytime. Common PERSONA tables where data is inserted are:

- Persona
- Customer Persona

CustManageLyltyAccount

This web service allows the user to perform request types in the Loyalty Account table. Archiving of this web service inserts data in the common persona tables and persona loyalty account (PRSNA_LYLTY_ACCT) table. This Web service has the following request types:

- **ADD**

This request type is invoked every time a user needs to add data in the table 'Loyalty Account' (LYLTY_ACCT).

Sample Rest Request/Response

- Resource:

```
/mdm/xcorexml/services/CDHServices/rules/  
CustManageLyltyAccount?response=application/json
```

- Request:

```
<root>  
  <TransactionID Value="5009"/>  
  <RequestType Value="ADD"/>  
  <UserName Value="admin"/>  
  <Password Value="admin"/>  
  <SourceProviderCD Value="AIMIA"/>  
  <MarketingProgramID Value="1"/>  
  <BrandCD Value="KNOWN"/>  
  <CustID Value="6"/>  
  <LoyaltyAccountNum Value="L000001"/>  
  <LoyaltyProgramId Value="1"/>  
  <FirstName Value="Dan"/>  
  <MiddleName Value="Ni"/>  
  <LastName Value="Lou"/>  
  <Gender Value="Male"/>  
  <DOB Value="10/18/1980"/>  
  <LoyaltyLevelCd Value="1"/>  
  <LoyaltyAccountStartDt Value="01/22/2019"/>  
  <LoyaltyAccountEndDt Value="01/22/2020"/>  
  <LoyaltyPoints Value="1001"/>  
  <LoyaltyStatusCode Value="B_01"/>  
</root>
```

- Response

```
{"RESPONSES": {
```

```

    "Status": "Success",
    "RESPONSE": {
      "Status": "Success",
      "TransactionID": 5009,
      "EE_Code": 10,
      "EE_Message": "Webservice CustManageLyltyAccount  for Request
Type ADD successful!",
      "TotalResultCount": 1
    }
  }
}

```

- **UPDATE**

This request type is invoked every time a user needs to modify/update data in the table 'Loyalty Account' (LYLTY_ACCT). User is required to pass the primary key of the table to update a record successfully.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
CustManageLyltyAccount?response=application/json

- Request:

```

<root>
  <TransactionID Value="5010"/>
  <RequestType Value="UPDATE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="1"/>
  <BrandCD Value="KNOWN"/>
  <CustID Value="25"/>
  <LoyaltyAccountNum Value="L000001"/>
  <LoyaltyProgramId Value="1"/>
  <LoyaltyLevelCd Value="2"/>
  <LoyaltyPoints Value="2000"/>
</root>

```

- Response

```

{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 5009,
    "EE_Code": 10,
    "EE_Message": "Webservice CustManageLyltyAccount  for Request
Type UPDATE successful!",
    "TotalResultCount": 1
  }
}
}

```

- **DELETE**

This request type is invoked every time a user needs to delete data in the table 'Loyalty Account' (LYLTY_ACCT). Deletion in this table is updating the Loyalty Account End Date to the current system date. User is required to pass the primary key to delete the record successfully.

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
CustManageLyltyAccount?response=application/json

- Request:

```
<root>
  <TransactionID Value="7000"/>
  <RequestType Value="DELETE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="1"/>
  <BrandCD Value="KNOWN"/>
  <CustID Value="59"/>
  <LoyaltyAccountNum Value="L000001"/>
  <LoyaltyProgramId Value="1"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 7000,
    "EE_Code": 10,
    "EE_Message": "Webservice CustManageLyltyAccount for Request
Type DELETE successful!",
    "TotalResultCount": 1
  }
}}
```

Web Service Value Notes

Value notes are as per current metadata configured in the tables
WS_REQ_ATT_MDL_COL_MAP.

CustManagePymntAccount

This web service allows the user to perform request types in the Payment Account and Customer Payment Account tables. Archiving of this web service inserts data in the common persona tables and persona payment account (PRSNA_PMT_ACCT) table.

- ADD**

This request type is invoked every time a user needs to add data in the tables Payment Account (PMT_ACCT) and Customer Payment Account (CUST_PMT_ACCT).

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/
CustManagePymntAccount?response=application/json

- Request:

```
<root>
```

```
<TransactionID Value="333"/>
<RequestType Value="ADD"/>
<UserName Value="admin"/>
<Password Value="admin"/>
<CustID Value="265"/>
<PymntAccountID Value=""/>
<PymntAccountNum Value="302"/>
<PymntAccountTypeCD Value="PT_IND"/>
<PymntAccountSubTypeCD Value="PST_IND_S"/>
<PymntAcctCurrCD Value="USD"/>
<PymntAcctCOAID Value="12345"/>
<CreditRatingCD Value="CR_501"/>
<PymntAcctIssueDt Value="01/01/2019"/>
<PymntAcctExprDt Value="01/01/2030"/>
<SetOfBooksCD Value= "SB_S"/>
  <CustPymntAcctStrtDT Value="01/01/2019"/>
  <CustPymntAcctUsageCD Value= "AUTOPAY"/>
</root>
```

- **Response**

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "EE_Code": 10,
    "EE_Message": "Webservice CustManagePymntAccount for Request
Type ADD successful!",
    "TotalResultCount": 1
  }
}}
```

- **UPDATE**

This request type is invoked every time a user needs to modify/update data in the tables Payment Account (PMT_ACCT) and Customer Payment Account (CUST_PMT_ACCT). User is required to pass the primary key of the table to update a record successfully.

Sample Rest Request/Response

- **Resource:**

```
/mdm/xcorexml/services/CDHServices/rules/
CustManagePymntAccount?response=application/json
```

- **Request:**

```
<root>
  <TransactionID Value="333"/>
  <RequestType Value="UPDATE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <PymntAccountID Value="302"/>
  <PymntAcctCurrCD Value="SGD"/>
  <PymntAccountTypeCD Value="PT_IND"/>
  <CustID Value="265"/>
  <CustPymntAcctUsageCD Value= "MANUAL PAY"/>
</root>
```

- **Response**

```
{ "RESPONSES": {
```

```

    "Status": "Success",
    "RESPONSE": {
      "Status": "Success",
      "EE_Code": 10,
      "EE_Message": "Webservice CustManagePymntAccount for Request
Type UPDATE successful!",
      "TotalResultCount": 1
    }
  }
}

```

- **DELETE**

This request type is invoked every time a user needs to delete data in the tables Payment Account (PMT_ACCT) and Customer Payment Account (CUST_PMT_ACCT). User is required to pass the primary key to delete the record successfully.

Sample Rest Request/Response

- Resource:

```

/mdm/xcorexml/services/CDHServices/rules/
CustManagePymntAccount?response=application/json

```

- Request:

```

<root>
  <TransactionID Value="02061608"/>
  <RequestType Value="DELETE"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <PymntAccountID Value="302"/>
  <CustID Value="265"/>
</root>

```

- Response

```

{"RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "EE_Code": 10,
    "EE_Message": "Webservice CustManagePymntAccount for Request
Type DELETE successful!",
    "TotalResultCount": 1
  }
}
}

```

Web Service Value Notes

Value notes are as per current metadata configured in the table WS_REQ_ATT_MDL_COL_MAP.

CustManageTrait

This web service allows the user to perform request types in the Customer Trait Value table. Archiving of this web service inserts data in the common persona tables and persona trait (PRSNA_TRAT) table.

- **ADD**

This request type is invoked every time a user needs to add data in the table 'Customer Trait Value' (CUST_TRAT_VAL).

Sample Rest Request/Response

- Resource:

/mdm/xcorexml/services/CDHServices/rules/CustManageTrait?response=application/json

- Request:

```
<root>
  <TransactionID Value="5009"/>
  <RequestType Value="ADD"/>
  <UserName Value="admin"/>
  <Password Value="admin"/>
  <SourceProviderCD Value="AIMIA"/>
  <MarketingProgramID Value="1"/>
  <BrandCD Value="KNOWN"/>
  <CustID Value="12"/>
  <TraitCD Value="AAA_01"/>
  <TraitSequenceId Value="1"/>
  <TraitVal Value="CustValA"/>
  <TraitValDt Value=""/>
  <TraitStartDt Value="01/20/2019"/>
  <TraitEndDt Value="12/31/2019"/>
</root>
```

- Response

```
{ "RESPONSES": {
  "Status": "Success",
  "RESPONSE": {
    "Status": "Success",
    "TransactionID": 5009,
    "EE_Code": 10,
    "EE_Message": "Webservice CustManageTrait for Request Type ADD
successful!",
    "TotalResultCount": 1
  }
}}
```

Web Service Value Notes

Value notes are as per current metadata configured in the table WS_REQ_ATT_MDL_COL_MAP.

Additional Information

This section describes the source code and error code descriptions.

Source Code Description

The [Table 5](#) provides the description for various source codes.

Table 5: Source Code Descriptions

Code	Description	Response Message/Note
10	Successful web service transaction.	Webservice <webservice_name> for Request Type <request_type> successful!

The [Table 6](#) provides the description for various error codes.

Table 6: Source Code Descriptions

Code	Description	Response Message/Note
100	Undefined error.	Error, Webservice <webservice_name> for Request Type <request_type>!
101	Invalid request type.	Error, Webservice <webservice_name> Request Type <request_type> invalid!
11	No records found to update/delete in some tables. Error code applicable for webservices with multiple tables being processed.	Record/s in table/s <table_names> successfully <updated/deleted>. No record/s <updated/deleted> in table/s <table_names> for Webservice <webservice_name> Request Type <request_type>.
12	No parameter sent by the request.	Error, No request parameter received for Webservice <webservice_name> Request Type <request_type>.
13	No records found to update/delete in ALL tables. Error code applicable for webservices with multiple tables being processed.	No record/s <updated/deleted> for Webservice <webservice_name> Request Type <request_type>.
14	During update, parameters received corresponds to multiple records in a table thus cannot update.	Cannot update multiple records in table <table_name> for Webservice <webservice_name>!
15	During update, parameters received corresponds to multiple records in a table thus cannot update.	Cannot update multiple records in table <table_name> for Webservice <webservice_name>! Record/s in table/s <table_name> successfully updated.
20	Incomplete/no meta data configuration.	Error! Meta Data not found for Webservice <webservice_name> Request Type <request_type>!
21	Invalid configuration setup for PK or FK fields.	Error! Error Meta Data setup for Webservice <webservice_name> Request Type <request_type> on PK or FK parameters!
30	Data to be inserted is already existing in the system.	Error! Data already in the system (PK value/s:<PK_data>)!

Table 6: Source Code Descriptions

Code	Description	Response Message/Note
31	FK value not defined in the referenced table.	Error! FK value for <field_name> cannot be found in <table_name> table!
32	PK value not defined in the referenced table.	Error! PK value for <field_name> cannot be found in <table_name> table!
40	Required request parameters not found.	Error! Required column/s <field_name> not found!

CHAPTER 6 **Connected Identity Web Analytics**

What's In This Chapter

This chapter provides information on connected identity web analytics.

Topics include:

- [Introduction](#)
- [Connected Identity Product](#)
- [Connected Identity UI Web Metrics](#)
- [Server Side Configurations](#)
- [Database Development](#)

Introduction

Connected Identity (CI) ingests data from web analytics and social media platforms to track user interactions using a cross reference of known and unknown identities. Connected identity provides the viewpoint on every interaction which a user might have with online and offline data.

The Internet is broken down into multiple connected websites which provide premium and free services which cater to their core audiences.

Any website functions to:

- **Attract traffic:** websites as part of their media strategy create campaigns by targeting customers who are already known to them using online or offline campaigns or target unknown users by acting as a buyer of ads, content or offline campaigns.
- **Provide service:** website captures users information (PII) and what transactions the user has performed.
- **Generate revenue:** websites can either provide their services for free or premium. If free services are provided it either generates revenue by serving ads or other paid content by being a publisher or providing offline services, which generates revenue in the website's core business.

To increase revenue, websites have two overarching objectives:

- Get new customers - acquisition
- Retain existing customers - retention

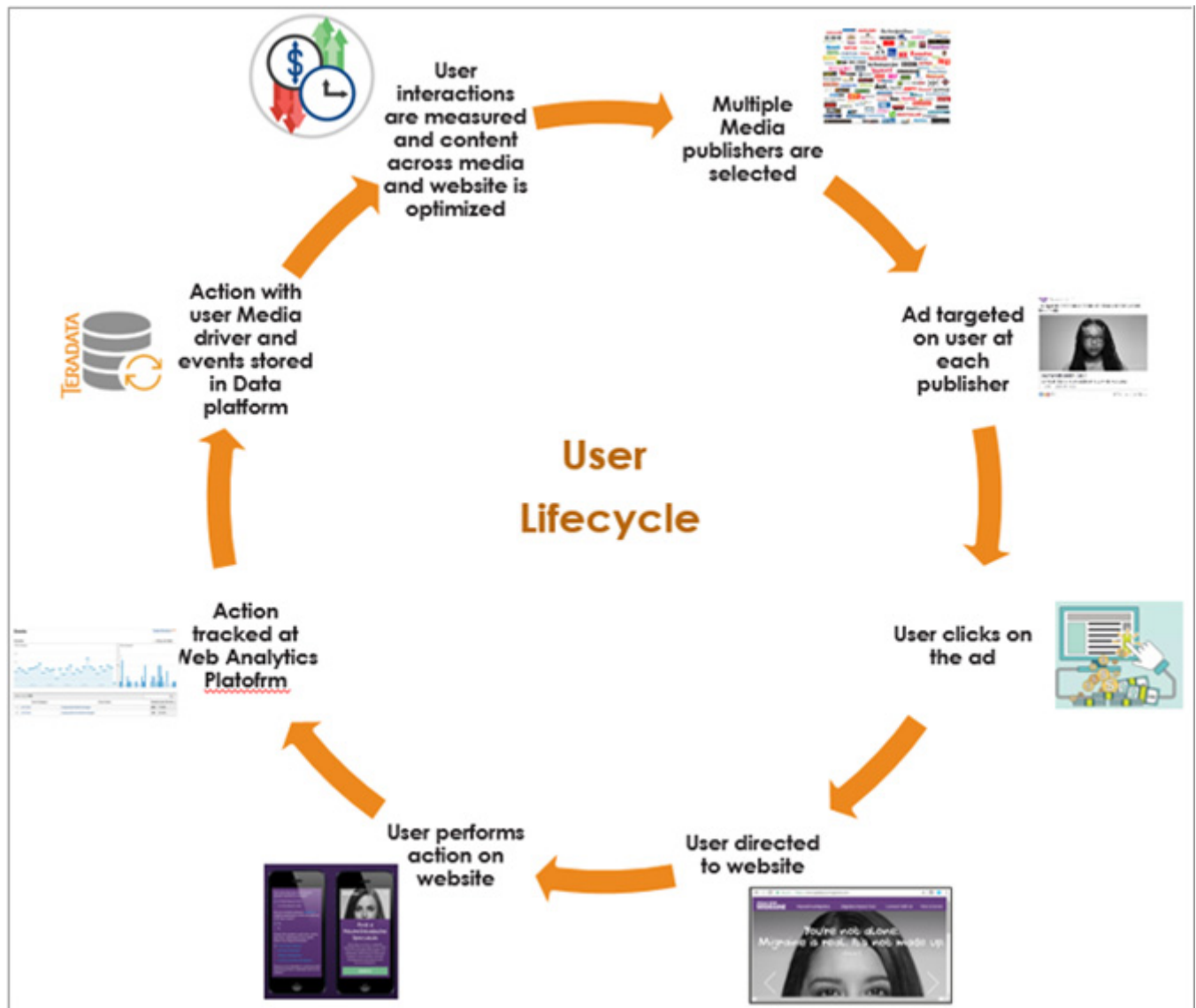
To add value, every website needs feedbacks such as:

- How customers interact with their content.

- What drove the customer to the website.
- What influenced the customer to generate revenue.
- How loyalty is influenced.

Figure 39 displays a depiction on how Web Site systems work.

Figure 39: How Web Site Works



There are multiple technologies which help websites get feedback:

- Websites Transaction Systems: Website transactional system captures user personal information, transactions performed on the system (buy, registration, wish-list). Websites users and websites offline customer data are mastered using CDI to link website and offline data.
- Web Analytics Platform: These platforms track user's clickstreams across the website using clicks or navigations made by the user. In addition, it tracks user sessions, time on

the website. It tracks typically at cookie level at the browser level. Platforms also allow cookie and website user to be linked allowing websites to measure how website traffic drives offline sales. This platform provides some factors to measure how the user interacts to the website content, what content influences revenue and how loyalty is influenced.

Providers: Google Analytics, Adobe Analytics, Web trends, Piwik, Kissmetrics

- Ad serving platforms: These platforms are responsible to place content or ads in relevant publisher's websites to send customers to purchaser websites who buy ads. These platforms have ad exchanges which connect publishers to purchasers based on the target audience, cost/revenue, ad quality and time of the year. These are responsible to increase traffic to websites.

Providers: Google Doubleclick, App Nexus, Rubicon, OpenX, Bidswitch.

Connected Identity Product

The Connected Identity product is a solution to connect web analytics platforms so that Teradata customers who store website data can analyze web traffic, web interactions and offline data within Teradata platform. The product provides the following key features:

- An adaptable framework with base functionality to maintain user (Known & Unknown) identities across systems.
- A Web UI which provides functionalities to manage identity merge, unmerge rules and provides metrics on identities.
- Connectors to various second and third party systems such as Google Analytics.

The below section provides description on the following:

- Configuration: How the Connected identity product needs to be initially configured.
 - New User configuration on what the user needs to perform when the product is first installed.
- Day in the life: What are the daily activities which can be performed in the connected identity product.
 - Web Metrics which user can access to get aggregate and detail metrics.
 - Existing user configuration on what metrics can be configured to fetch the data.
- ETL to represent how the data will flow once the configuration is complete.

Assumptions

- Google Analytics will be setup by the IT Admin or Business Analyst.
- GA is configured by the system owner to link Application User ID and Google client ID.
- Google Analytics Client_secret.json is downloaded and stored in Google Analytics Configuration folder which contains all the backend scripts for ETL and Authentication Process.

Terms Used

- GA- Google Analytics which is a web analytics platform.

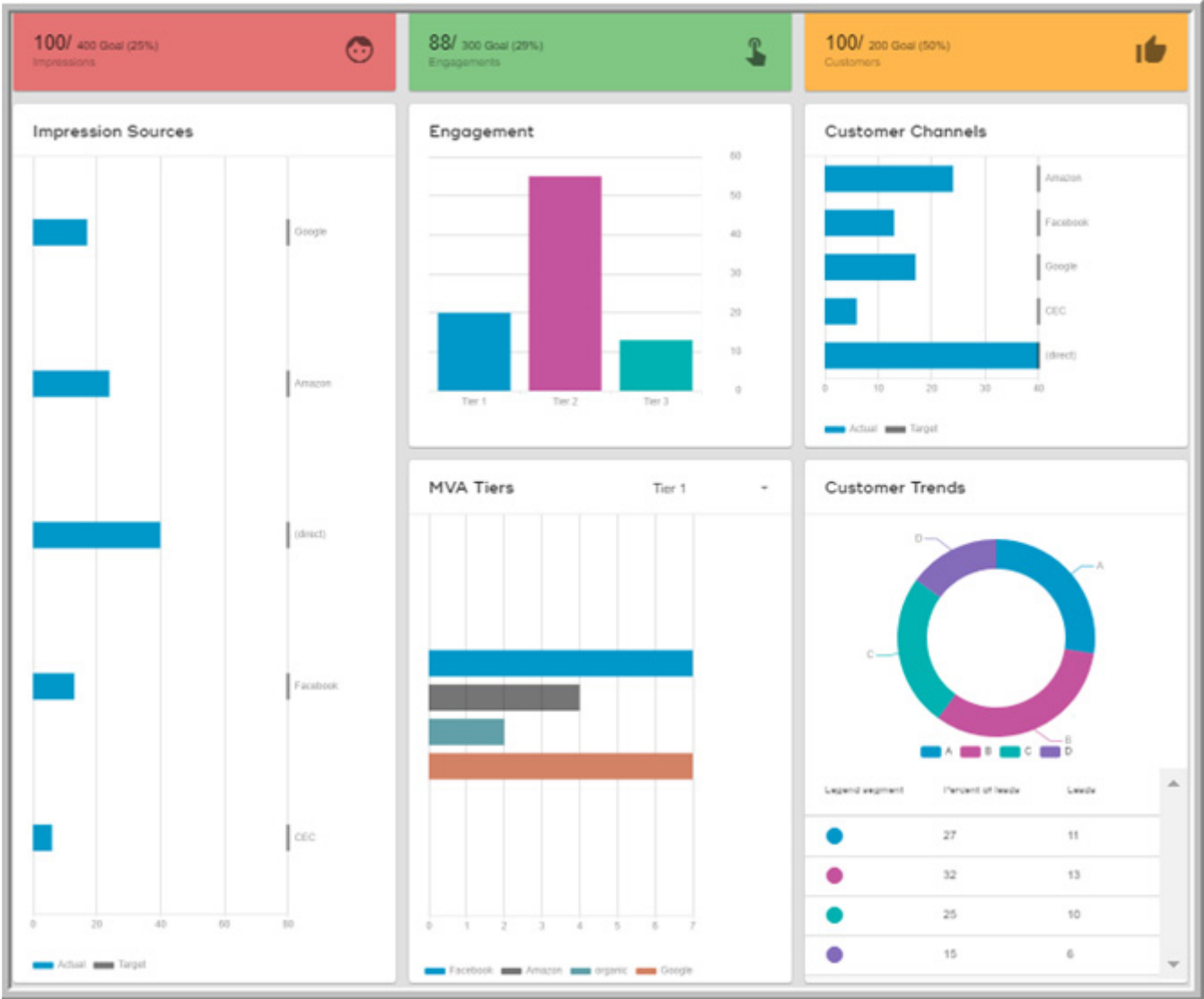
- **Account:** Is associated with the list of analytics accounts, which the user has access to, typically this is associated with a Domain area or client level.
- **View:** Each view is associated with a website within an account, there can be a view per environment such as test, production.

Connected Identity UI Web Metrics

MDM CI solution deals with customer interaction across different channels through different media like a call center, Website, etc and ensures that the various transactions are captured to create golden master records that can be used in Web Analytics. The Web Analytics provides information to the business clients about their customer interaction on their websites, source from where the customers are coming into client's websites, and the product the customers are interested in their websites and how valuable are the customers. These insights about the customers lead to further analytics to improve their business.

[Figure 40](#) displays the Web Analytics Summary UI. The Web Analytics Summary page displays information like how the customers are exposed to the different brands from different media, how the customers are engaged on the business website, and how many of the customers have given their consent. It also displays the count of customers from different channels.

Figure 40: Web Analytics Summary

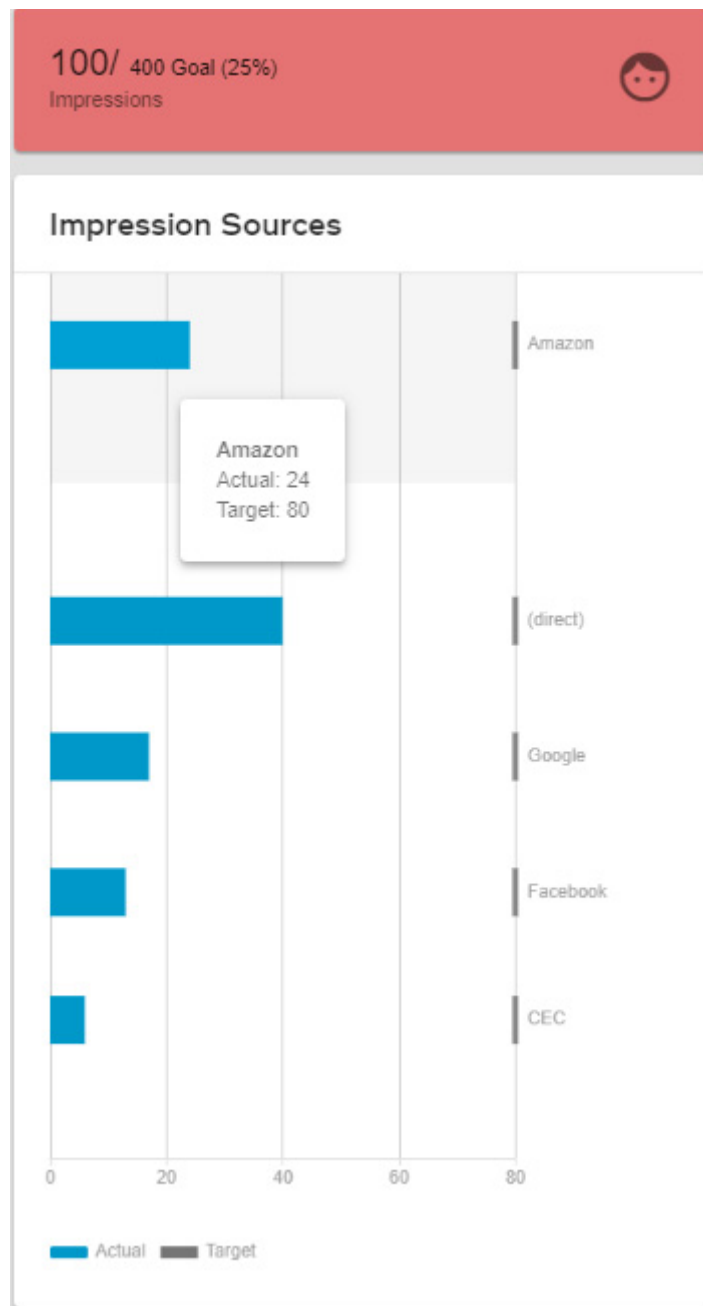


The below section describes each of the panels on the Web Analytics Summary UI.

- Impression Sources

Figure 41 displays the Impression Sources panel. The Impression Sources panel provides information on how many of the client's customers are actually exposed to their brands and how many of these branded content are shown across the different media like Youtube, Google Ad or search engine. In the upper panel, the count shows the number of customers who have seen the branded content from different media. For example: 100 impression means 100 people have seen this branded content from different media and the goal count shows the target count. The bar chart displays the different sources like Amazon, Google, etc and the actual and target impressions.

Figure 41: Web Analytics Summary—Impression Sources

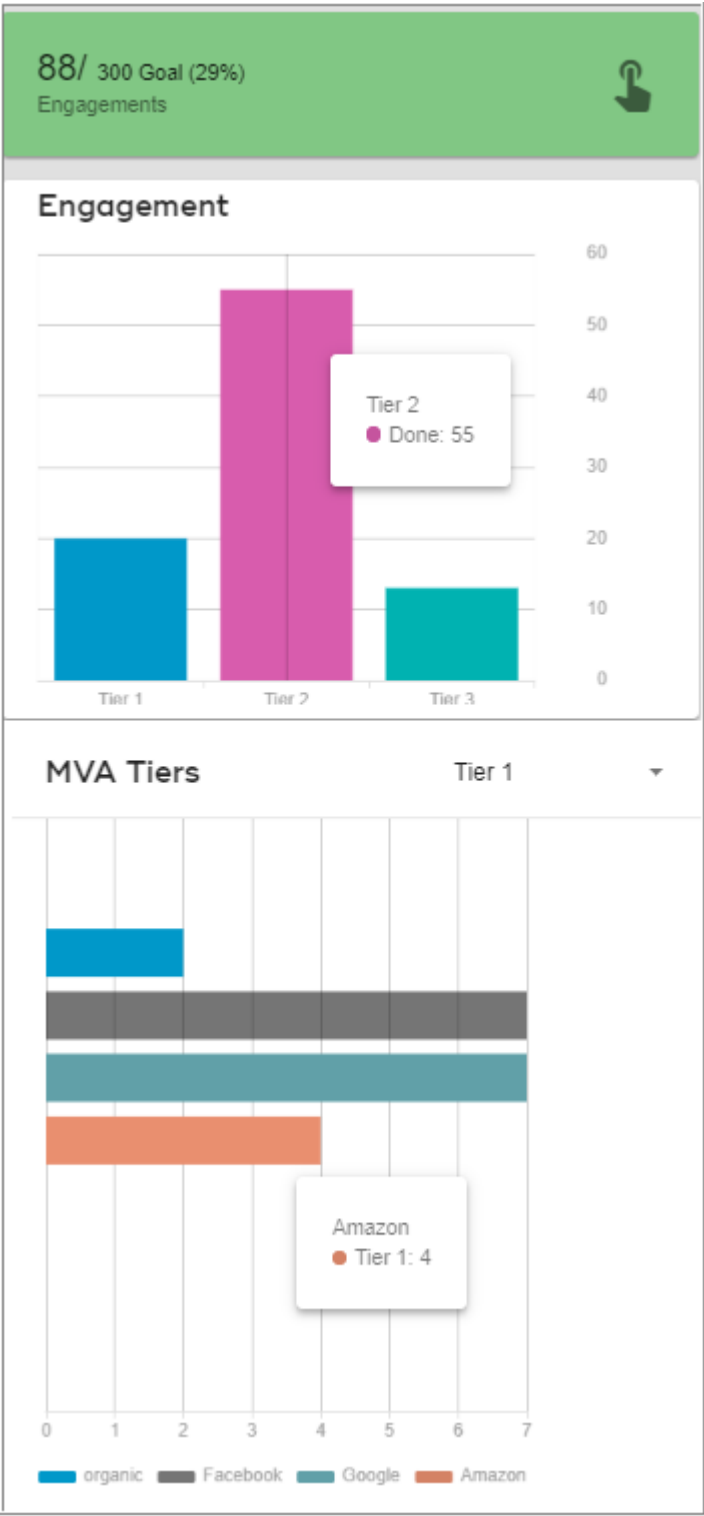


- Engagement and Market Value Added (MVA) Tiers

Figure 42 displays the Engagement and Tiers panel. The Engagement panel shows how engaged are the customers on the website on the branded content, how many brands are exposed to customers. The engagements are divided into multiple tiers.

Tiers are directly proportional to the value which an event provides in relation to business ROI. Based on the website content, it shows which tier has the highest engagement and which tier has the least engagements. The bottom line shows the different sources from where the customers have come. Tier activity shows how many of them have got value for that specific tier. Click on tier 1, tier 2 and it shows the different breakdowns.

Figure 42: Web Analytics Details—Engagement and Tiers



- Customer Channels and Customer Trends

Figure 43 displays the Customer Channel and Customer trend panel. It shows how many people were engaged through the website and have given consent to contact. It displays

the target and the actual count of customers for each of the channel. The Customer trend is based on segmentation. It shows the high value customers or low value customers based on the channel from which it comes in.

Figure 43: Web Analytics Details—Customer Channels and Customer Trends

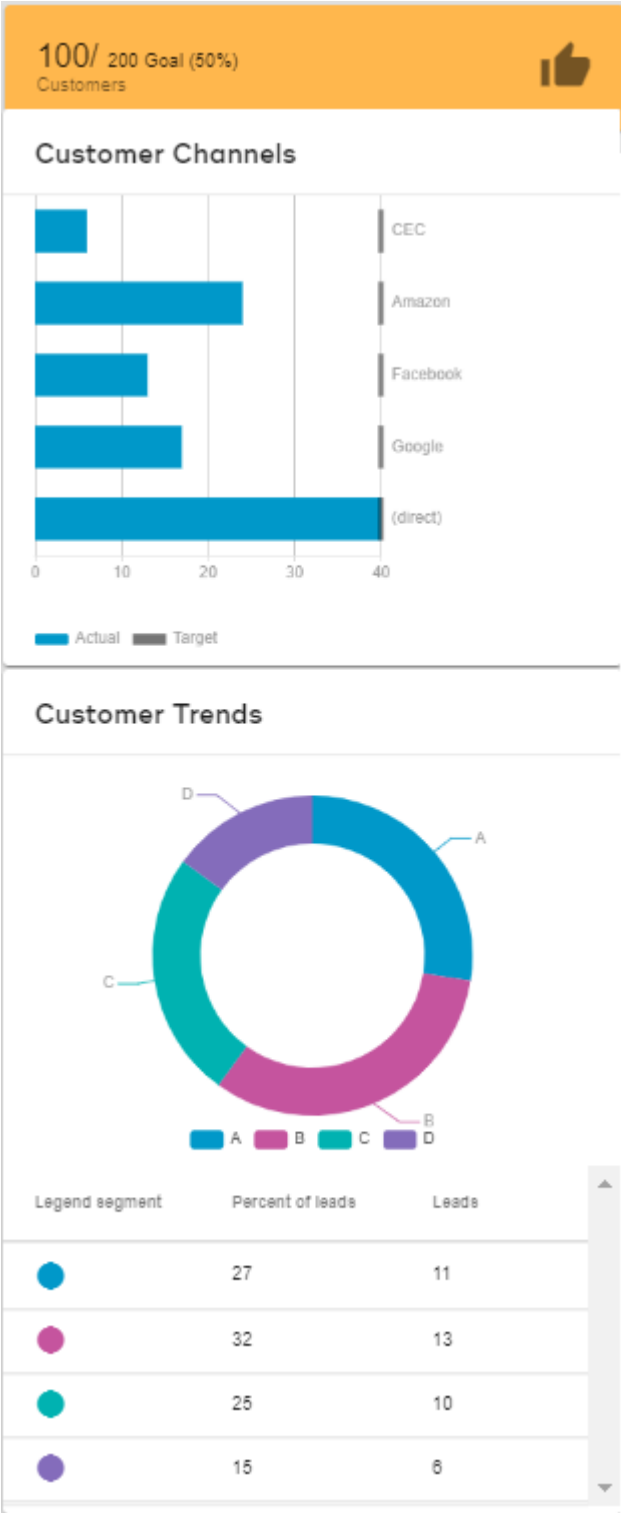


Figure 44 displays the Web Analytics Demographics Details. It shows the drill-down of the customer channels, that is it displays the location where the customer can actually be contacted and their breakdown. It also displays the Age range of the customers and the device type that they have used when they were actually engaging or while providing their consent.

Figure 44: Web Analytics Summary—Demographics

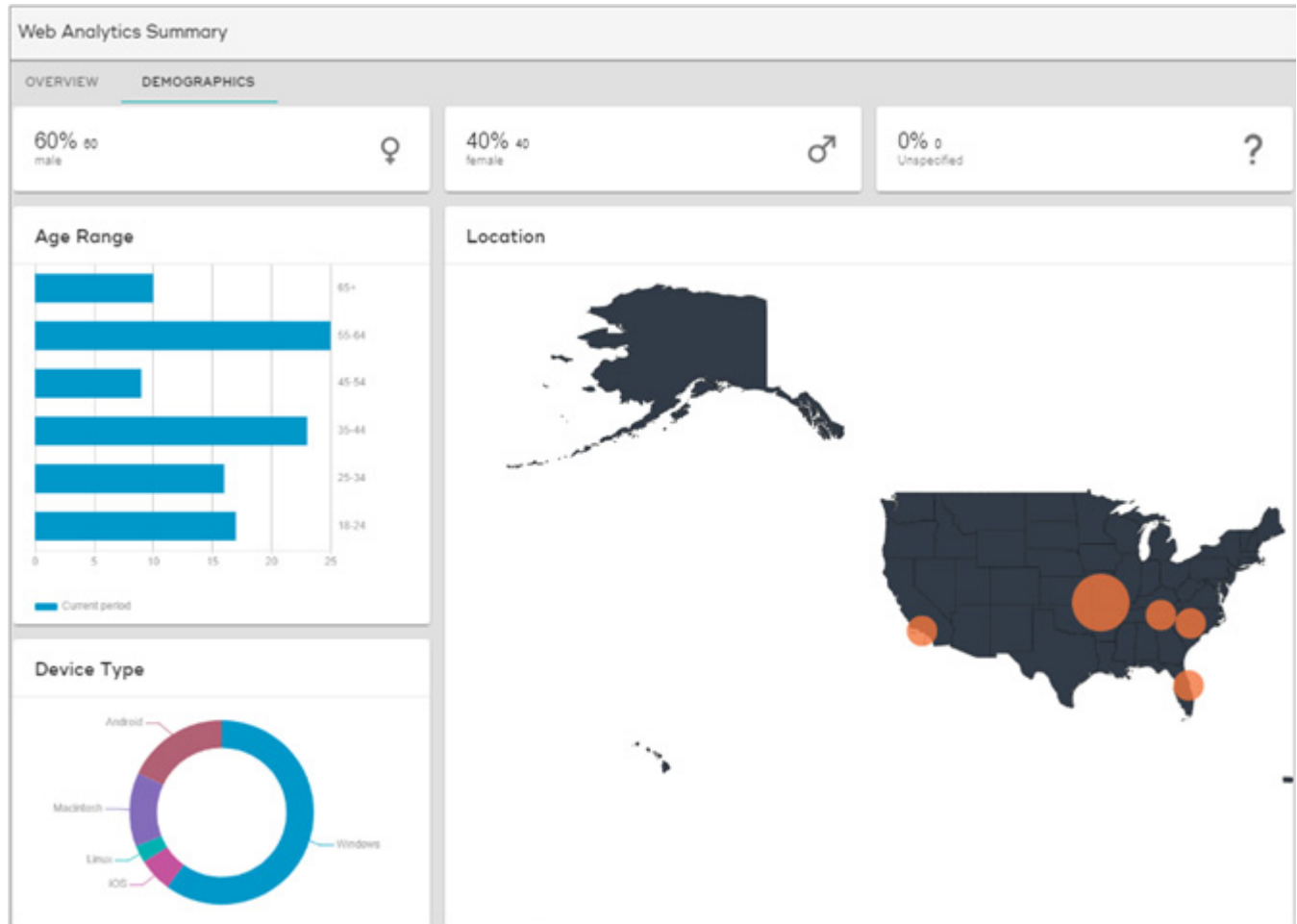


Figure 45 and Figure 46 displays the Web Analytics Details UI. It displays details on most traffic sources, top keywords, device types, Traffic trends and most important actions.

The bar chart in the Customers panel displays the customer segmentation details. The Y-axis shown as A, B, C and D are the segmentation. The segmentation shows high value and low value customers. The horizontal axis shows the source from where the customers are brought in like Facebook, Google, etc. From the chart, if A segmentation shows the most high value customers. The pink color shows that the customers are directed to the client websites directly. And the second high source is displayed as Amazon. It provides information to the business client about the source that fetches high customer ratio and the business can decide to invest their funds wisely based on the insights.

Figure 45: Web Analytics Details

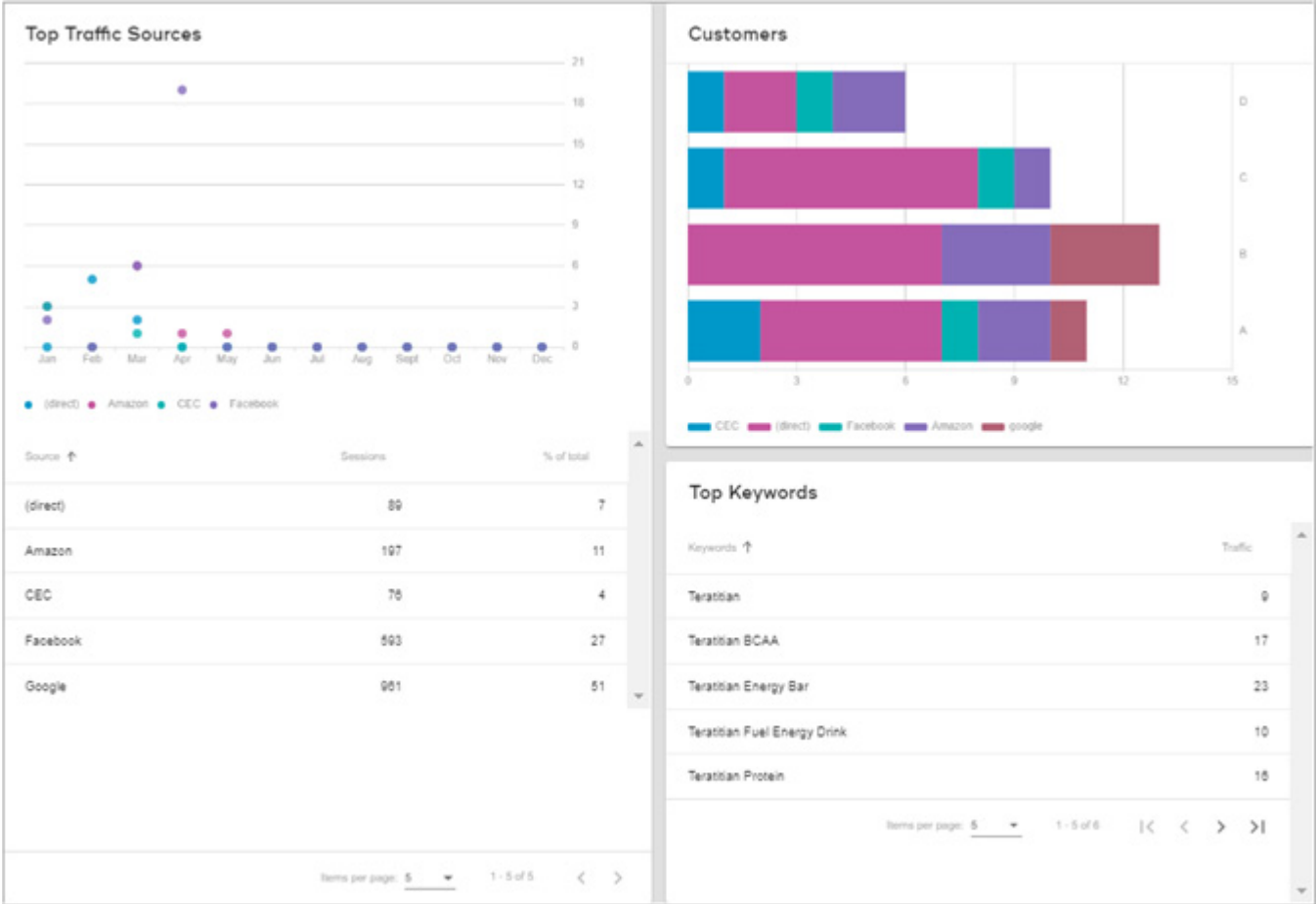
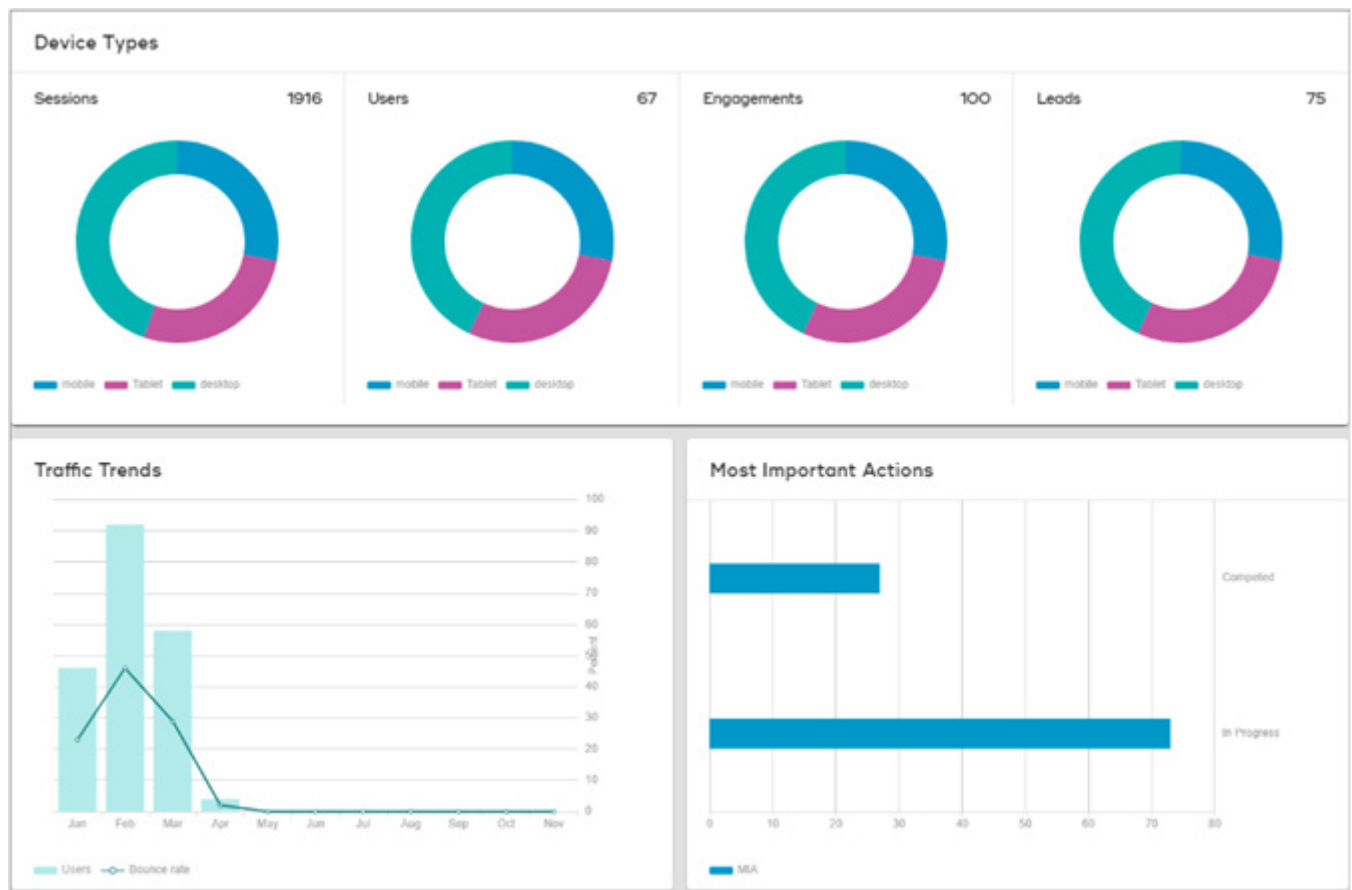


Figure 46: Web Analytics Details



Server Side Configurations

The below section describes the connected Identity server side configurations.

Access Token or Client Secret files

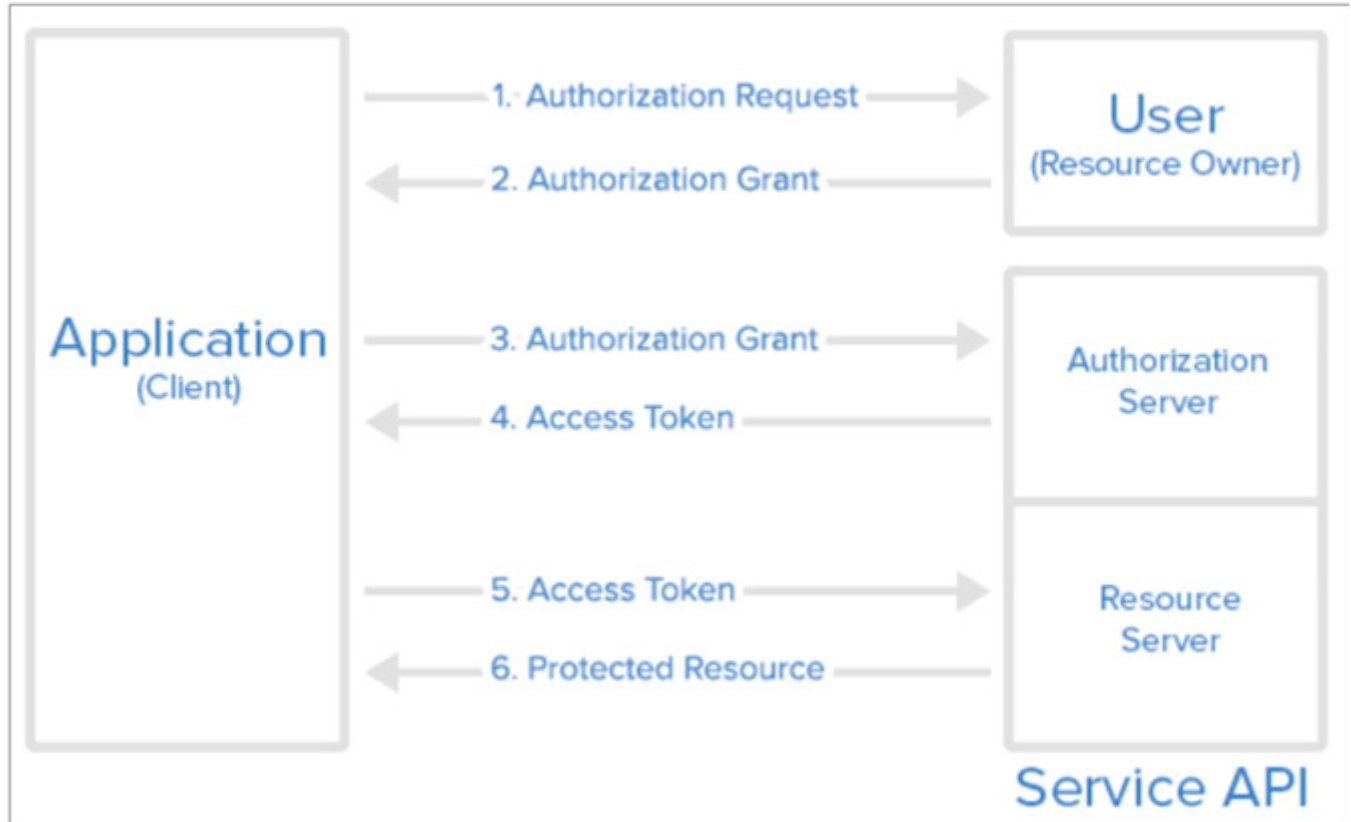
OAuth2.0 is the authentication method used by Google to make sure you are an authorized user for your Google Analytics account. You need a way to tell the Google API client that you are an authorized user to access the selected Google Analytics account. OAuth 2.0 is a standard specification for allowing end users to securely authorize a client application to access protected server-side resources. OAuth 2.0 follows the below flow:

- It provides the access token.
- This access token is then attached with the query request.
- When Server determines that your request and access token are valid, it returns the requested data.
- The ETL process for extracting data from Google Analytics-Python Client API is making use of server-side authentication.

Implementation Details

Figure 47 displays the server side implementations.

Figure 47: Server Side Configurations



Functionality:

- When running the server-side scripts, you need a way to tell the Google API client that you are an authorized user to access the selected Google Analytics account.
- Instead of sending usernames or passwords to Google directly, the script will use the Analytics.dat file for authorization and will not include username or password of the user who is accessing GA data.
- As a first step: A web portal needs to be shown to add username and password for authentication.
- Once authentication is completed, Google will create client_secret.json which needs to be downloaded from the web portal.
- An Analytics.dat will be created with client_secret.json which as Access token, scope - This is a necessary file to support OAUTH 2.0 authentication.
- On every ETL call, the script which calls will need to use Analytics.dat to authenticate itself, once done OAuth exposes endpoints that you can use to authenticate users and get their authorization.

Outcomes: The above steps will authenticate the user and check the user authorization for GA account access.

List of Accounts, Property and Profile Information

The below server-side configuration will be used to create and list down all the Account/Profile Information of the Google Analytics Account configured.

Functionality:

- The User will submit the Account ID, View ID, and Profile Name as a one-step Configuration for Fetching the Data related to a profile from Google Analytics.
- The above-mentioned configuration data will be stored in REF_USR_CONFIG Table.
- In the UI, the user will click on settings button, and click the GA ID (which is a Profile ID), and all the associated detail will be fetched automatically from the REF_USR_CONFIG Table.
- The REF_USR_CONFIG Table contains the Account id, Profile id, Profile name and Web property ID.
- The Server-side Python script will fetch the data from the REF_USR_CONFIG table to query the dimension and metrics at a view/profile level.

Outcomes: The account information will be fetched from the REF_USR_CONFIG Table.

Metrics and Dimensions to Fetch to Database

The Serve Side Metrics or Dimensions would be fetched using an ETL Process which would load the data into the Master Tables as per schedule.

Dimensions are attributes of your data. For example, the dimension City indicates the city, for example, "Paris" or "New York", from which a session originates.

Metrics are quantitative measurements. The metric Sessions is the total number of sessions.

Functionality:

- The Metrics and Dimensions Metadata Values displayed on the Left Side of Configuration Table will be shown using the reference data tables REF_DOMAIN, REF_DIMENSION, REF_METRIC.
- Once the Metrics and Dimensions have configured by the user it will be stored in USER_METRIC_CONFIG Table.
- Once the data configuration step is complete, then ETL process will fetch the data based on the frequency set by the user:
 - The master tables would be populated with actual data.
 - The master tables would be then used for displaying the aggregate counts and summary counts.
 - List of metrics and dimensions to be fetched will be maintained by configuration tables.

Exceptions: "Not every metric can be combined with every dimension. Each dimension and metric has a scope: user-level, session-level, or hit-level. In most cases, it only makes sense to combine dimensions and metrics that share the same scope. For example, Session is a session-based metric, so it can only be used with session-level dimensions like Source or City.

Outcomes: The ETL Integration process would enable creation for customer journey and customer 360.

Database Development

The data integration process for Google Analytics would be done using Google Analytics Python client. The data would be fetched using the configuration parameters provided in previous Server-Side Configuration Page. The data once fetched would be stored in staging tables and linked to the master data using MDM CDI Process.

Functionality: ETL Process

- Once the user has configured the metrics/dimensions for a domain and scheduled the time frequency in the UI Configuration Page
- The above User input includes: - domain, metrics, dimensions, profile/view id, start date (year, month, date) and end date (year, month, date)
- The user configured inputs would be provided to the ETL Processes to extract Dimensions/Metrics from Google Analytics using Python Client and load them on Teradata Staging Tables.
- The python client request is made to GA server using Http request and get the result in a JSON format and then load that JSON format data in our staging Teradata table in a structured format (row-column format), [this process is done via the same python script]
- There will be a start and end date to define the date range for your data extraction process. This is needed for the initial Job run. Every data extraction job has two phases. The first phase is the initial job run and the second phase is the incremental job run.
- By giving the start and end date the initial job run has these date boundaries to work with. It submits these dates to the API and extracts the data. Once the initial load is successfully completed it switches the job run into “incremental” mode. In the incremental mode, the job automatically picks the last run date as the start date and current date as the end date. This way it ensures it does not miss a single day of data collection and keeps appending rows of data to the existing tables.
- Incrementally load the data from GA Staging Tables to Master Tables based on Delta logic.
- Map the Google Analytics Data and golden data in MDM.

Outcomes: The ETL Integration process would enable creation for customer journey and, customer 360.

Index

- A
- ABC model 19
- Account 92
- Ad serving platforms 91
- Admin Web Services 60
- C
- Campaign 38
- Child Data Population 17
- Cleansing 4, 12
- Cleansing and Standardization 4
- Connected Identity 1
 - Cleanup Workflow 10
 - Dashboard 26
 - Installation 8, 10
 - Installation Pre-Requisites 8
 - Package Structure 6
 - Process Flow 2
 - Process Orchestration 23
 - Solution 2
 - UI Login 11
 - UI Web Metrics 92
- Consent 30
- Cross-Reference 39
- Customer 34
- Customer 360 36
- Customer 360 UI 7
- D
- Data Ingestion Dashboard 31
- Data Issues 32
- Data Load 31
- Data Model 7
- Data Volumes by Source Systems 33
- Demographics 30, 37
- Documentation vi
- E
- EDW Source 2
- Enrollment by Channel 28
- Enrollment by Trend 29
- G
- Golden Master 4
- Google Analytics 91
- H
- Household 41
- I
- Installing
 - MDM 8
- L
- Loyalty 37
- M
- Master Profiles 27
- Matching 4, 14
- Matching Profile 4
- MDM
 - About 1
 - Installation 8
 - Packaging Overview 6, 12, 26, 43, 89
- P
- Process Flow 3
- Profile 36
- Purchases 42
- Purpose iii
- S
- Search Web Services 51
- Server Side Configurations 99
- Source 2, 33
- Standardization 4, 12
- Static Data 6
- Survivorship 4, 16
- Survivorship Profile 4
- U
- Update Web Services 80
- V
- View 92
- W
- Web Analytics 89
 - Summary UI 93
- Web Analytics Platform 90
- Web Analytics Solution 7
- Web Service
 - Implementation 44

Request Types 44
Web Service APIs 7
Websites transaction systems 90
Workflows 6

