



Master Data Management




Developer Guide

Release 4.8.0
B035-0000-9703
December 2022

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.

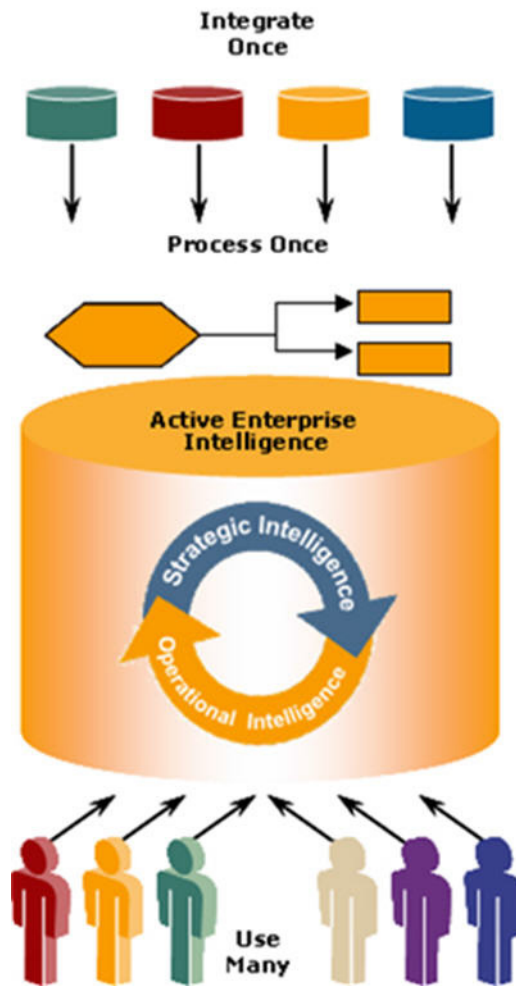
Teradata's MDM provides a framework for building business workflows by defining the necessary data in workflows via X-Docs and expressing the business logic that operates on the data via X-Rules. This framework enables the definition of business process workflows in a configurable manner with little 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 Management 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 is intended to provide the overview and recommended approach and uses of Teradata's MDM capabilities when Designing, Building, and Deploying an MDM Application. The Developers Guide is not intended to replace or repeat the detail information located in the various User Guides, but rather reference them, as appropriate, from the overall life-cycle and development of a MDM application.

Target Audience

This guide is intended primarily for those individuals who are planning, participating, and responsible for Designing, or Building, or Deploying a MDM application. The Teradata's MDM product is a Teradata optimized product and knowledge of Teradata capabilities and architecture are embraced within this developers guide.

What You Should Know

This guide assumes that the audience is familiar with basic development practices of a software application. In addition, the audience should have some familiarity with Services oriented architectures and Web Services. The Teradata's MDM product provides the capability to develop and deploy a business process as a workflow, therefore, in the design of the MDM application any experience in business process analysis and workflow analysis would be beneficial. The data model used by the Teradata's MDM product is aligned to the utilization of Data Models in the Teradata EDW/ADW context, therefore, any experience with Teradata data model implementations would be beneficial.

Document Structure

This book contains the following sections:

- [“Section A—Developer Reference”](#) describes MDM Architecture and provides guidelines for development in MDM.
- [“Section B—Sample Application”](#) describes MDM solution and its key features. It provides information on creating a sample application and sample application (CDI) process. Explains data modeling environment provided by MDM Studio and defines Web components.

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
August 2013, 3.3	Updated sample application installer images.

Date and Release	Description
April 2014, 3.3.1	Updated sample application installer images.
November 2014, 3.4	Updated sample application installer images.
January 2015, 3.4	Version details updated.
February 2015, 3.4.1	Version details updated.
November 2015, 3.5	Version details updated.
March 2016, 3.5.1	Version details updated.
June 2016, 3.5.2	Version details updated.
November 2016, 3.5.3	Version details updated.
August 2017, 4.0.0	Version details updated.
September 2017, 4.0.1	Version details updated.
December 2017, 4.1.0	Version details updated.
June 2018, 4.2.0	Version details updated.
April 2019, 4.3.0	Version details updated.
November 2019, 4.4.0	Version details updated.
June 2020, 4.5.0	Version details updated.
March 2021, 4.6.0	Version details updated.
March 2022, 4.7.0	Version details updated.
November 2022, 4.8.0	Version details updated.

Related Documentation

For more information on MDM, refer the following documents:

- *Master Data Management Studio User Guide*
(Master Data Management 4.8.0 Studio User Guide.pdf)
- *Master Data Management Release Definition*
(Master Data Management 4.8.0 Release Definition.pdf)
- *Master Data Management Server Guide*

- (Master Data Management 4.8.0 Server Guide.pdf)
- *Master Data Management Installation Guide*
(Master Data Management 4.8.0 Installation Guide.pdf)
- *Master Data Management Reference Guide*
(Master Data Management 4.8.0 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	v
About Teradata's Master Data Management	vi
About This Book	vii
Target Audience	vii
What You Should Know	vii
Document Structure	vii
Changes to This Book	vii
Related Documentation	viii
To Read The Documentation	ix
Customer Support	ix
Documentation Feedback	ix

Section A: —Developer Reference

Chapter 1: Teradata MDM Overview

Product Overview	2
Product Composition	4
Business Architecture	5
General Process	7
Technical Architecture	8
MDM Studio Architecture	9
MDM Platform Architecture	11
MDM Database Topology	14
Data Architecture	17

Chapter 2: MDM Development Guidelines

Overview	18
Model Development	18
Model Naming Conventions	18
Model Customization Guidelines	19

Business Logic Customizations	20
Workflows	20
Rules	20
Validations	20
DataPersist Rules	21
Additional Best Practices on Business Logic Customizations	21
User Interface Customizations	21
X2 based	21
PGL based	22
Best Practices - PGL based UI Workflow Development	22
Reuse of Code Modules	24
Creating Reusable Modules	25

Chapter 3: Development of an MDM Application 38

Overview	38
Scoping an MDM Application	38
Planning an MDM Application	39
Skill Set Requirements	39
Building an Application in MDM Studio	40
Overview	40
MDM Development Elements	44
X-Documents	44
X-Rules	45
X-Operations	45
X-Path	46
Key Features of Studio	46
Hierarchy Management	46
Web Services Support	47
OMI	49
Data Authorization	49
Data Quality Monitors	49
MDM Test Services Framework	50
Deployment Support	50
Sample Application Support	51
List of Key Services	52

Chapter 4: Customer Service 54

World Class Support	54
---------------------------	----

Web Access	55
------------------	----

Chapter 5: Training..... 56

Training Information	56
----------------------------	----

Section B: —Sample Application 57

Chapter 6: Introduction 58

Introduction	58
MDM Key Features	59
Studio Sample Projects.....	60
MDM Sample Application.....	60
Custom Application.....	60
MDM—Sample Application (CDI) Solution.....	61
MDM—Sample Application (CDI) Solution with Trillium Software.....	61

Chapter 7: Custom Application 62

Introduction	62
Custom Application Creation.....	62
Sample Application Setup	74
Installation	74
Load Pre-defined Data.....	84
..... Configuring Sample Application Project in Eclipse	84
Enable Web Services (Optional Step)	84
Custom Application Folder Structure.....	87

Chapter 8: Custom Models 89

Creating a Model using Studio.....	89
Importing a Model into MDM Studio	89
Import from Relational Database.....	90
Import from ERwin	91

Import from XDocs	93
<hr/>	
Chapter 9: Define Web Component	94
Define UI Navigation Structure	94
Include into Web Component	96
<hr/>	
Chapter 10: Sample Application Process	98
Introduction	98
Sample Application Data Model	98
Sample Application CDI Process Flow	99
CDI Process Flow	100
Sample Application with Trillium Process Flow	113
Summary	113
Trillium Process Flow	113
<hr/>	
Appendix A: Publication Services	119
Introduction	119
Publication Object	121
Publication Method	121
Publication Node	121
Logical Data Model	122
Metadata Tables	123
Following an Audit Trail of a Publication Request	127
<hr/>	
Appendix B: Configuration of Trillium Client	130
Trillium Client Setup in MDM	130
<hr/>	
Appendix C: MDM Custom Web Services	136
Introduction	136
Incoming Teradata MDM Web Service	136

- Installation and Setup Instructions for MDM Web Service. 137
- To Enable Custom Web Service in SampleApplication 137
- Sample Web Service Function in MDM 137
- Core Service Workflow 145
- Outgoing Third Party Web Service 150
 - Third Party Web service - Request Format 150
 - Third Party Web Service - Response 151

Index 154

List of Figures

Figure 1: MDM Technical Architecture	3
Figure 2: Solid Framework and Solution Components	4
Figure 3: MDM Process Flow	6
Figure 4: Connected Identity (CI) Uses for Various Workflows	7
Figure 5: Sample Development Flow	8
Figure 6: Development and Deployed Architecture	9
Figure 7: Studio Architecture (High Level View)	10
Figure 8: MDM Platform Architecture	11
Figure 9: Development and Deployment relationship: MDM/RDM Server	12
Figure 10: Logical Deployment	12
Figure 11: Application Server MDM Platform environment	14
Figure 12: MDM Database Topology	15
Figure 13: Logical Database Table	16
Figure 14: Holistic Data View	17
Figure 15: Master Data examples	17
Figure 16: Service Setup-Setup Nature	26
Figure 17: Service Setup-Setup Options	27
Figure 18: Service Added	28
Figure 19: Service Configuration—Archived Service	29
Figure 20: Setup Options	30
Figure 21: Run Configurations	31
Figure 22: Classpath	32
Figure 23: Service Setup-Setup Nature	33
Figure 24: Service Setup-Setup Model Instance	34
Figure 25: Service Setup-Setup Options	35
Figure 26: Service Added	36
Figure 27: Service Configuration—Existing Service	37
Figure 28: Development Flow	41
Figure 29: Data Quality related functionality	42
Figure 30: Data Load Template Workflow	43
Figure 31: Web Services Support—Incoming Web Service	48
Figure 32: Web Services Support—Outgoing Web Service	49

Figure 33: Sample Application Creation process	52
Figure 34: Custom Application—Studio Project Files	63
Figure 35: Custom Application—Insert Model	64
Figure 36: Custom Application—Models and Dictionaries	65
Figure 37: Service Setup—Setup Nature	66
Figure 38: Service Setup-Setup Model Instance	67
Figure 39: Custom Application Service	68
Figure 40: Enter Database Name in Schema Config	69
Figure 41: Folder Structure.	70
Figure 42: x2.properties file	72
Figure 43: Solution Setup	73
Figure 44: Introduction	75
Figure 45: Choose MDM Installation Folder	76
Figure 46: Deploy Custom Application/Change Password	77
Figure 47: Database Settings	78
Figure 48: Incorrect DB Information	79
Figure 49: Choose Custom Application jar to Deploy	79
Figure 50: Enter Project ID & Your Name	80
Figure 51: Database Preparation	81
Figure 52: Default Mode of Deployment	82
Figure 53: Please Choose Mode of Deployment	83
Figure 54: Install Complete	84
Figure 55: New Group	85
Figure 56: WSDL Input	85
Figure 57: WSDL Input Dialog Box	86
Figure 58: WSDL Input	86
Figure 59: Import from Relational Database	90
Figure 60: Import from ERwin	91
Figure 61: Import from XDocs	93
Figure 62: Define UI Navigation Structure	95
Figure 63: RLDM Model and Customer Model relationship	99
Figure 64: Sample Application process flow	100
Figure 65: Overall CDI Process Flow	100
Figure 66: Load Data	101
Figure 67: Rule Type: SQL_Filter	101
Figure 68: Rule Type: SQL_Validation	102

Figure 69: Rule Type: SQL_Validation with Severity WARNING.	102
Figure 70: Rule Type: SQL_Validation with Severity Error	103
Figure 71: Rule Type: SQL_Insert.	104
Figure 72: LEGACY_CUSTOMER.	106
Figure 73: Customer Dashboard.	107
Figure 74: Customer Dashboard—Summary.	108
Figure 75: MDM Survivorship UI	108
Figure 76: Interactive Merge Workflow UI	109
Figure 77: Manage Customer workflow UI	110
Figure 78: New Customer Introduction UI.	111
Figure 79: Define Service File	112
Figure 80: Workflow to Publish Data.	112
Figure 81: CDI Process Flow (Trillium)	113
Figure 82: Trillium CDI Process Flow.	114
Figure 83: Publication Services	120
Figure 84: Publication Services Databases.	122
Figure 85: Database Tables	123
Figure 86: Publication Object Audit Tables.	124
Figure 87: Publication Request Tables.	125
Figure 88: Publication Request Audit Tables.	126
Figure 89: WSDL Transform Input	139
Figure 90: WSDL Transform Output	141
Figure 91: MDM_Login_Page	145
Figure 92: Welcome	146
Figure 93: Add Details	146
Figure 94: Get Details.	147
Figure 95: Details	147
Figure 96: Modify Details	148
Figure 97: Mass Update	149
Figure 98: Mass Update	149
Figure 99: WSDL Transform Input	151
Figure 100: WSDL Transform Output	153

List of Tables

Table 1: MDM Supported Teradata Data type Mappings 16

SECTION A —Developer Reference

Section A provides link to the various chapters on Developer Reference.

- [Chapter 1: “Teradata MDM Overview.”](#)
- [Chapter 2: “MDM Development Guidelines.”](#)
- [Chapter 3: “Development of an MDM Application.”](#)
- [Chapter 4: “Customer Service.”](#)
- [Chapter 5: “Training.”](#)
- [Appendix A: “Publication Services”](#)

CHAPTER 1 Teradata MDM Overview

What's In This Chapter

This chapter provides information about MDM Architecture and core framework.

Topics include:

- [Product Overview](#)
- [Product Composition](#)
- [Business Architecture](#)
- [General Process](#)
- [Technical Architecture](#)

Product Overview

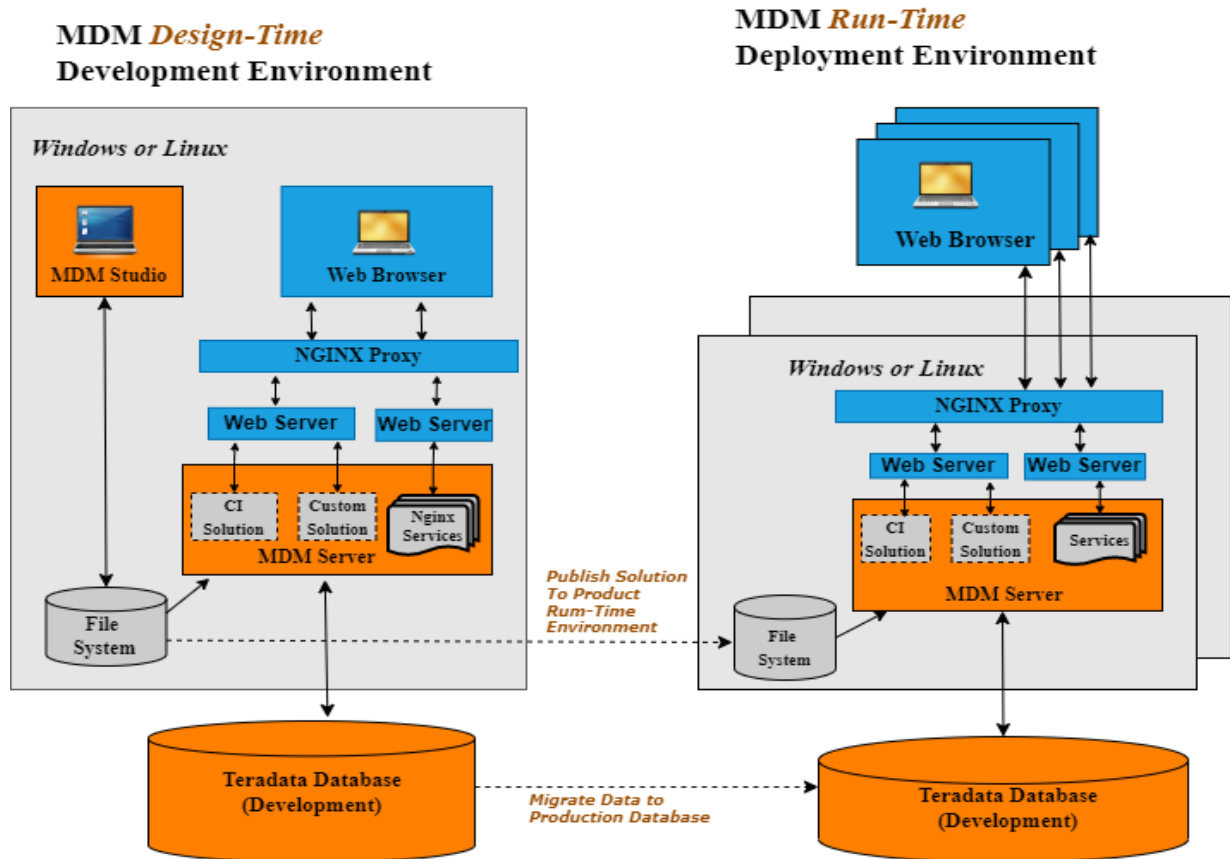
Master Data Management refers to the methods by which clean, accurate, and consistent master data are managed, referenced, and synchronized across the enterprise and made available to users, as required. It may leverage policies and procedures for access, update, and overall management of this central resource and its coordination with other participating systems across the enterprise. The Teradata Master Data Management Product consists of software assets that provide this functionality. In addition, the Teradata Professional Services organization has a specific Master Data Management Solution Implementation Methodology.

Teradata MDM is built upon open standards (such as xml and Web Services) and provides the capability to leverage existing IT and Business investments. The key to the MDM Platform is its ability to enable flexible business process management by employing a Services Based Architecture (SBA). It provides a powerful business programming environment to describe business rules, workflows, and data models.

The MDM Product is composed of software, documentation, a reference application and solution accelerators. The Teradata MDM product consists of a set of framework software components that are installed in a development platform and in a run-time environment.

The Teradata MDM run-time product software is casually declared to be a 3-tier application. The three tiers consist of the Teradata Database, an application server, and a client workstation. The Teradata MDM product requires the Teradata database.

Figure 1: MDM Technical Architecture



There are two major components to Teradata MDM, Teradata Studio (MDM Design-Time Development Environment), and MDM Server (MDM Run-Time Deployment Environment).

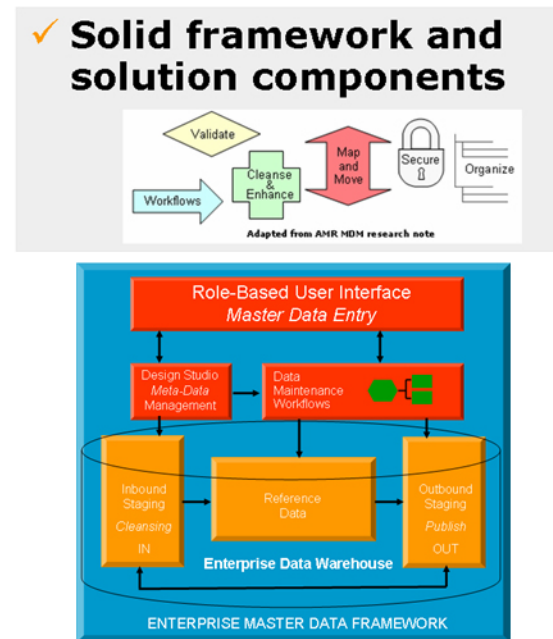
The Teradata MDM Studio is used to develop MDM applications. The MDM Studio provides the developers with an environment that can define/ingest Data Models, build business process workflows, create User-interfaces, import Web Services (which in turn can be used to define touch points with external applications and used seamlessly within workflows).

Business process workflows are assembled in Studio to reflect any process flow. Standard workflows are provided with the product for basic functions such as Data Upload. The supplied Reference Application provides other examples of workflows which demonstrate the various support functions. A workflow is primarily a business process that is required for the Master Data. It may include a User-interface, as well as, any other function. MDM workflows typically consist of both system-to-human interactions and system-to-system interactions.

User-Interfaces are contained in a workflow. All of the processing logic for the UI is contained within this workflow, as well. The User-Interface itself is built within Studio starting with defining a Page Layout (PGL) that is then customized using various PGL constructs. The input and output Data Specification is defined in xml format and is correlated to the layout at run time. X-Rules are used to process the data.

Figure 2: Solid Framework and Solution Components

- Flexible process-driven framework & workflow to tailor master data governance to your business's unique processes
- Open Service Architecture supports 3rd party tools & content
- Integration with Teradata's logical data models
- Hierarchy management → organize & reorganize master data as needed
- MDM Design Studio -- toolkit adapts reusable/common MDM services to each customer's unique needs



The Data Models, Business workflows, and user-interfaces designed in MDM Studio are deployed into the appropriate environment using the MDM Server. The MDM Server is a set of Java services run on a Java Virtual Machine. It consists of a set of services (Locator, MDM Server) that powers the data and process models built in MDM Studio. The MDM Deployment Manager actually provides the support for moving a Studio built MDM application into the appropriate target environment (Test, Production, etc.). The deployed target environment must have Teradata and an application server. Refer to *MDM Platform Release Definition*.

Product Composition

The Teradata Master Data Management Product is composed of Software, Documentation, and a Reference or Sample Application. The Teradata MDM solution additionally contains the Professional Services value add methodology, Best Practices, and Service resources. Please refer to the following for inquiries into the MDM CoE.

Teradata Product Package contains 2 installations CD-ROMs.

- Teradata MDM Platform – Teradata Software
- Teradata MDM Platform – 3rd Party Software

The “3rd Party Software” Installation CD contains various Java 3rd Party and Open Source Libraries (jar files) that must be extracted from the appropriate archive files (zip or

tar.gz) *before* running the installation packages from the “Teradata Software” Installation CD.

The Teradata Software consists of all of the user documentation in both PDF and HTML formats, and all of the MDM Platform installable software packages. In the “\Install” directory the installation includes MDM Studio, a Java, Swing-based, Windows only Desktop/Workstation application. In addition, the installation package also includes the Java, J2EE-based MDM Client and MDM Server, including all MDM platform services and all of the necessary ‘run-time’ components that can be installed on the same workstation as MDM Studio, or on a separate platform. Please note that there is a Windows and a UNIX (AIX or Linux) installation package.

Refer to *MDM Platform Release Definition* for a detail description of each item.

Teradata MDM provides a ‘Solution Accelerator’ or Reference Application along with the product. This sample application, “SampleCDIApplication” is included with the product CD and is installed per instructions supplied. The purpose of the “Reference Application” is defined as follows:

- Provide an example of the use of Key MDM 2.0 features
- Provide a consistent example application that is used for training, user guide examples, and potentially a solution accelerator for customers interested in building a Customer Data Integration application with our MDM product.
- Provide an example application that contains the Trillium API’s for Data Cleansing and Matching.

The Reference or Sample Application is a defined CDI (Customer Data Integration) Application. CDI is a comprehensive set of technology components, services, and business processes that create, maintain, and make available an accurate, timely, integrated, and complete view of a customer across lines of business, channels, and business partners. Our sample CDI application can optionally utilize the Trillium Data Quality Server to perform such functions as Cleansing, Matching, De-Duplication, etc. This option utilizes an API to interface into the Trillium DQ Services. This Java API is contained with the shipped product, but ordering, installing and configuring the Trillium Data Quality Services are not included.

Business Architecture

The Business architecture is most important from the ‘Deployment’ perspective, but the Development portion of the MDM Application must be successful, in order to attain a successful Deployment. In this section, we shall describe the basic recommendations for Development and Deployment, from the Business perspective.

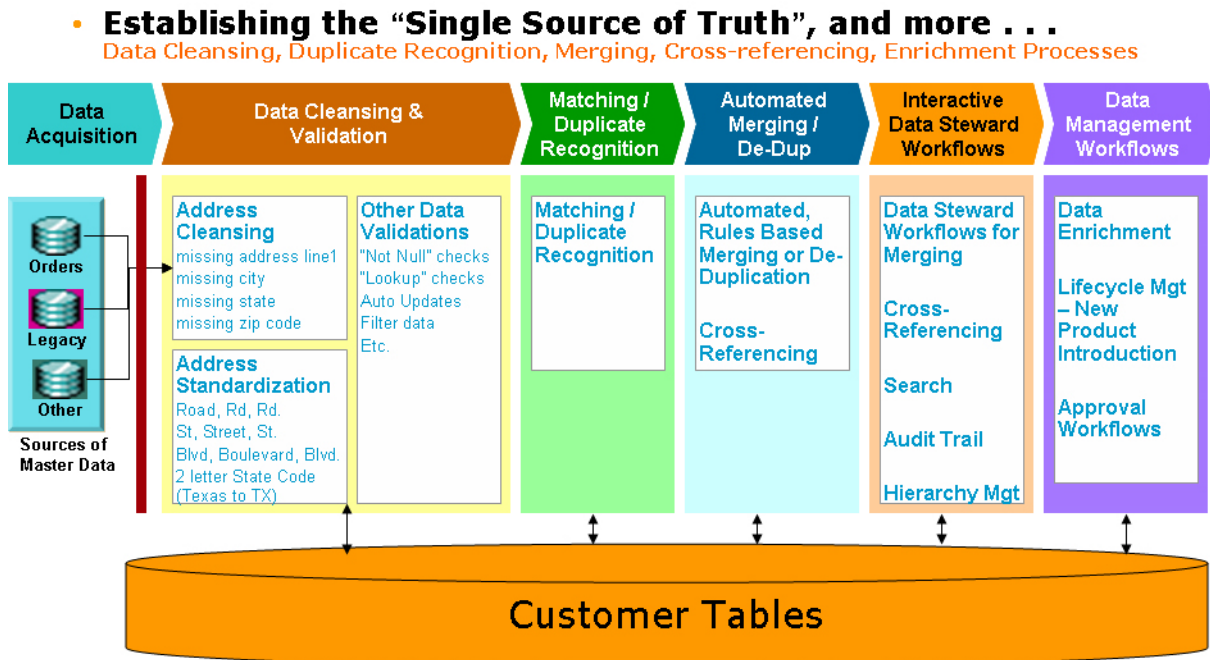
We have utilized our Reference Application to provide a description of the architecture of our MDM product from a Business Perspective.

The successful development of a MDM application is not unlike the development of any other software application. There are basic development practices that should be followed when developing an MDM Application, as well as any other application. With MDM, a Data

Centric approach, coupled with Business Process analysis supported by Business users, and with appropriate management support are all a must. Based upon experience our analysis, design, and development teams have found that an iterative approach (with each iteration between 16 and 20 weeks) has been most successful and is recommended (the 16 to 20 weeks iteration cycle time will act as natural scoping). The common practice of Agile development is also encouraged, as the Teradata Studio (MDM Development environment) lends itself to agility, by definition. The Teradata Studio was designed to support a Data Modeling approach with Data related workflows and then Rapid prototyping of the solution workflows and User Interfaces.

This Business Architecture Diagram represents our Reference Application, CDI. It defines the basic Business flow from Data Acquisition through Data Management workflows. In the following section we relate these business functions to the appropriate feature of our MDM Product.

Figure 3: MDM Process Flow



As with all development efforts proper planning and appropriate resources are key success factors. Business sponsorship and approval are absolutely necessary, as it is the business processes that will govern the Master Data.

Our Professional Services Methodologies have captured years of experience in successful development efforts of MDM Applications and incorporated that experience into the Teradata MDM Implementation Methodology. The MDM CoE has resources that are experienced and successful in these development efforts. It is recommended to structure the solution requirements into the following major work-streams:

- Source System Analysis
- Subscribing System Analysis

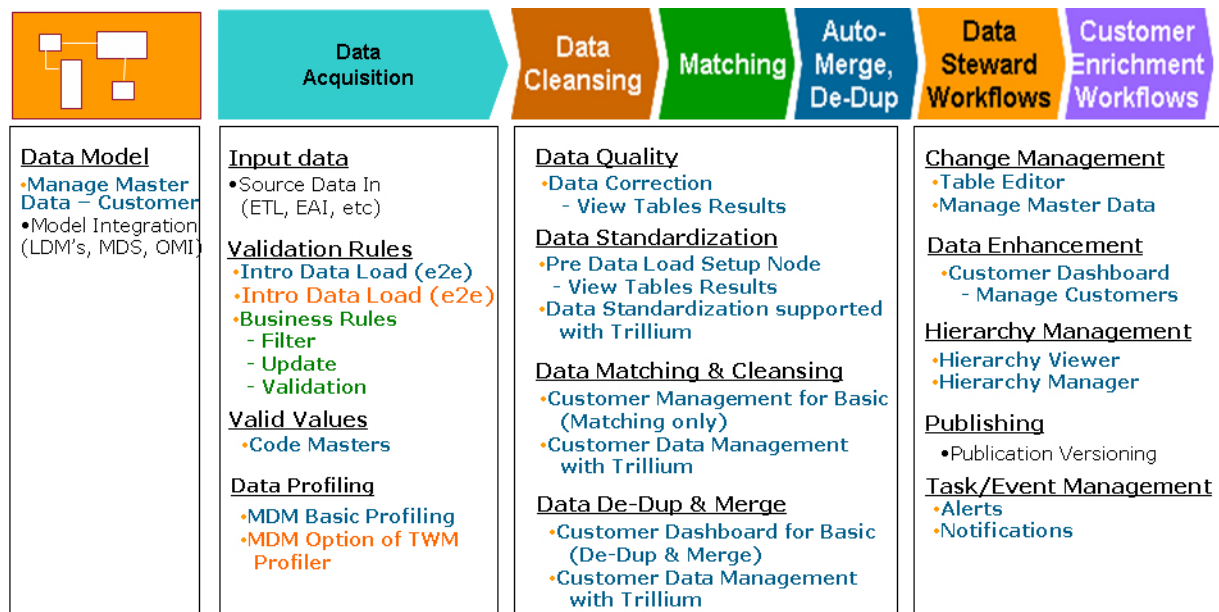
- Data Model Analysis (Logical and Physical Modeling)
- Data Integration (Web Services integration, EAI, etc)
- Data Validation, Business Rules and Data Quality requirements
- Data Management Workflow requirements

Refer to *MDM Platform Studio User Guide* for detailed information regarding the Studio architecture, features, and functions.

General Process

The following diagram relates the basic functions of the Reference MDM application to the high-level features of the MDM solution. All of the features represented below are designed and built within the Teradata Studio environment. The Data Model is defined within Studio and then the validation and Data Quality rules are created. The specific Business workflows to perform functions such as Change Management are created for Data Stewards, and the User Interfaces are then created. Once all of the ‘System’ is approved the application is deployed into the appropriate environment. The diagram below represents the MDM Application of CDI. Please note that with in this diagram we represent the ‘optional’ Trillium usage for various functions.

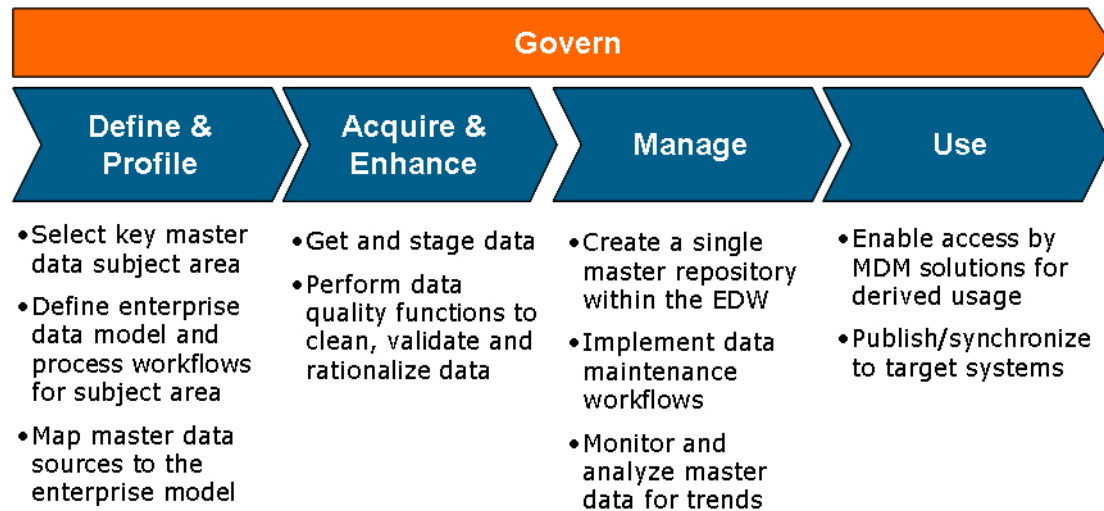
Figure 4: Connected Identity (CI) Uses for Various Workflows



From a generic process perspective the following diagram represents a sample development flow.

Figure 5: Sample Development Flow

More than a system, an overall data governance process.



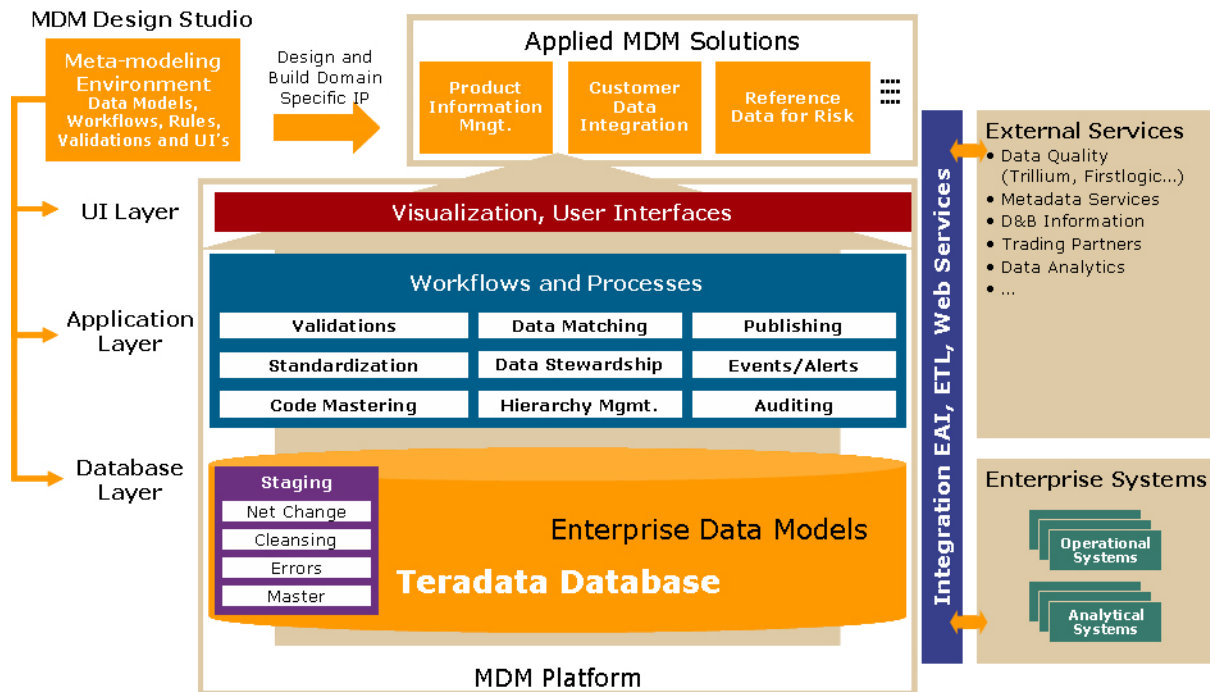
Technical Architecture

In this section we provide Technical Architecture concepts and definitions. We shall begin with a high-level technical architecture of the overall MDM Product, including the development environment (MDM Studio) and the deployed application environment (MDM Platform). This section is organized as follows:

- Overall Technical Architecture
- MDM Studio Architecture
- MDM Platform Architecture
- Database Topology

The following diagram attempts to place both the development and the deployed architecture into a single graphical representation. Our MDM detailed user and administration guides will provide more detailed descriptions.

Figure 6: Development and Deployed Architecture



MDM Studio is the development environment that is used to develop MDM Applications. The Studio is a Java Swing Application that currently is installed on a Windows XP Pro or 2003 workstation. The Teradata database is required (Teradata 6.1, 6.2, or 12.0). With Teradata 12.0 the Teradata database can be installed on the workstation if chosen to do so, that is, the Demo version can be used to develop the MDM Application.

The heart of the MDM Studio resides with the utilization of XML (eXtended Markup Language). In addition, extensive use of a services based architecture is embraced. MDM Studio should not be mistaken for or compared to a full Integrated Development Environment (IDE), as it was produced to only create MDM applications. The Studio User Guide contains a complete description of the various features of the development environment.

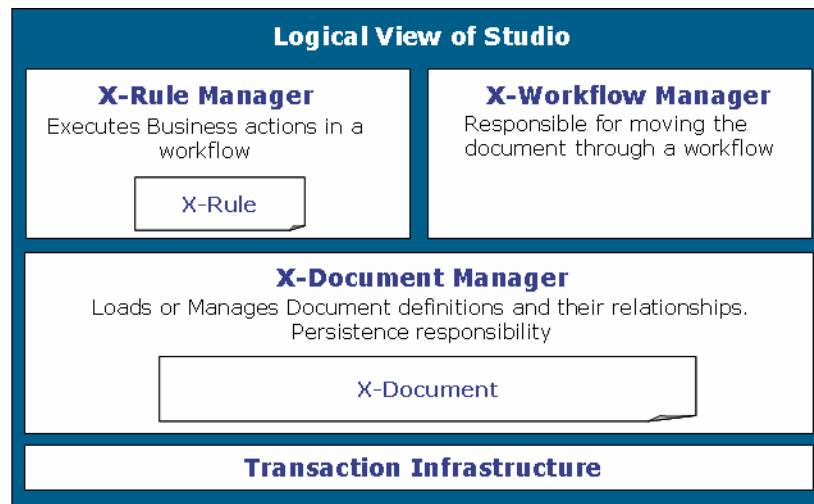
As described in the diagram above, Studio has the capability to provide custom User Interfaces, business workflows (Application layer) and the Data Model. Business Validations rules can be defined, the workflows themselves, business logic, Web Services, etc, are all supported and/or built with Studio. Once the MDM application (Applied MDM Solution) is built the Deployment Manager can provide the support to migrate the appropriate assets, in the appropriate form, to a targeted environment. The supported deployment environments include Web Sphere or Web Logic, on AIX, Windows, Linux Operating systems. Internet Explorer 6.0 is required for the MDM Application Client.

MDM Studio Architecture

The MDM Studio is the development environment for building MDM applications. MDM Studio was initially built from a framework known as X-Core, hence many names are prefixed with 'X-'. The intent of this framework was to build data centric workflows. The X-

Core framework intensely embraced the use of xml. You will note that in the detailed documentation much of the terminology reflects the intense use of xml. The diagram below depicts the high-level logical view of the Studio architecture:

Figure 7: Studio Architecture (High Level View)



At the core to the Studio architecture is the Business Document, referred to as the X-Document. A Business document or X-Document instance is analogous to a HTML page. It has properties (data) and relationships (navigable links) to other Business documents (or X-Documents). The valid properties and relationships that an X-Document instance can have is specified through an XML Specification, referred to as the X-Document Definition. The X-Document Definition is analogous to a DTD (Document Type Definition) for that specific X-Document instance. A DTD defines the legal building blocks of a xml document, including the structure with a list of legal elements and attributes. The X-Document Manager manages the X-Documents. Management of X-Documents includes:

- Creating, Modifying, querying and removing X-Document instances
- Navigating X-Document instances through specified relationships
- Adding a X-Document instance and its related instances in a single nested xml request
- Serving a X-Document instance and its related instances as a nested xml structure
- Finding X-Document instances through an xml based query

Business rules and actions in a workflow are supported through an xml based scripting language known as X-Rules. X-Rules support the expression business logic on the X-Document notifications and generic methods. A X-Rule has a condition and a set of actions. The X-Rules Manager loads the X-Rules at start-up time and converts them into internal data structures to ensure optimal execution. One or more rules can be associated to a workflow step. The Rules interpreter will send all of the associated X-Rules whenever the workflow step is activated. X-Rules Manager provide hooks to call Java code to perform complex actions from within a X-Rule execution context.

The Transaction Manager is responsible for starting and ending transactions. Transaction Manager creates a concrete implementation of Transaction Context interface. Transaction

context (i.e. transaction resources such as JDBC connections) is accessible by all X-Operation components participating in the transaction.

The X-Workflow Manager is responsible for managing the movement of documents through a workflow. The Workflow Manager matches the event nodes of the workflow instance to the corresponding request or event by matching the request or event name and keys with the event descriptor and matching keys of the event node respectively.

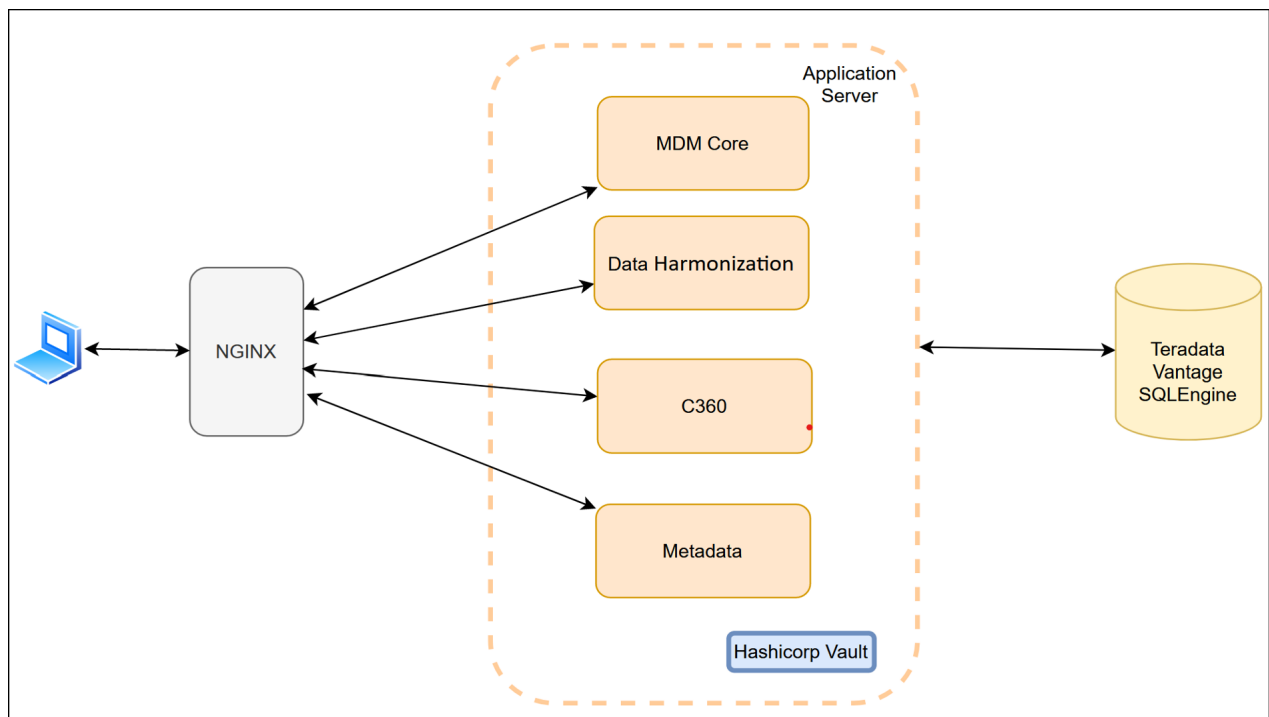
MDM Platform Architecture

The MDM Platform is defined as the target environment that a MDM Application is deployed into. The physical environment is casually referred to as a 3-tier architecture. Once the MDM Development is deemed acceptable or complete, the MDM Deployment Manager can be used to migrate the software assets into the targeted environment.

The MDM is installed with Nginx proxy server which is the third party component. Nginx proxy configuration enrout all the URL prefixed to the individual services like Data Harmonization, Customer 360, Metadata in application server.

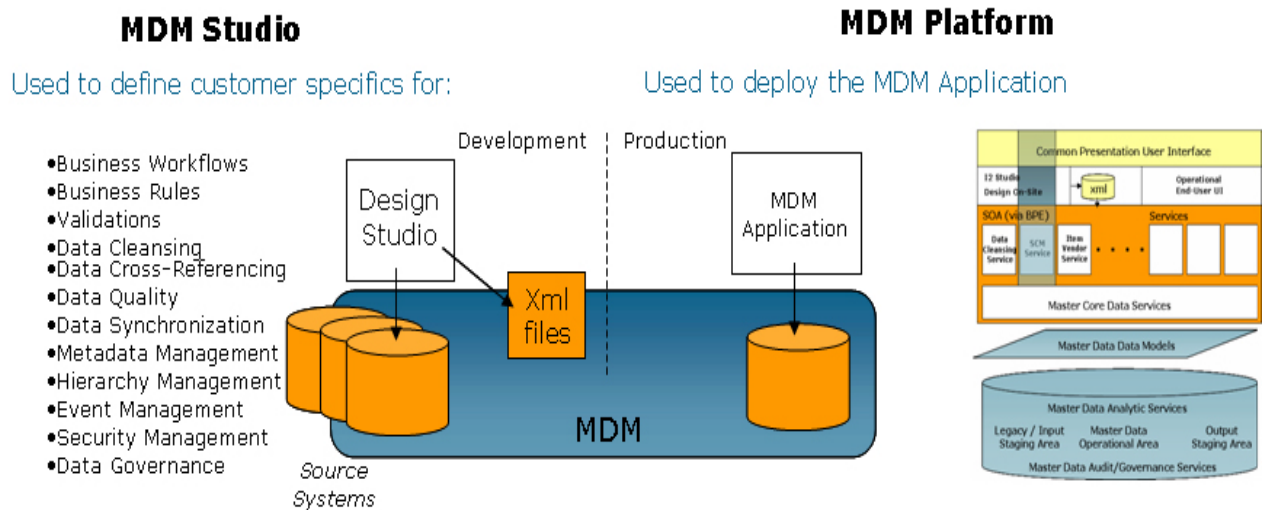
Hashicorp vault can also be installed in MDM Server. The vault stores password and other secret information which Data Harmonization, Customer 360, and Metadata services used.

Figure 8: MDM Platform Architecture



Below diagram below depicts the general high-level description of the relationship between the development and deployment environments:

Figure 9: Development and Deployment relationship: MDM/RDM Server

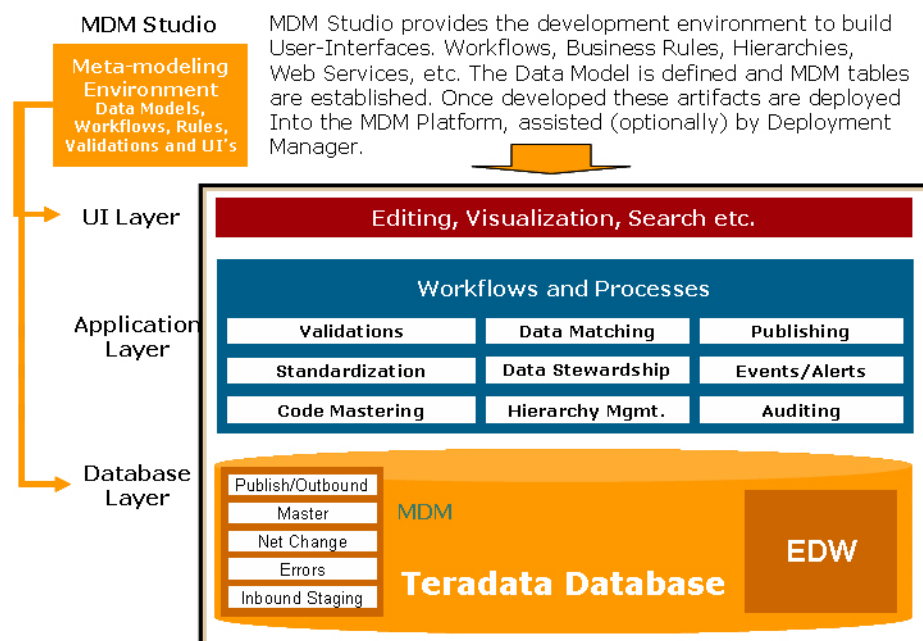


The MDM Platform is responsible for the MDM Web Application. The MDM Server runs on the Application Server in conjunction with either Web Sphere or Web Logic. The Teradata Studio development environment produces all of the various xml files that are required for an MDM Application. The MDM server is the Java portion that will read and process all of the xml files. The Deployment Manager will assist the user in migrating from Studio to any target MDM Platform environment.

From a logical perspective, Studio provides the development environment and produces a project and its associated services that becomes a MDM Web Application.

The diagram below represents what gets logically deployed into the MDM Platform.

Figure 10: Logical Deployment



The client of a MDM Application requires Internet Explorer 6.0 on the users workstation. This workstation will access the MDM Web application on a Application Server that supports Java based Web Services. The Application Server software can be either Web Sphere or Web Logic. Refer to *MDM Platform Release Definition* document for specific versions and recommended configurations. The deployed MDM application runs in the MDM Server set of services. The database schema is created as part of the deployment process and supported by Deployment Manager.

The MDM Server code is deployed on the application server and the MDM “.war” files are deployed into the Application server.

The Server Physical System should be configured per our recommendations. Our Teradata MDM Product strategy is to fit within the configuration models that are defined by the respective Application servers. The MDM Server does not heavily depend upon functionality provided by the application servers, at this release time.

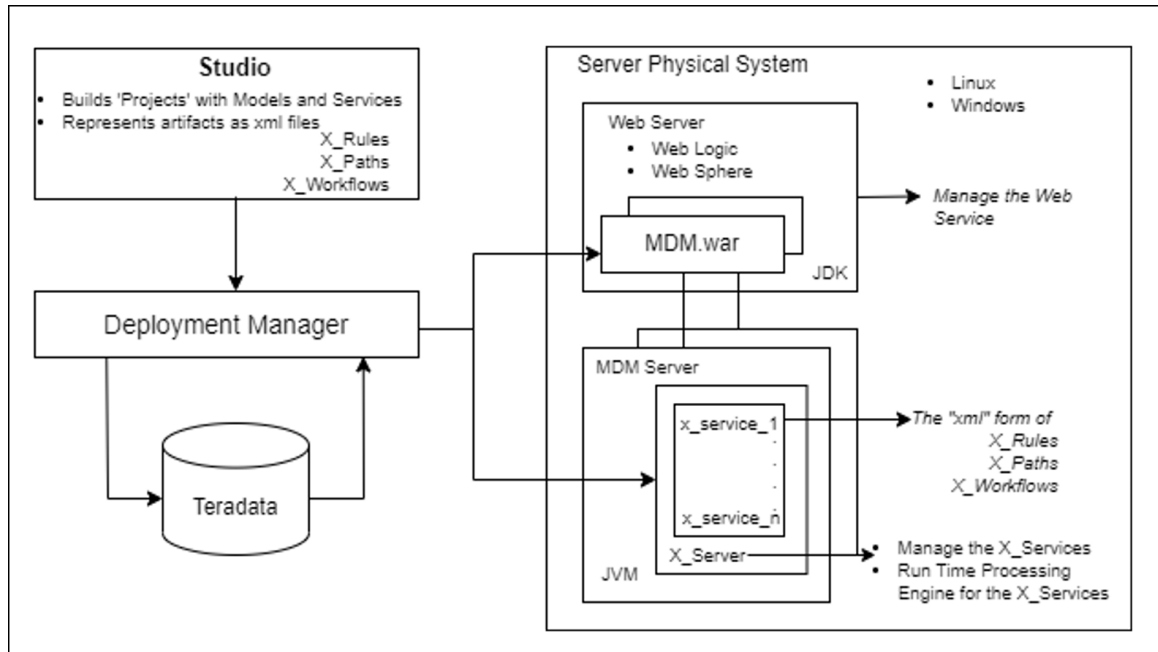
We deploy a MDM .war file into each respective Web Server for each MDM Server that is desired to run on the specific server. You may have multiple MDM servers executing, although they will require a distinct port number. For each MDM Server a distinct MDM .war file is placed, appropriately into the Web Server.

The MDM Server can be considered a X-Server, but additionally, containing all of the pre-built MDM elements (i.e. standard workflows) for a MDM application. It is executing in a single Java Virtual Machine (JVM). The MDM Server is the Runtime Processing engine for the X_Servers.

A X_Server is an instance of a MDM application, again, running in a single JVM. It is the run-time deployment of the application developed using MDM Studio. The X_Server manages the various X_Services that were produced from Studio for the specific MDM application. The X_Server is considered a ‘stateless’ server.

An X-Service is a component of the MDM application. It can be considered a specific Business contract which would be analogous to a business function. A X_Service contains one or more workflows that together form a logical function or unit. The diagram below represents the basic Application Server MDM Platform environment:

Figure 11: Application Server MDM Platform environment



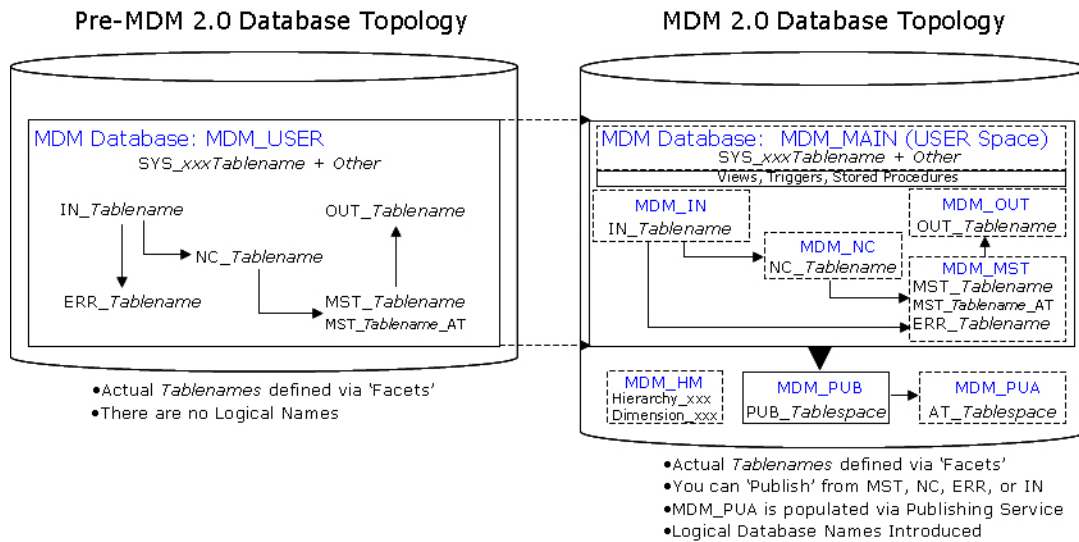
MDM Database Topology

In the previous versions of the Teradata MDM product there was a single database in which all MDM tables were located. That is, all Input Staging, Net Change, Error, Master, and Output Staging were all defined to be in a single Teradata database. With MDM 2.0 a new database topology is being introduced. Please note current MDM implementations need not change their topology.

The MDM database instance can contain separate and distinct databases to house the various permutations of tables as they are related to services. There will be logical databases for which the user can assign names conformant to their naming conventions.

The following diagram below provides a description of that topology:

Figure 12: MDM Database Topology



The MDM 2.0 product does not require strict adherence to the new topology. The topology can remain as in Pre-MDM 2.0, except for the newly introduced feature databases such as, Hierarchy Management (MDM_HM), Publication (MDM_PUB), and Audit (MDM_PUA).

With Teradata MDM 2.0, during installation of an MDM instance, the install script will prompt for the physical names that will be used at that customer's site that correspond to the function database. The number and purpose of the databases to be created is a function of the Service to be offered in this MDM instance. This permits flexibility in data location. Using this schema the customer may choose to either consolidate all tables into a single database or have them spread across several databases or some intermediate combination.

The Metadata table SYS_DB_MAP contains the mapping of the logical db to the physical db. This information has to be ingested into the SCHEMA generation code so that the generated XML contains the appropriate table location. Likewise the generation of the staging tables needs to have this information to create the tables in the appropriate databases. Outward pointing primitive views resident in MDM_Main reference the tables in the dispersed Database topology.

The following table describes the various logical Databases being introduced:

Figure 13: Logical Database Table

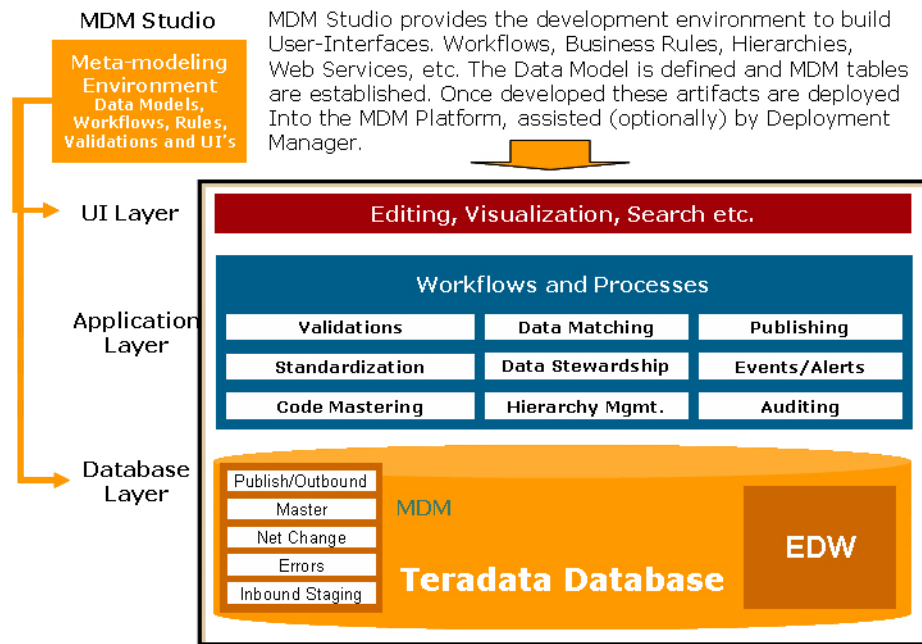


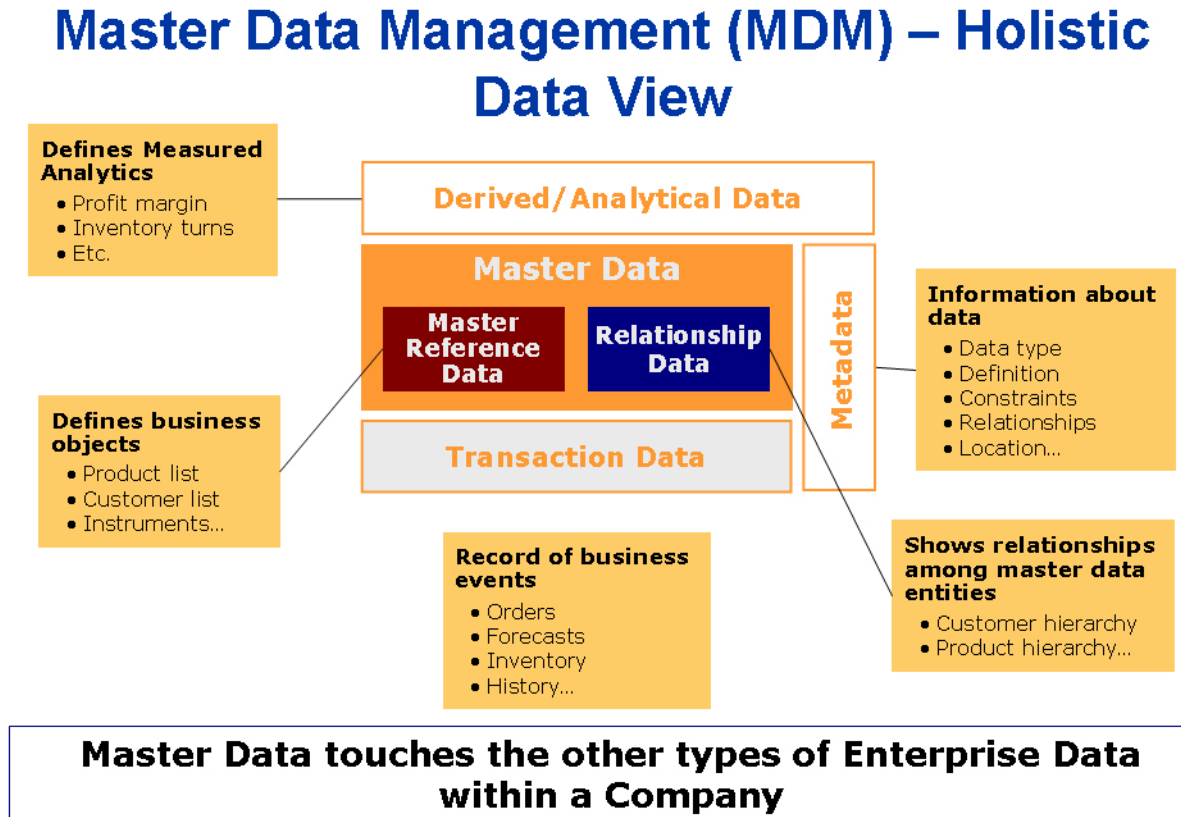
Table 1: MDM Supported Teradata Data type Mappings

MDM Studio	Teradata Database
BOOLEAN	CHAR(1)
CHAR	CHAR(N) CASESPECIFIC
CLOB	CLOB
DATE	DATE
DECIMAL	DECIMAL(n,m)
DOUBLE	FLOAT
ENCRYPTED STRING	VARCHAR(n)
FLOAT	FLOAT
INTEGER	INTEGER
STRING (<64000)	VARCHAR(n) CASESPECIFIC
STRING (>63999)	CLOB
TIMESTAMP	TIMESTAMP(0)
XML	VARCHAR(n)

Data Architecture

Description of the data aspects of the product, that is, what data is located where.

Figure 14: Holistic Data View



Examples of Master Data include:

Figure 15: Master Data examples

Organization	Examples
Sales and Marketing	Customer Record
Service	Warranty Codes, Service Codes
Product Development, Sourcing	Product, Supplier, Item, BoM, Product hierarchies
Finance	Dept. Codes, Cost Centers, Financial hierarchies, Chart of accounts
HR	Employee, Org hierarchies
Misc.	Code masters for common entities like currency, UoM etc.

CHAPTER 2 MDM Development Guidelines

What's In This Chapter

This chapter provides guidelines for development in MDM.

Topics include:

- [Overview](#)
- [Model Development](#)
- [Business Logic Customizations](#)
- [User Interface Customizations](#)
- [Reuse of Code Modules](#)

Overview

The Teradata MDM Application is based on a Service Oriented Architecture (SOA) composed of multiple loosely coupled services.

Each Service has the following development facts associated with it:

- Model
- Workflows
- Rules
- Validations
- Static Data
- Data Persistence Rules

The key service in MDM is the master data management service technically called BCM_MASTER. This service provides the capabilities around input staging, net-change and output staging.

Model Development

The MDM Model is composed of entities, relationships and their associated facets. All aspects of a model can be customized via the MDM Studio.

Model Naming Conventions

- Documents (Tables) and Properties (Columns/Attributes)

- Document logical names and the logical names of the properties (columns) are defined in upper camel case example: Item, ItemLocationMaster. The logical names should convey as much semantic meaning as possible. Exceptions to this are industry standard names like EANNumber etc.
- The physical names should be all upper case and should be abbreviated and logical parts should be separated by _ (underscore). The vowels and repeating letters can be dropped in this e.g. logical property ForecastQuantity can have a physical name of FRCST_QNTY.
- All properties should ideally use a data dictionary type. This enables consistent type semantics.
- Keys
 - The logical name of the keys should be in lower camel case. The recommended naming conventions for the various types of keys are as below:
 - Primary Key: Use the suffix _pk
 - Unique Key: Use the suffix _uk
 - Index: Use the suffix _idx
 - The physical names should be prefixed by PK_, UK_, IDX_.
- Links
 - The logical name of the link which points from the child to the parent entity should be of the form roleName<childEntityLogicalName>To<parentEntityLogicalName>.
 - This will make sure that just by looking at the document link you can understand the relationships. Here the roleName is the role played by the child in this relationship with parent.
 - The reverse link should be of the form <role><parentEntityLogicalName><childEntityLogicalName>
- Facets
 - The facets are mostly check boxes corresponding to design time flags.
 - For display names, the name suitable to that deployment should be provided.
 - For Physical table names, the same coding conventions/standards can apply which apply for Physical table names.

Model Customization Guidelines

- New Documents/Tables that are added for that deployment should be defined in the Model Instance of the customized service.
- The new table logical names should be prefixed with UD (short form for User Defined) and similarly the physical name for these should be starting with UD. This provides easy demarcation of product versus customized entities.
- Any properties that are added as a part of the customization should also be prefixed with ud, and physical column names with UD.
- These tables could reference a data dictionary which is defined for these new tables or existing data dictionaries provided in MDM.

- Customizations should be additive in cases where behavior reuse is expected.
- Customizations can be subtractive in cases where existing business logic is not going to be used and new logic is being introduced.

Business Logic Customizations

Workflows

A service can have multiple workflows corresponding to the business processes owned by that service. Workflows can be triggered by events and can wait on events. Workflows can carry out tasks which are executed by rules.

Coding Convention

- The workflows are placed in the Workflows folder of the service. The workflow file name is recommended to be wf<Service/ModuleName><ProcessName>.xml.
Example: wfManufacturingSupplyChainSetup.xml
- Nodes in the workflow should be logically named based on the business task or event of the workflow.
Example: approveItem
- UI Nodes should be named based on the logical screen name.

Rules

A service can have various rules corresponding to the business logic. The rules can correspond to the public API which can be invoked from other services/external systems or private rules API which can only be invoked from the service.

Coding Convention

- The file which contains rules should be named like rule <Service/ModuleName><Entity><Functionality>.xml.
Example: ruleManufacturingBOMManagement.xml
- Individual Rules should be based on the functionality.
Example: saveBOM, getBOM, deleteBOM

Validations

A rule which is executed upon request can have validations that need to be executed.

Coding Convention

- The validations can be split by request/rule/functionality and can be named as vldn<Service/ModuleName><Entity><Functionality>.xml.

DataPersist Rules

This capability allows declarative specification of persistence rules for an entity in terms of what the create, edit, delete, massupdate api is, what the associated validations are and what alerts need to be raised on success/failure. This mechanism is used in TableEditors and back end data loading.

Coding Convention

- The specification is specified in a single file.
- The file name should be of the type datapersist <servicename>.xml.
Example: datapersistManufacturing.xml

Additional Best Practices on Business Logic Customizations

- Customizing Workflows/Rules/Validations/DataPersistRules/Data
 - This is recommended to be prefixed by the customer/projectName/ud.
 - Studio creates a separate custom service folder to keep the customizations.
 - Using custom service, models and behavior can be inherited and customized.

User Interface Customizations

X2 based

The reference screens packaged with MDM are based on X2. X2 is based on commands which are written in XML script in the .cnd file, and JSP and XSL wherein JSP acts as a container for the cnd to XSL binding. XSL is used to render the XML data that the server returns. It is recommended to use PGL for new UI development.

Coding Convention

- Example: For Resource UI, package your code as follows:
 ..\bcmclient\bcm\resource\resourceDetail.jsp for JSP code
 ..\bcmclient\bcm\resource\xsl\ for XSL code
 ..\bcmclient\WEB-INF\bcm\model\bcm\resource for cnd code

Typically there are two .cnd files associated with every page. The view.cnd which generates XML related to rendering the page and controller.cnd which contains the commands related to button actions. The folders can have subfolders which have similar structures for separation by functionality.

To customize an X2 based user interface, the following steps are recommended:

- 1 The customization can involve .cnd files, the .jsp files or .xsl files.
 - For .cnd file customization, similar custom directory structure can be created under bcmclient\WEB-INF\<customer> and the whole .cnd file can be overridden. The model-attribute.modelpath needs to be modified in WEB-INF\classes\x2.properties.

- For JSP and XSL files their references can be modified in pages.cnd located in WEB-INF\bcm\model\bcm and the corresponding JSP's along with the xsl's can be overridden in the customer directory.

Example: bcmclient/<customer>

PGL based

The UI workflows which are part of the UI_WORKFLOW service should only contain the UI processing logic/validations and should be named as uiWf<ModuleName><ProcessName>.xml.

Example: uiWfManufacturingSupplyChainSetup.xml

- This workflow captures the flow between pages which have a PGL associated with it. The PGL file can have <ModuleName>PageName.pgl as the name of the file.
- This workflow can callout rules which are again named as any other rule. The UI workflow should callout a rule on the UI_WORKFLOW service which should just do presentation logic. The rule should further callout to a back-end service for server side business logic.

To customize a PGL-based user interface, the following steps are recommended:

- 1 The customization done can involve the workflow files, the .PGL files or the rules associated with the workflow.
- 2 The workflow can be redefined in the customization either by redefining the whole workflow with the same name or at a rule level where the rule that gets called is overridden.
- 3 The name of the workflow file, pgl file and the rule file that gets overridden should be prefixed by customerName/Project Name/ud.

Best Practices - PGL based UI Workflow Development

- The code in the workflow should not have extensive business logic. It should ideally just be a call to various methods which actually implement the business logic.
- All requests to the respective services (bcminputstaging, bcmmaster etc.) should happen using the UI rules or subworkflows.

For example the following code is not advisable to be written in the workflow:

```
<IF_TEST Test="count ($suppIDDocVar/*)!= 0">
  <THEN>
    <GET_DOC_SMART Name="Supplier" ServiceName="BCM_MASTER"
AssignToVar="suppliersDocNameVar">
      <TO_XML DocVar="suppIDDocVar"/>
    </GET_DOC_SMART>
    <XML_SORT SortKey="supplierID/@Value" DataType="String"
AssignToVar="suppliersDocNameVar">
      <TO_XML SelectList="$suppliersDocNameVar/*"/>
    </XML_SORT>
  </THEN>
<ELSE>
  <TO_DOCVAR AssignToVar="suppliersDocNameVar">
```

```

        <ROOT/>
      </TO_DOCVAR>
    </ELSE>
  </IF_TEST>

```

Note: The code above must always be written in an API and the API can be invoked using UI rule. Workflow should never have code which makes db calls.

The code above can be written like:

```

<IF_TEST Test="count ($suppIDDocVar/*) != 0">
  <THEN>
    <REQUEST Name="getSortedListOfSuppliers" AssignToVar="
suppliersDocNameVar"
      <TO_XML DocVar="suppIDDocVar"/>
    </REQUEST>
  </THEN>
<ELSE>
  <TO_DOCVAR AssignToVar="suppliersDocNameVar">
    <ROOT/>
  </TO_DOCVAR>
</ELSE>
</IF_TEST>

```

The rule `getSortedListOfSuppliers` is a UI rule and this rule calls an API, which actually does the GET_DOC part.

The UI rule should ideally look like the following:

```

<DEFINE_METHOD Name="getSortedListOfSuppliers">
  <RULE>
    <ACTION>
      <REQUEST Name="getListOfSuppliers" ServiceName="BCM_INPUT_STAGING"
AssignToVar="listOfSuppliers">
        <TO_XML SelectList="$thisParam/*"/>
      </REQUEST>
      <ADD_CHILDREN DocVar="thisReturn" FromSelectList="$listOfSuppliers/*"/>
    </ACTION>
  </RULE>
</DEFINE_METHOD>

```

The API which actually does a database operation looks like the following:

```

<DEFINE_METHOD Name="getSortedListOfSuppliers">
  <RULE>
    <ACTION>
      <GET_DOC_SMART Name="Supplier" ServiceName="BCM_MASTER"
AssignToVar="suppliersDocNameVar">
        <TO_XML DocVar="suppIDDocVar"/>
      </GET_DOC_SMART>
      <XML_SORT SortKey="supplierID/@Value" DataType="String"
AssignToVar="suppliersDocNameVar">
        <TO_XML SelectList="$suppliersDocNameVar/*"/>
      </XML_SORT>
      <ADD_CHILDREN DocVar="thisReturn" FromSelectList="$ suppliersDocNameVar
/*"/>
    </ACTION>
  </RULE>
</DEFINE_METHOD>

```

Note: Writing code in the above manner ensures modularity and promotes re-use. Putting business logic into the workflow reduces the modularity and maintainability.

Alternate method of writing UI workflows

- The previous approach to writing UI workflows was to have calls to various rules using UI rules from the workflow.
- A more intuitive way of writing workflows can be to have sub workflows being invoked from the main workflow.
- The sub workflow can be invoked like how any other API is invoked. This method is more visually intuitive as the workflow can be viewed and the flow is easily understood.

Example:

This is how a sub workflow is invoked from a workflow. The name of the sub workflow is "createPO".

```
<REQUEST Name="createPo" ServiceName="ORDER_ADMIN"
AssignToVar="response" HandleException="yes">
  <TO_XML SelectList="$currentOrder"/>
</REQUEST>
```

To enable the sub workflow to be invoked from a workflow, the following must be done.

```
<workflow mgcCount="489" ShowInBreadCrumbs="true" Version="1.00.01"
Name="poCreateWf" Description="true">
  <variables>
    <variable Name="lockingNeeded" Type="string" Comment=""/>
    <variable Name="poDoc" Type="xml" Comment=""/>
    <variable Name="keepIds" Type="boolean" Comment=""/>
  </variables>
  <nodes>
    <start Name="start2" Descriptor="createPo" IsDefault="false">
      <actions>
        <action Name="assign PO Doc">
          <SET Var="poDoc" FromValue="{ $thisParam/ORDER}"/>
          <SET Var="lockingNeeded" FromValue="{ $thisParam/
LOCKING_NEEDED/@Value}"/>
          <SET Var="keepIds" FromValue="{boolean($poDoc/KEEP_IDS/@Value,
false())}"/>
        </action>
      </actions>
      <next_nodes>
        <next_node Name="AcquireLocks" Description=""/>
      </next_nodes>
    </start>
```

Reuse of Code Modules

The benefits of reusing code are obvious. For every application you can reuse the generic functions that you created for the one before. This saves time, improves the features in your program, and generally makes for more cost-effective programming. As your libraries grow it becomes easier and easier to lay the framework for more sophisticated applications, without any effort at all. Most third party products employ this philosophy, but section explores how you can take advantage of these features in MDM.

The MDM uses the concept of service for execution of workflows. This service is also called as XService. A service satisfies a specific business contract. For example, an Order Capture service caters to workflows for capturing customer orders. Multiple services along with a UI make up a business application. The XService consists of XDocuments and XRules. A single instance of the MDM can deploy multiple XServices. You can create the following types of service:

- **Model Based**—the model based service is a standard MDM service type which can be used to build an application.
- **MDM Based**—the MDM based service is a service type which is customized for the MDM application. MDM related functions such as automatic creation of Input staging, NetChange staging, and Output staging child-services are available in this type of service.
- **Document Based**—the document based service is a service which uses the X-Documents for schema definition. You can insert documents and document views in this type of service. This helps in the backward compatibility of document-based services developed with earlier versions of MDM. You can view a list of archived service with this kind of service.

Creating Reusable Modules

Perform the following steps to create reusable module:

- 1 **Packaging jar file**
 - Create a service using MDM studio
 - Insert workflow files, rule files and customize
 - Archive it as a jar file
- 2 **Reusing packaged jar file**
 - Import service from archive file
 - Setting class path for newly created service

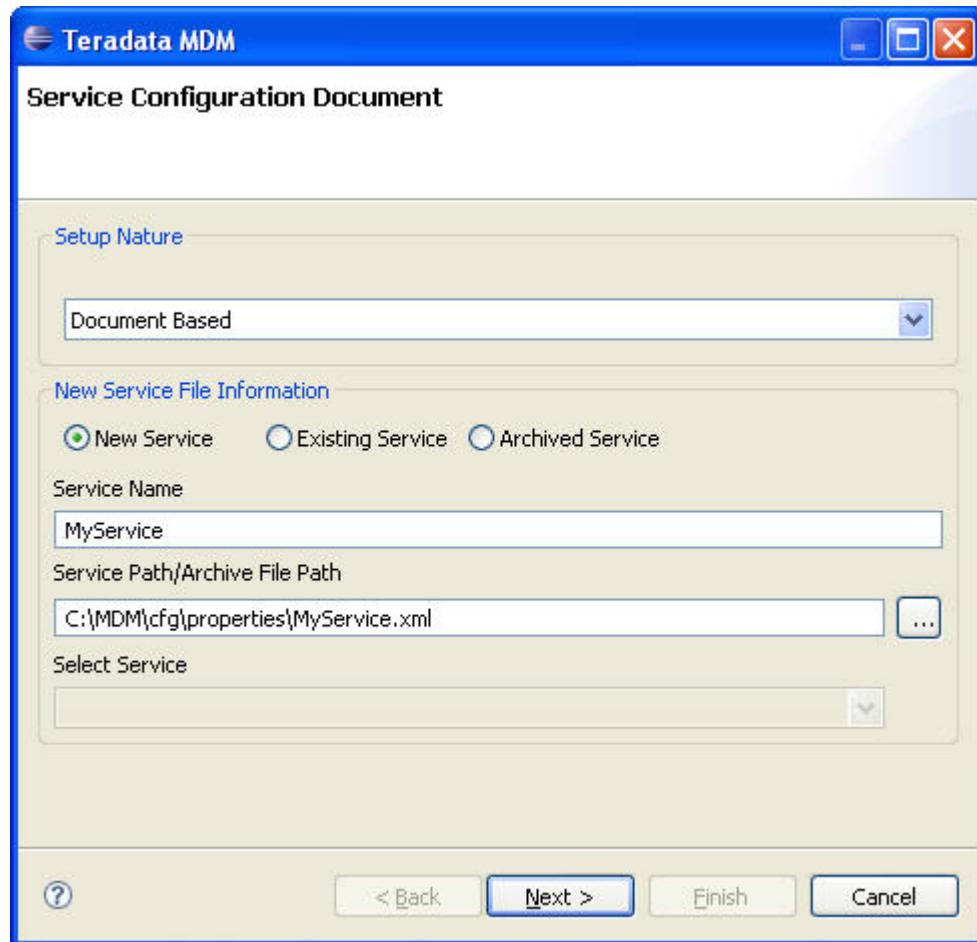
Packaging Service as jar for Reusing Document based Service

Perform the following steps for packaging service as jar for reusing:

- 1 **Create a document based service using MDM studio**
 - Open MDM studio and create a document based service. Refer *Chapter 4 Process Modelling in MDM Platform Studio User Guide* for details.

The [Figure 16](#) and [Figure 17](#) displays the steps involved in creating a document based service.

Figure 16: Service Setup-Setup Nature



The image shows a Windows-style dialog box titled "Teradata MDM" with a subtitle "Service Configuration Document". The dialog is divided into two main sections. The first section, "Setup Nature", contains a dropdown menu currently set to "Document Based". The second section, "New Service File Information", contains three radio buttons: "New Service" (which is selected), "Existing Service", and "Archived Service". Below these are three text input fields: "Service Name" with the value "MyService", "Service Path/Archive File Path" with the value "C:\MDM\cfg\properties\MyService.xml", and "Select Service" which is an empty dropdown. At the bottom of the dialog are four buttons: a help button (question mark icon), "< Back", "Next >", and "Cancel".

Teradata MDM

Service Configuration Document

Setup Nature

Document Based

New Service File Information

☒ New Service ☐ Existing Service ☐ Archived Service

Service Name

MyService

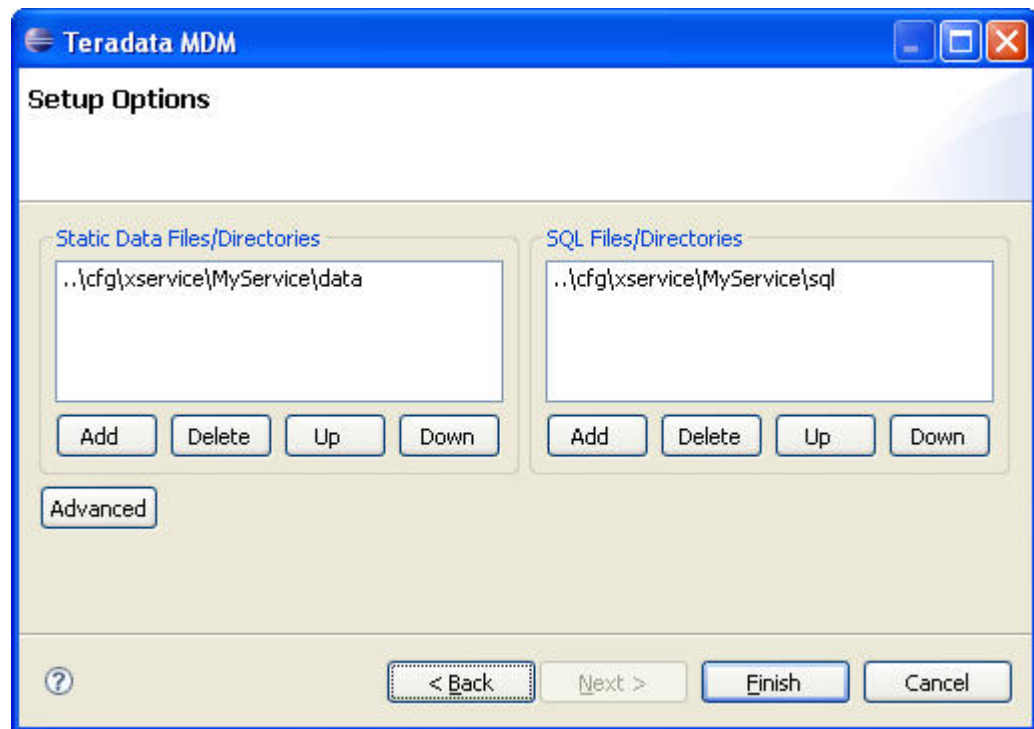
Service Path/Archive File Path

C:\MDM\cfg\properties\MyService.xml

Select Service

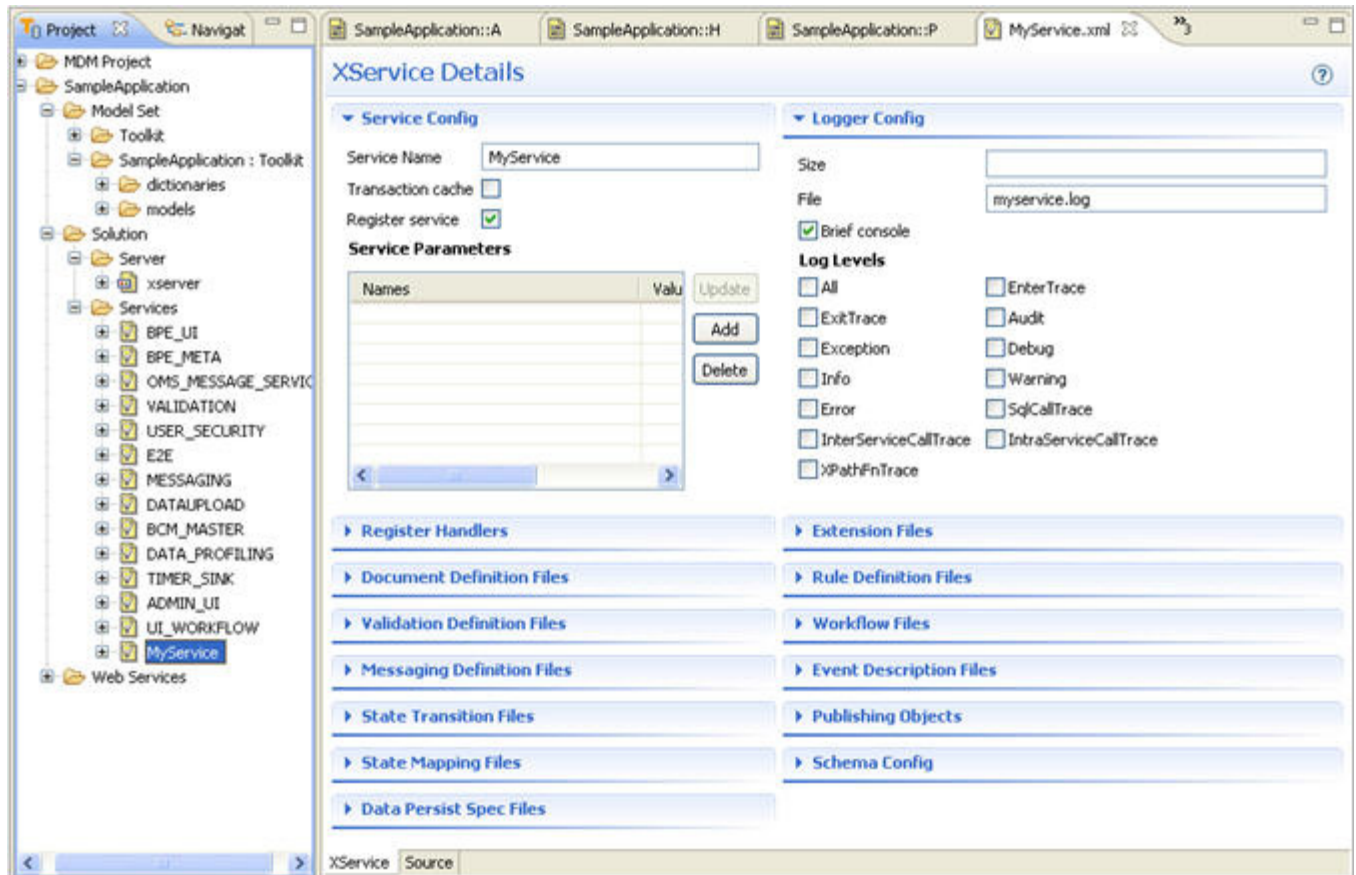
? < Back Next > Finish Cancel

Figure 17: Service Setup-Setup Options



The [Figure 18](#) displays the service added in the left navigation pane. Double-click on the added service and in the right pane, configure the Logger details accordingly.

Figure 18: Service Added



- 2 Insert workflow files, rule files to the newly added service and customize as per requirements.

Refer *Chapter 4 Process Modelling in MDM Platform Studio User Guide* for details.

- 3 Archive service as a jar file.
 - Add `SERVICE_NAME.xml` from `MDM_Install_Directory/custom/APPLICATION_NAME/cfg/properties/` to folder `MDM_Install_Directory/custom/APPLICATION_NAME/cfg/xservice/SERVICE_NAME` and zip the `SERVICE_NAME` folder.
 - Rename the zip file as `SERVICE_NAME.jar`
 - Save the newly created jar file to `MDM_Install_Directory/custom/lib`.

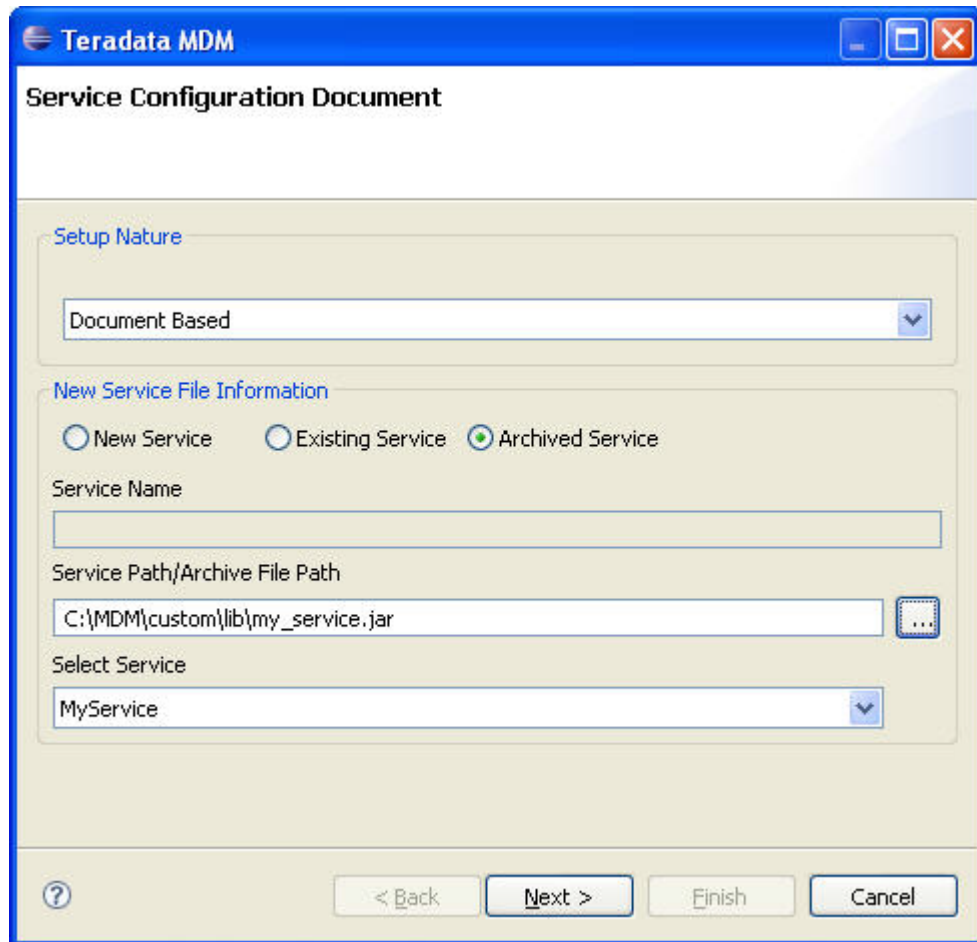
Reusing Packaged jar File

Perform the following steps to reuse the packaged jar file:

- 1 Import document based service from archive file.

Figure 19 and Figure 20 displays the steps involved.

Figure 19: Service Configuration—Archived Service



The image shows a Windows-style dialog box titled "Teradata MDM" with a subtitle "Service Configuration Document". The dialog is divided into two main sections. The first section, "Setup Nature", contains a dropdown menu set to "Document Based". The second section, "New Service File Information", contains three radio buttons: "New Service", "Existing Service", and "Archived Service", with "Archived Service" selected. Below these are three input fields: "Service Name" (empty), "Service Path/Archive File Path" (containing "C:\MDM\custom\lib\my_service.jar" with a browse button), and "Select Service" (a dropdown menu set to "MyService"). At the bottom, there is a help icon, a "< Back" button, a "Next >" button, a "Finish" button, and a "Cancel" button.

Teradata MDM

Service Configuration Document

Setup Nature

Document Based

New Service File Information

☐ New Service ☐ Existing Service ☒ Archived Service

Service Name

Service Path/Archive File Path

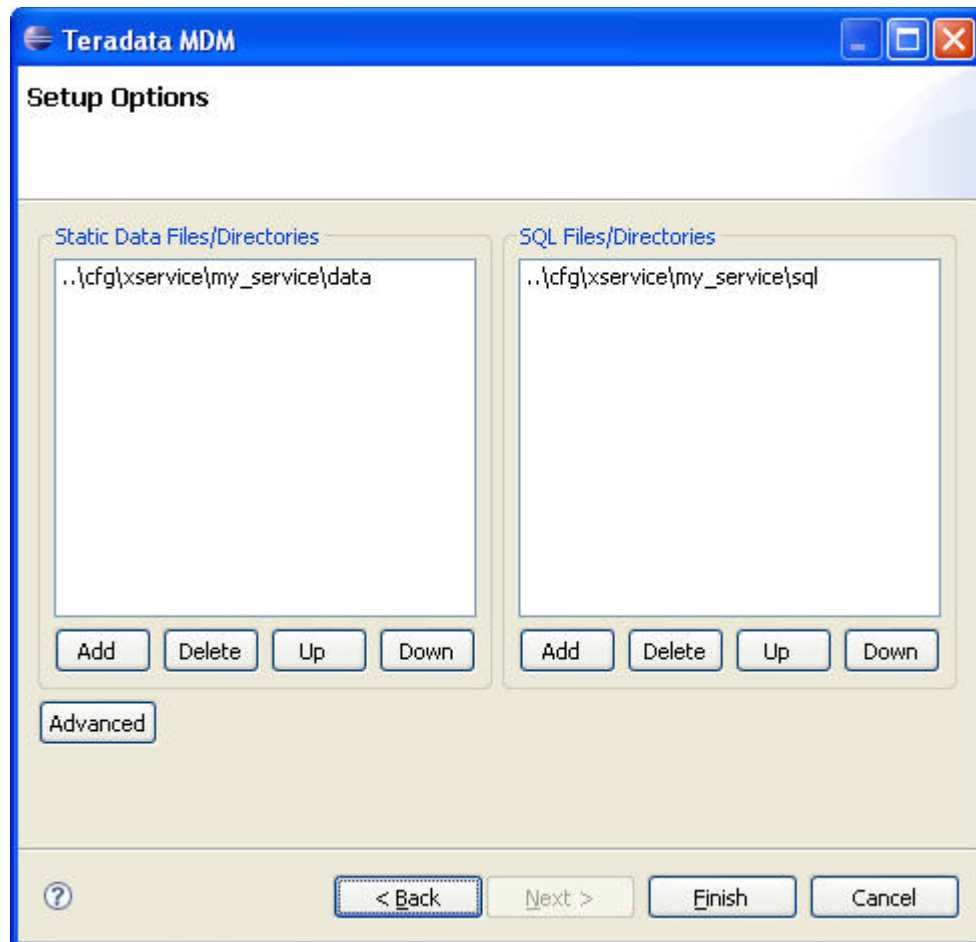
C:\MDM\custom\lib\my_service.jar

Select Service

MyService

? < Back Next > Finish Cancel

Figure 20: Setup Options



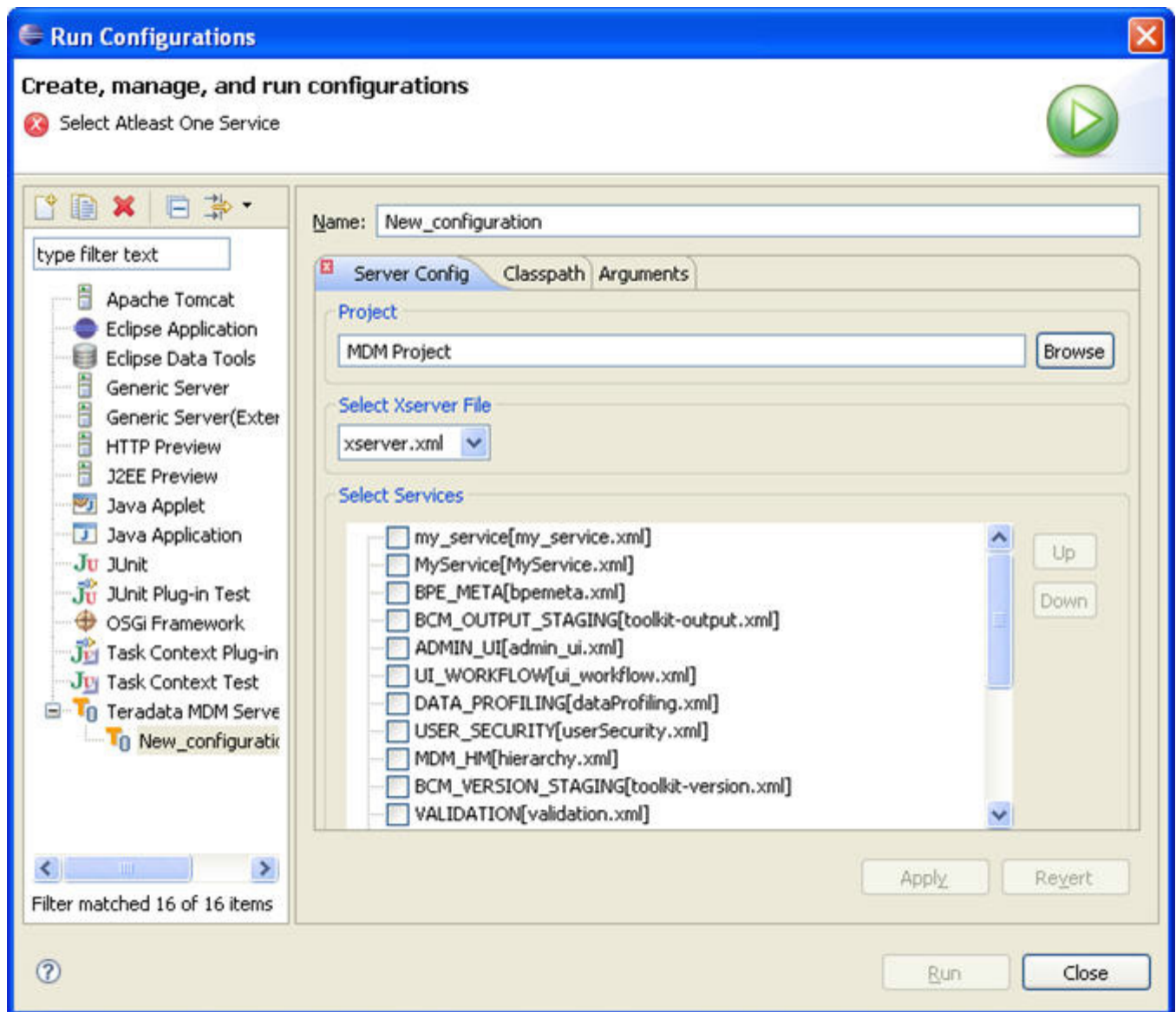
The left navigator displays the added document based service.

2 Setting class path for newly created service

- Start MDM server through studio.
- On the MDM Studio, click **Run** menu and select **Run Configurations**.

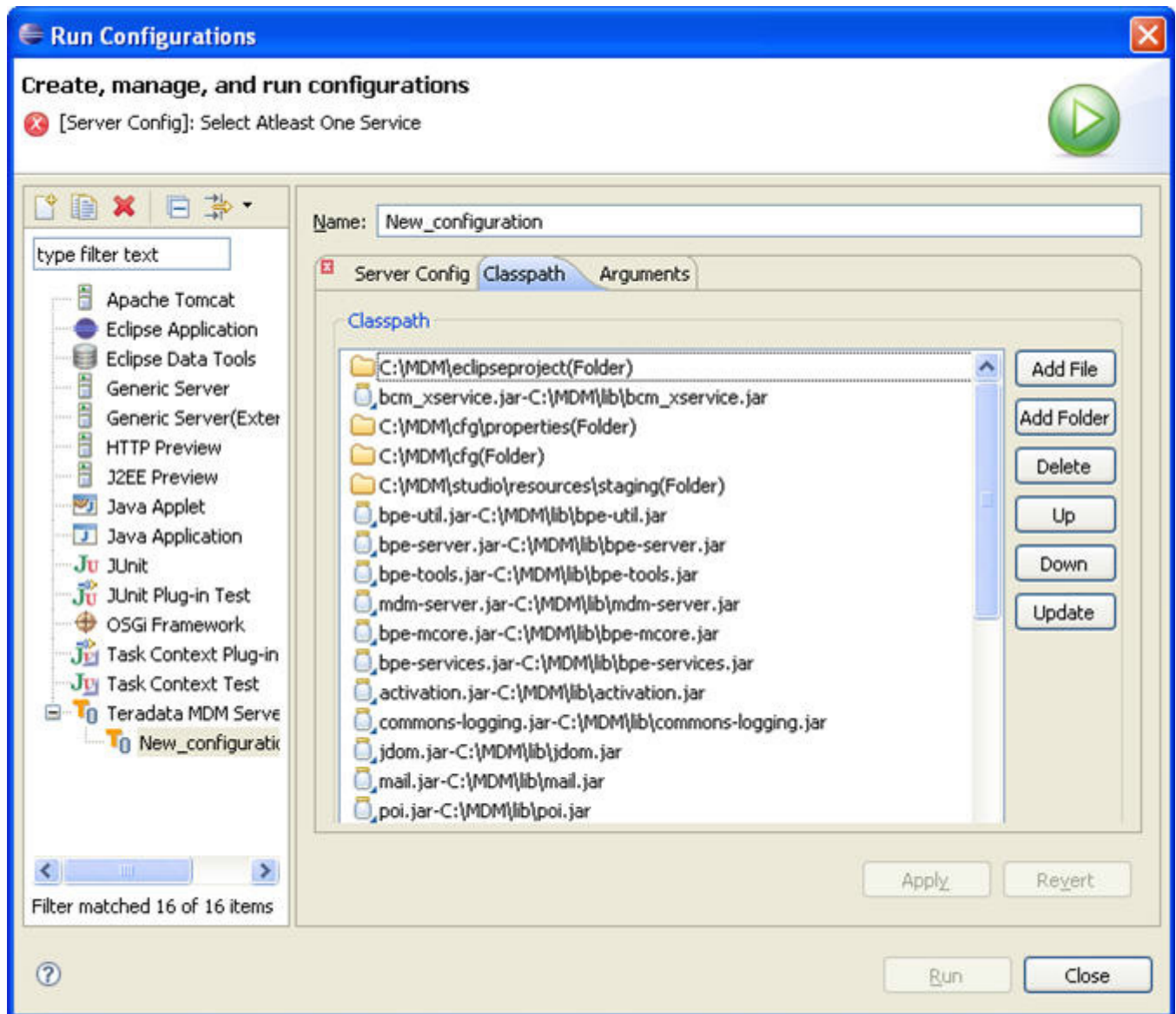
The **Run Configurations** window (Figure 21) is displayed.

Figure 21: Run Configurations



- On the **Run Configurations** window (Figure 21), insert new configuration and select the server from the **Select XServer File** drop-down and select the required services from the **Select Services** pane and click **Classpath** tab as in Figure 22.

Figure 22: Classpath



You can add classpath files, delete and rearrange the classpath files using the appropriate buttons.

- If the MDM server is started through command line, add respective jar file to class path in *bcmenv.bat* and add new service name in *startServices.bat*.

Note: Make sure that you have added the new service in your war file (for web sphere).

Packaging Service as jar for Reusing

MDM Based or Model Based Service

Perform the following steps for packaging service as jar for reusing:

- 1 Create a MDM based or model based service using MDM studio
 - Open MDM studio and create a MDM or Model based service. Refer *Chapter 4 Process Modelling in MDM Platform Studio User Guide* for details.
- The [Figure 23](#) to [Figure 25](#) displays the steps involved in creating a MDM based service.

Figure 23: Service Setup-Setup Nature

The screenshot shows the 'Teradata MDM' window with the 'Service Configuration Document' tab active. The 'Setup Nature' section has a dropdown menu set to 'MDM Based'. The 'New Service File Information' section contains three radio buttons: 'New Service' (selected), 'Existing Service', and 'Archived Service'. Below these are text fields for 'Service Name' (containing 'MyService'), 'Service Path/Archive File Path' (containing 'C:\MDM\cfg\properties\MyService.xml'), and a 'Select Service' dropdown menu. At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

Figure 24: Service Setup-Setup Model Instance

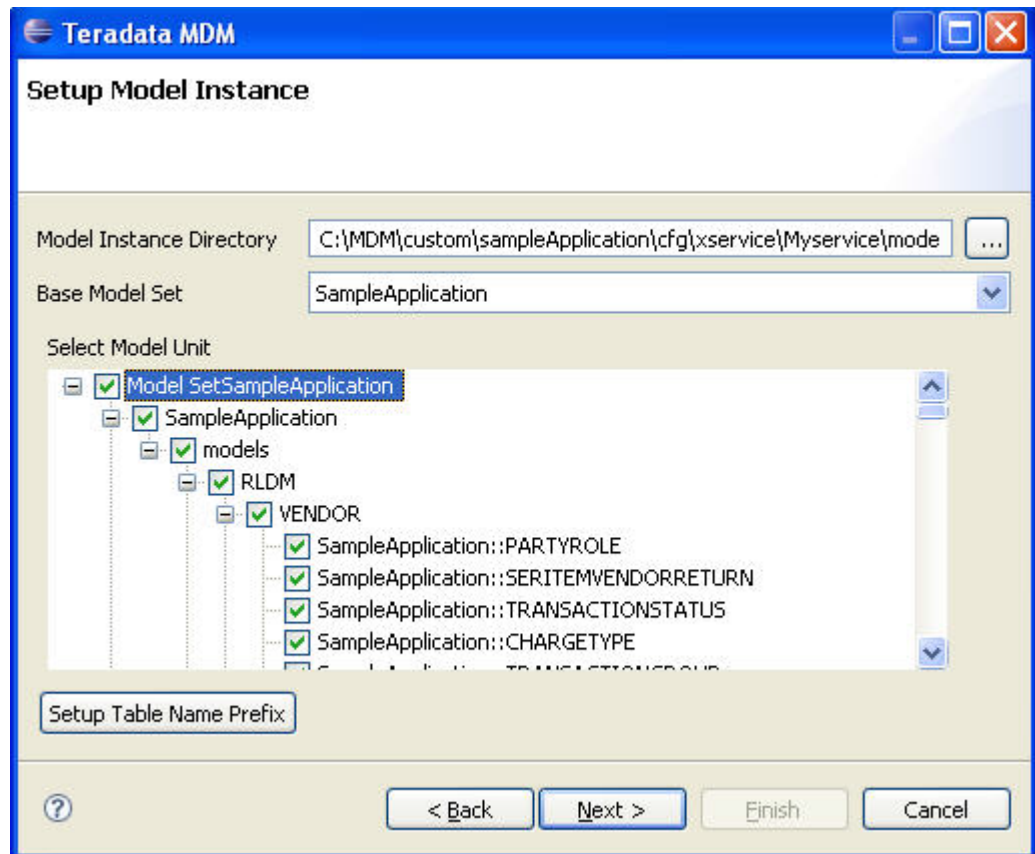
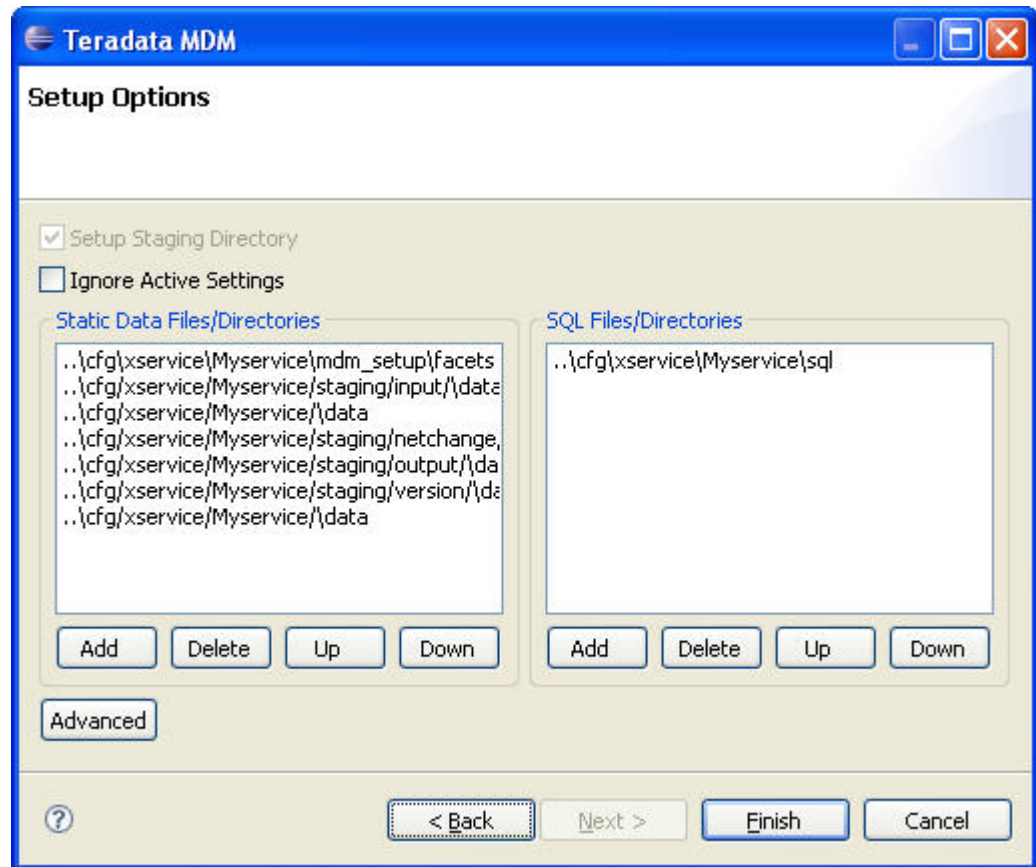
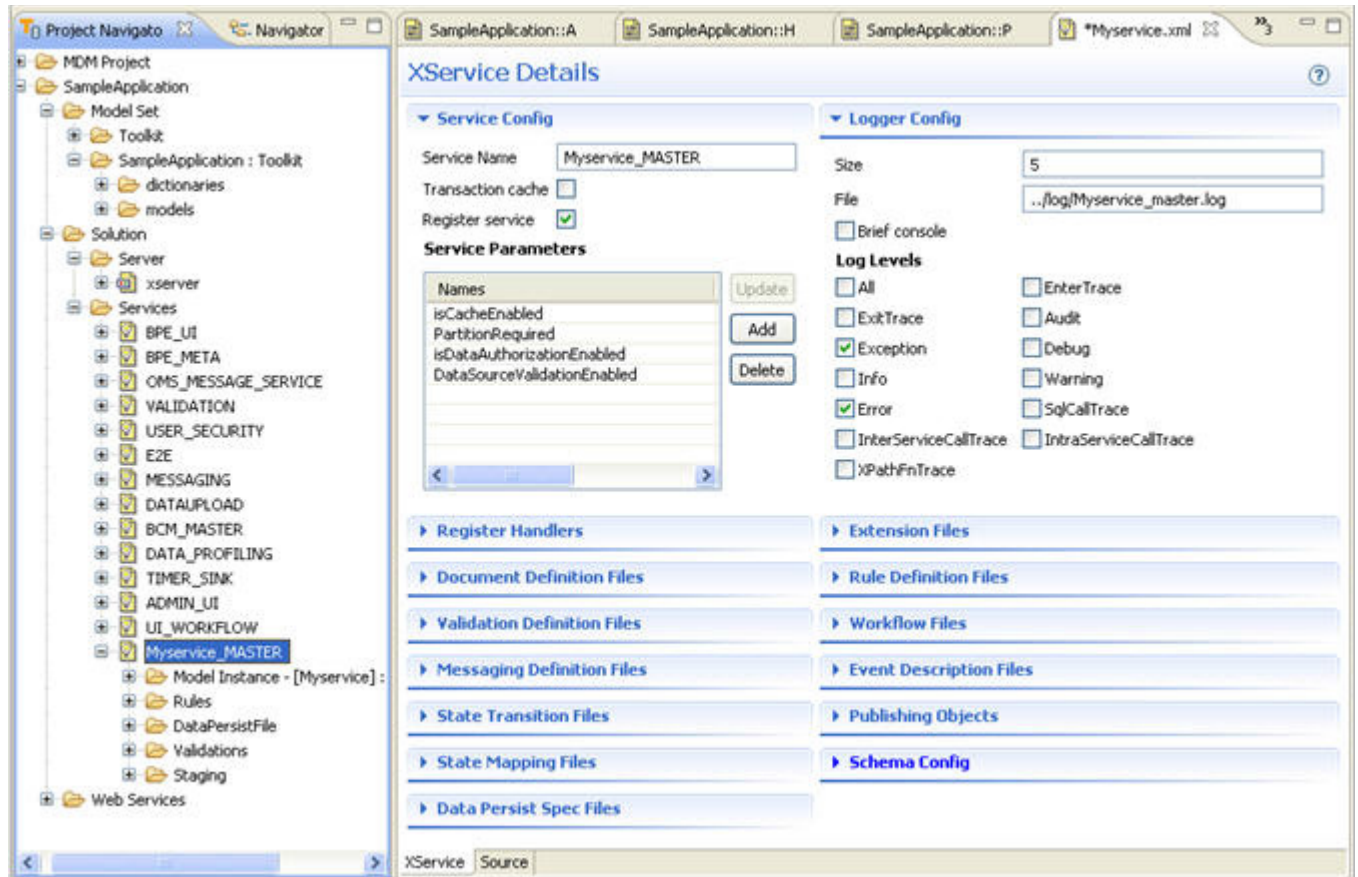


Figure 25: Service Setup-Setup Options



The [Figure 26](#) displays the service added in the left navigation pane. Double-click on the added service and in the right pane, configure the Logger details accordingly.

Figure 26: Service Added



- 2 Insert workflow files, rule files to the newly added service and customize as per requirements.

Refer *Chapter 4 Process Modelling in MDM Platform Studio User Guide* for details.

- 3 Archive Model based or MDM based service as a zip file
 - Create a temp folder.
 - Create cfg/properties folder under temp folder.
 - Copy *APPLICATION_BASE/cfg/properties/SERVICE_NAME.xml* to *temp/cfg/properties* folder.
 - Create a folder called xservice under temp folder.
 - Copy *APPLICATION_BASE/cfg/xservice /SERVICE_NAME* folder to *temp/xservice* folder
 - Zip temp/xservice folder as *SERVICE_NAME.jar*.
 - Create a folder called lib under the temp folder, cut and paste the *SERVICE_NAME.jar* file into it
 - Optionally delete temp/xservice folder since its no longer required.
 - Zip temp/cfg and temp/lib folder as *SERVICE_NAME.zip*.

Note: Models can't be packaged as zip files, so all the model related files need to be copied manually to the new respective location.

Reusing Packaged jar File

Perform the following steps to reuse the packaged jar file:

- 1 Unzip service file—unzip SERVICE_NAME.zip to respective APPLICATION folder
Make sure to extract the files to lib and cfg/properties folders. [Figure 27](#) displays the option of selecting MDM based service from the existing service.

Figure 27: Service Configuration—Existing Service

The screenshot shows the 'Teradata MDM' window titled 'Service Configuration Document'. It contains several sections: 'Setup Nature' with a dropdown menu set to 'MDM Based'; 'New Service File Information' with three radio buttons where 'Existing Service' is selected; a 'Service Name' text field; a 'Service Path/Archive File Path' text field containing 'C:\MDM\cfg\properties\<service>.xml'; and a 'Select Service' dropdown menu. At the bottom, there are four buttons: '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

The remaining steps remain the same as above. The left navigation pane displays the newly created service. For more details steps, refer to *MDM Platform Studio User Guide*.

- 2 Setting class path for newly created service—see step 2 in section [“Reusing Packaged jar File”](#)

CHAPTER 3 Development of an MDM Application

What's In This Chapter

The purpose of this Chapter is to provide the known Best Practices, Key Considerations, and Recommendations. Development practices, procedures, and methodologies may be different at any given customer site. The intent of this developers guide is to not replace or change any practices/methodologies, but rather, to relate MDM to those practices. The Teradata MDM Solution provides a proven development and implementation methodology with the Teradata MDM Center of Expertise (MDM CoE). Teradata recommends engaging the MDM CoE for everything from consulting, planning, execution, and delivery. Specific topics in this chapter include:

Topics include:

- [Overview](#)
- [Building an Application in MDM Studio](#)

Overview

This section describes the Key considerations and recommendations related to the scoping of a MDM Implementation, Best Practices in the Planning of a MDM Implementation, Skill Set Recommendations, Best practices in Managing a MDM implementation, References to Teradata Methodologies, and general Recommendations and Best Practices.

Scoping an MDM Application

There are obviously many different techniques and considerations when determining the scope of any application. Regardless of the technique to define the scope of the MDM implementation, the scope should be aligned directly to the MDM Strategy. The MDM Strategy should embrace the Business Analysis. The Business Analysis must define the Goals, Objectives, Business Improvement Opportunities, Critical Success Factors, Key Performance Indicators, and any Constraints. The Business Strategy, Business Requirements and MDM Implementation Plan must be understood and agreed upon before development begins. Teradata MDM implementations utilize the Proof of Concept (POC) engagements to facilitate this understanding and planning. The Teradata Professional Services MDM implementation methodology has been created and proven to support this approach, with both processes and resource experience.

Planning an MDM Application

Project planning for a MDM implementation is equivalent to most all business application development efforts. The utilization of the Teradata MDM Studio and taking advantage of its capabilities provides an agile iterative approach to the development itself. The Teradata MDM Studio development capabilities should be embraced within the project plan. For instance, MDM Studio typically starts with creating/defining the Data Model that will be used for the application and workflows, therefore organizing the project plan to define and create the Data model should be scheduled before UI's are built. Furthermore, if an Erwin data model is available for the desired subject area, then planning to ingest this model through the Open Model Ingestion (OMI) feature should be planned. Refer to *MDM Platform Studio User Guide* for details on the approach. Basic planning tasks such as review cycles, test plans, deployment planning, and application development milestones still must be incorporated into the project plan. Major project milestones such as requirements definition, design, development, test, integration, documentation, operational implementation are always required.

As Teradata MDM implementation is very centric to an Enterprise Data Warehouse implementation. Project planning steps or activities should consider orientation to Data Subject areas, data quality for that data and then the business processes pertaining to the subject area and in the scope of the MDM implementation.

When building the MDM implementation plan the following skill sets requirements should be sought.

Skill Set Requirements

A Teradata MDM Implementation requires minimally the following skill sets:

- Project Manager
- Solution Architecture
- Technical Leadership
- Business Modeling
- Data Modeling
- Analytical Modeling
- MDM Application Development
- Teradata Database

The experience for each of the above skill sets will impact the success and risk factors of the MDM implementation proportionally.

Additional skill sets include:

- Workflow Design and Development
- Web Based Networking and Administration
- Test Specifications and Planning
- Target Systems Interface Planning (i.e. SAP XI)
- Security Administration

The Teradata Professional Services team provides these experienced skill sets.

Recommendations and Best Practices

The following can be generalized high-level best practices for any MDM implementation:

- **Ensure Business User involvement** and acceptance wherever possible. The most successful MDM implementations include Business owners are dedicated to the MDM implementation. **Understand the Data Model and the Data Requirements** - Understand the Data Quality requirements and implement as much as practical in the data load process, also known as End-to-End (e2e).
- **Utilize the power of Teradata** wherever applicable. For example, in the various data load workflows use either DB_Persist or DIRECT_SQL_LOAD wherever possible. Even in Table editors use DO_DB_Persist.

The supplied DataLoad Workflow should be re-used or considered the template for all Data Load workflows.

Use Teradata Stored Procedures in 'batch' oriented workflows for complex rule support, by invoking the Stored Procedure Node.

- **Use the MDM 2.0 a Sample Application** is provided. This application contains examples of the major or key features of the MDM Studio usage and the MDM Platform itself. Use this application as a solution accelerator wherever possible.

The Sample Application contains the capability to invoke a Trillium API, provided that the appropriate Trillium software is installed.

- **Keep the business workflows simple.** Workflows can call or embed other workflows. Exploit this capability to re-use basic nodes or business processes.

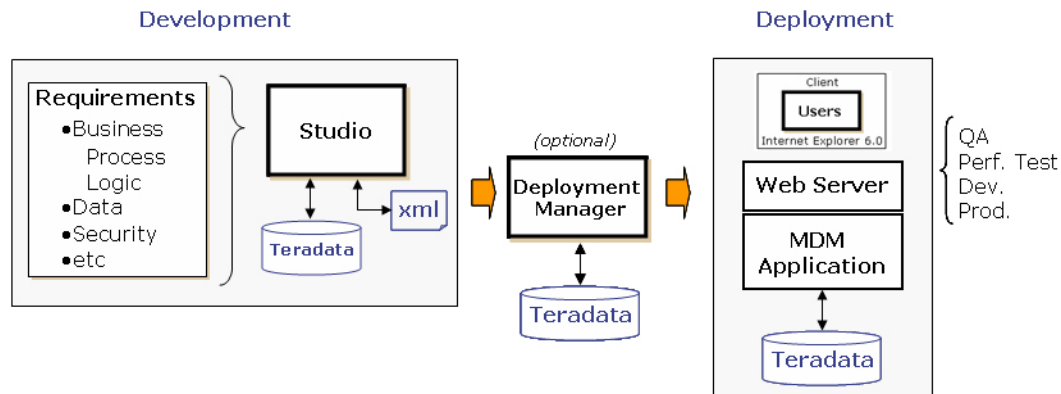
Building an Application in MDM Studio

Overview

Teradata Studio is the development or design-time component for designing, building, debugging, and maintaining MDM workflows, Data models, and User-Interfaces that together define a Business process, which become a MDM Application. The MDM Studio is a Java Swing application that enables the development team to create a MDM application. The Teradata Studio utilizes a 'project' based approach to defining the various components of a MDM Application. The project structure is detailed implicitly in the directory structure that contains all of the various xml files.

The basic flow of the development effort in Studio, starts with defining the Data Model. The Business Requirements are defined, including the various Process steps and business logic. Once these requirements are understood, the development of the Workflows, and User Interfaces can occur. After the workflows and UIs are approved, they can be migrated to a target environment, via Deployment Manager, for further QA testing or possibly Production.

Figure 28: Development Flow



The general flow of development within Studio starts with defining or creating the data model that will be used by the application. The data model can be defined explicitly by Studio, or you can ingest an Erwin model, or import Teradata database table definitions.

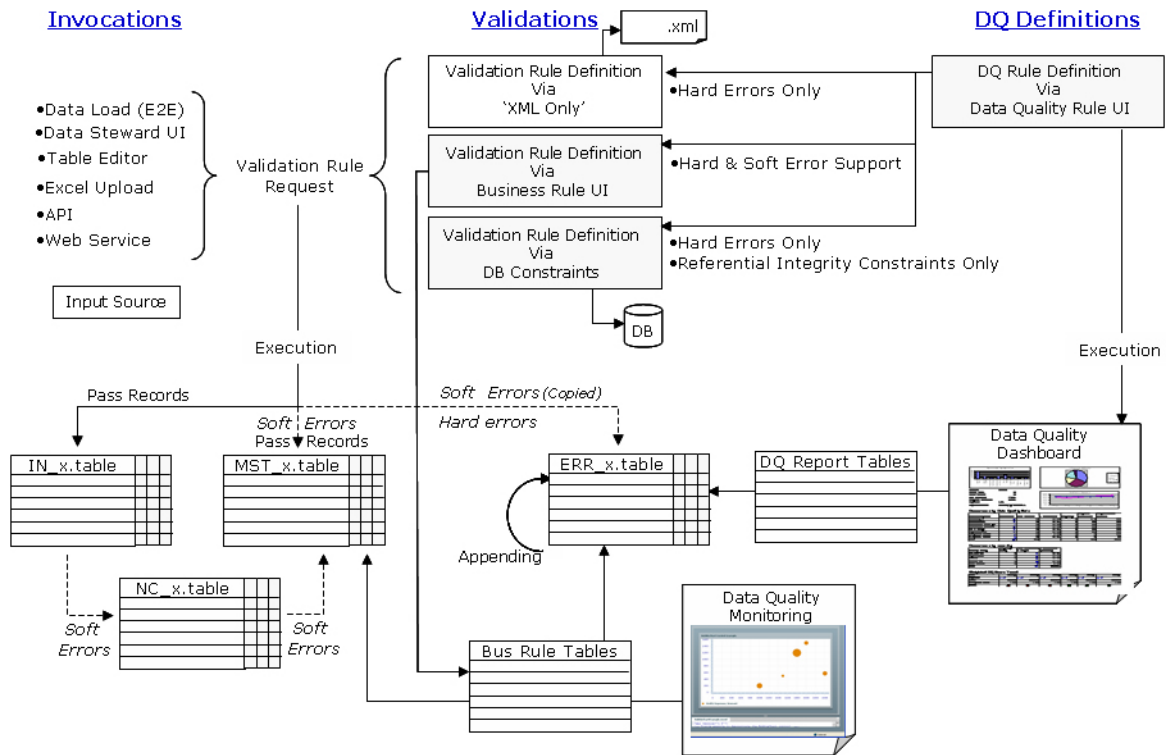
The incoming data can come from internal EDW database(s)/tables or any external system(s) via a file. The file can be supplied by an ETL tool or Teradata Utility. These files are known as 'Source' data and will 'land' in the MDM Input Staging area. The tables defined in the MDM Input staging are defined/generated when the schema generation (Full or Incremental) process was executed in Studio. Master Data Records can come from other 'Source' systems in a 'batch' load manner via Input Staging, or they can be created in Data Steward User Interfaces, or MDM application provided Table Editors, or they can be brought in through defined API's, Web Services or Excel.

Once a Data model is defined, 'Validation' rules are typically defined. Validation Rules can be defined with levels of severity, from Soft errors (Warnings) to highest severity. Typically, 'Warnings' can still be considered Master Records and continue processing, while the rest are sent to Error Tables. These validation rules define 'what is' the proper or accepted quality of the incoming Data. The Business Rules User Interface is provided in Studio, as well as, with the MDM application. This Rules builder provides a generalized mechanism to create SQL based validation rules on MDM specific tables. Please note that use of xml via a xml template for validation rules is still supported.

Teradata MDM provides Data Quality Reporting and Data Quality Monitoring capabilities. Data Quality Reports provide insight into the Quality of records received, typically from source systems, that are defined to be severe. These records are placed into the corresponding Error Table. The Data Quality Monitoring provides insight into Master records that have 'Warnings' due to quality issues or validation rules.

Both Data Quality Reporting and Data Monitoring on MDM Application provided functions, they are not necessary to 'build' in Studio. The diagram below represents a high-level description of the Data Quality related functionality of MDM:

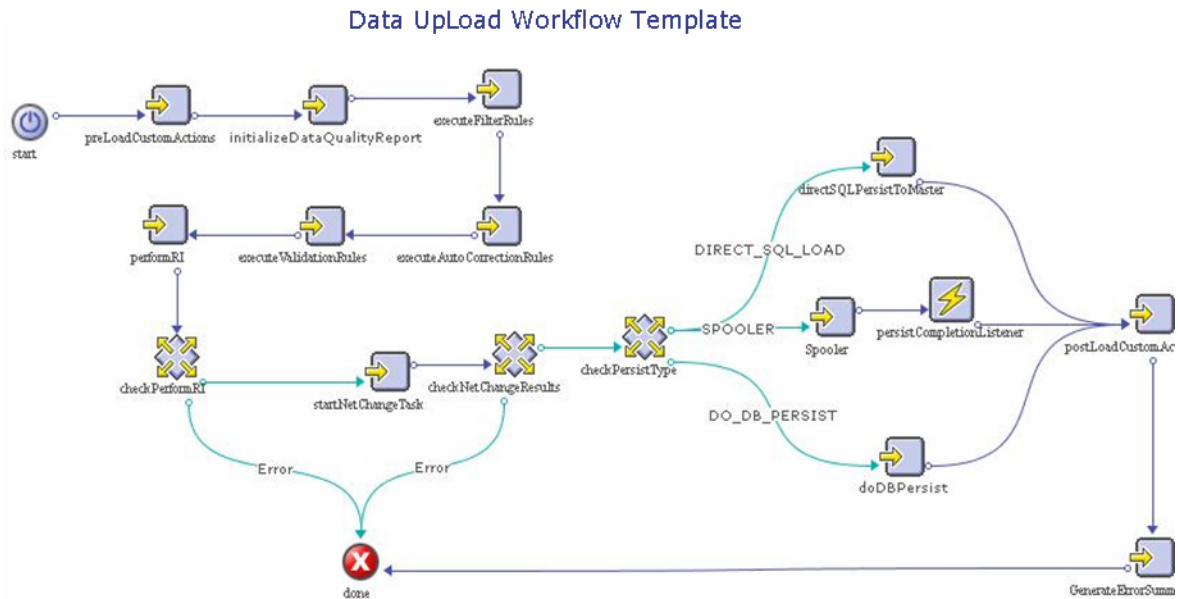
Figure 29: Data Quality related functionality



After the data (or any 'designed' / subset of the data model) is placed in the MDM Staging area, a Data Load workflow can be executed. A standard or 'Template' Data Load workflow is provided with the product. The Input data is processed via the Data Load workflow. Validation rule successful records are sent to the Net Change tables. The records from these tables are then 'UPSERTE'D into the appropriate Master tables. Please note that Input Staging, Error, Net Change, and Master tables all are defined through Schema generation. All have the same Primary Key structure except the Error tables, hence all of these table receive the benefits of Teradata's architecture.

The diagram below represents the Data Load template workflow:

Figure 30: Data Load Template Workflow



It is recommended that validation rules for the large or bulk data that is sourced from other systems utilize a 'Data Load' workflow and the 'DO_DB_Persist' or DIRECT_SQL_LOAD nodes to load data.

Data Load Workflows can be constructed for every table or a combination of tables, based upon the design of the MDM application, which in turn, is based upon the business requirements and source system constraints. The most important consideration is to ensure that once the 'records' have passed the Data Load they are 'ready' for any subsequent processing or reporting. For each node represented above, examples can be found with the MDM Sample application.

The Business requirements should define the Business processes and the Business Logic or Rules that are required for the MDM Application. The business process will be supported via Workflows and within the workflows there may be API's, Web Services, events, User Interfaces and other supported services. In addition, a workflow can call or embed other workflows. These capabilities ensure that the Teradata MDM workflows can satisfy every business requirement. The workflows are created in Studio, as well as any other service. The ["Section B—Sample Application"](#) and *MDM Platform Studio User Guide* contain an exhaustive detail list and descriptions of all of the various feature and functions provided.

It is recommended that the Business processes and logic/rules are defined in business terms during the requirements definition phase of the project. They should be defined in business terms and free from any perceived technological constraints. They should taken into the consideration the owners or responsible functions desired process and approvals, as applicable. The reasoning behind this is one of the value differentiates of the Teradata MDM product. That is, functional extensibility through our Open Services Oriented Architecture where MDM Studio supports the development and usage of Web Services, Java Code, and APIs.

MDM Studio provides the developer with the capability to develop and employ Workflows, User Interfaces, Web Services, Java Code, API's, as well as supplied functionality. The creativity of the developer should only be molded by the business requirements of the application. As stated previously, the Teradata MDM product provides a Sample Application that contains examples of the key functionality of the product.

In order to understand the landscape of creativity the various MDM elements, their operations, and MDM Services are highlighted below. This section is intended to be an overview. Refer to the detailed documentation for further information.

MDM Studio provides a significant amount of functionality and features that enhance the development experience. The functionality can be organized into MDM elements. Each of these MDM Elements are defined by xml schemas and are located in the appropriate directory structure(s) within Studio.

MDM Development Elements

- X-Documents: Description of the Data model or documents in the system
- X-Rules: Definition of the Business Rules relative to the X-Documentation
- X-Operations: Definition of the various operations that are supported
- X-Path: Contains a compact, non-xml syntax to facilitate use within URLs and xml attribute values. X-Path operates on the abstract, logical structure of an xml document.

X-Documents

The primary purpose of X-Documents is to:

- Provide the basis for database schema referential and integrity constraints
- Provide transparency to the physical database, as the xml specification is not database specific.
- Supports the relationships to other x-Documents; 1-1, 1-n, n-n.
- Extensible at deployment time to include additional properties and relationships.
- Audit trail and Dynamic attributes can be 'turned on' at deployment time.
- Provides querying and persistent business document instances – XML based data manipulation (DML) statements.

For details on the following, refer X-Documents in *MDM Platform Reference Guide.pdf*

- Surrogate Keys
- Document Index
- Document Properties
- Documents Links
- Foreign Key Constraints
- Overlay Documents
- Audit Trail
- Document Constraints

X-Rules

The primary purpose is to provide the xml Specification for expressing Business logic in Workflows. X-Rules can consist of Variables, Expressions, Conditional Statements, Specific Business Requests, and Predefined Requests.

X-Rules support Var, DocVar, Value, Property, Attribute, Service, Global, other (such as thisParam), and specific (such as username, REDIRECT_URL) variable types.

Conditional statements are supported and are defined as a statement that evaluates to a true or false. The conditional statement can be simple or compound.

Specific Business Requests are defined as one or more X-Rules, each containing a set of conditions and actions. The business request is analogous to a function in a programming language. It is called by an external entity, may take parameter as input, perform an action, and return an output. There are 8 business requests defined:

- DEFINE_METHOD – a generic request that contains one or more x-rules
- DEFINE_NOTIFICATION – used to perform a set of operations on multiple documents that match specified criteria
- DEFINE_PRE_CREATE – invoked before a new instance of a document is added to the database
- DEFINE_POST_CREATE - invoked after a new instance of a document is added to the database
- DEFINE_PRE_MODIFY - invoked before a new instance of a document is executed on the database
- DEFINE_POST_MODIFY - invoked when a new instance of a document is executed on the database
- DEFINE_INIT – used by the system to perform initialization tasks when the X-Service is deployed
- DEFINE_LISTENER – used to setup listeners for events

X-Operations

X-Operations are action statements that are used to write the business logic inside a workflow. A workflow consists of ‘nodes’ that perform specified functions, such as; Start, Task, Event, Branch, UI, Publish, etc. Within each node, X-Operations are used to execute specific tasks. The Logic of a Business rule (or X-Rule) is written using X-Operations. X-Operations are classified as follows:

- X-Document Operations
- SQL Operations
- XML Manipulations Operations
- Logic Operations
- User-Interface Operations
- Utility Operations
- Framework Services Operations

For more details, refer *Services Reference* section in *MDM Platform Reference Guide.pdf*.

X-Path

The primary purpose of X-Path is to provide a means to address parts of an xml document. X-Path makes available a number of functions for manipulating strings, numbers, and Booleans. X-Path is used as a concise way to express X-Rules. The primary syntactic paradigm in X-Path is the ‘expression’. An ‘expression’ is evaluated to yield an object, which has one of the following types:

- Node-set (an unordered collection of nodes without duplicates)
- Boolean (true or false)
- Number (a floating-point number)
- String (a sequence of characters)

Key Features of Studio

- Hierarchy Management
- Web Services Support
- Open Model Ingestion – OMI
- Data Authorization
- Data Quality Monitors
- MDM Test Services Framework
- Deployment Support
- Sample Application

Hierarchy Management

Hierarchy Management in Teradata MDM refers the definition, maintenance and viewing of user defined Hierarchies. In Teradata MDM Hierarchies are considered Master Data. The structure or relationships that define the structure are considered Master data as well as the actual members or nodes of the hierarchy.

Part of business critical master data consists of data that is hierarchical in nature. This data resides in a series of relational database tables that can be a part of the core master data. The data in these tables is hierarchical in nature, meaning that the data consists of parent-child relationships, generally represented through primary key – foreign key relationships in the underlying data. A common requirement for customers in a master data management context is the ability to manage and manipulate this hierarchical data, and central to this is the necessity to be able to view this data in a visual hierarchy.

The Hierarchy Manager is responsible for allowing users to define multiple hierarchies that in turn can be projected on the data, and used to visually present the data to users. Specifically, it allows users to define multiple Hierarchies that can then be used to visualize, manage, and manipulate their Hierarchy data that resides under the control of our MDM Framework. Furthermore, the Hierarchy Manager captures and stores these definitions as metadata that

allows for generalized rules and processes to be applied to the Hierarchy or to elements of the Hierarchy.

The Hierarchy Manager is integrated directly into the Master Data Manager Framework. It allows users to define Dimensions, a Hierarchy, Hierarchy Objects, and Hierarchy Relationships. Furthermore, it provides a direct linkage of the Hierarchy and its components to the data that is managed as part of the MDM workflows and processes.

Web Services Support

Web Services can be created from within MDM Studio. WSDL (Web Services Description Language) is an xml based language for describing web services and how to access them. It specifies the location of the service and the operations (or methods) the service exposes. MDM Studio provides the capability of exposing any X-Service API as a web service, by simply defining its interface in the WSDL description for that X-Service.

In addition, secured Web Services can be configured and generated using MDM Studio Plugins for Eclipse. [Figure 31](#) and [Figure 32](#) displays the MDM Web Services support for Incoming and Outgoing services respectively. For detailed description on incoming and outgoing services, *refer to Chapter Web Services Implementation in MDM Platform Studio User Guide*.

Figure 31: Web Services Support—Incoming Web Service

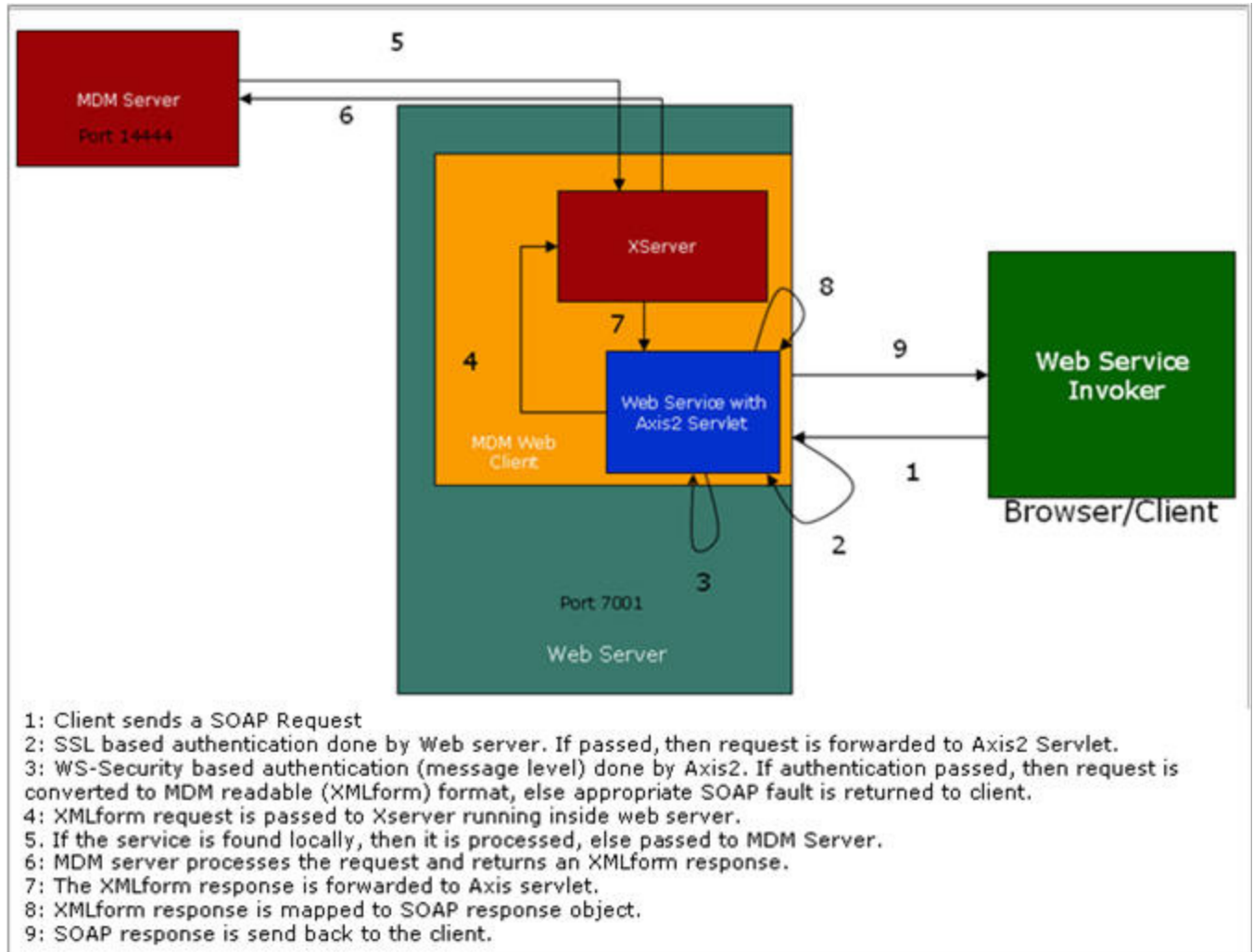
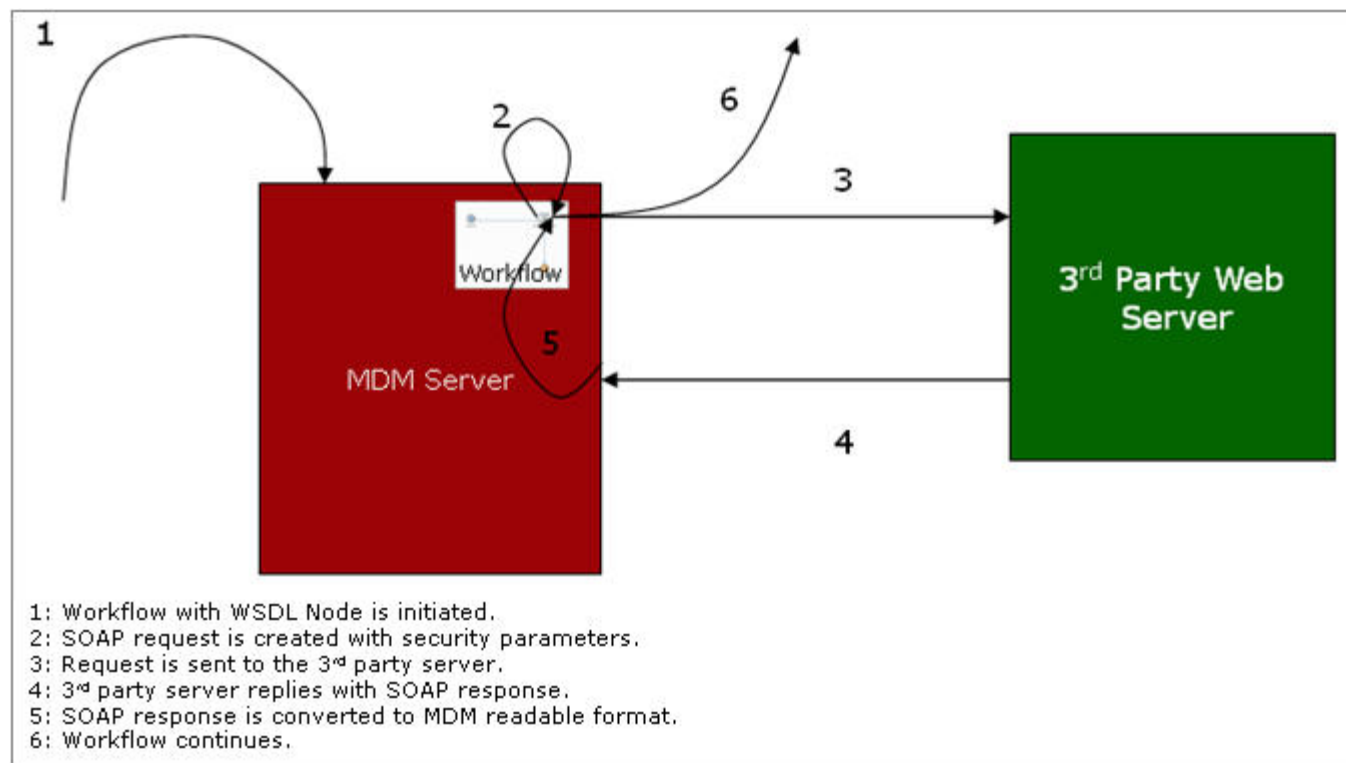


Figure 32: Web Services Support—Outgoing Web Service



OMI

Open Model Ingestion (OMI) allows externally defined data models to be imported into MDM Studio. The Data Model is the foundation of the MDM framework, upon which all workflows, business processes, validation rules and data quality tests are based. Models defined externally can be quickly imported into Studio, saving hours of time to manually recreate the model in Studio. Models can be imported from the following external sources:

- Erwin 4.1

Data Authorization

MDM framework creates a master copy of all of the data and then subsequently manages the access to this data using the data authorization feature. It incorporates the capability of having Row Level Security (RLS) and Column Level Security (CLS) on tables. This integration ensures that each user will have access to only data and data attributes to which they are authorized. This feature offers a significant advantage as data under the control of the MDM Framework is now secured.

Using MDM UI, you can assign roles for the columns and rows of different tables. Only users with proper authorization can view the rows and columns of the authorization enabled tables.

Data Quality Monitors

The Data Quality Monitors provide information about the quality of the data that resides in the Master tables. Data in these tables may contain 'soft' errors. 'Soft' errors are errors that

are not serious enough to prevent the data from updating the master record, but are still noteworthy, and should be corrected in the future. The Data Quality charts provide the following information:

- Chart showing the ratio of records containing ‘soft’ errors to ‘clean’ records (records having no errors)
- An analysis that can be used to show the number of records in a master table that have specific types of errors
- A breakdown of the number of rows having 0, 1, or more ‘soft’ errors

MDM Test Services Framework

The MDM test service provides the framework to execute test requests on services deployed on a specified MDM server. The request and the associated response from the server can then be used to verify the result of the test. The results of the test are then written out to a file.

The service exploits the capability of Teradata MDM’s workflow engine. Tests run as part of the test service, are essentially workflows. A test workflow can be viewed as a series of requests to the server, each request dependent on the previous one. Workflow acts as the glue that ties these requests through variables and provides for the flow of actions. Hence, every test workflow corresponds to testing a business workflow on the server through a series of API calls.

Example: Customer order creation and completion test would involve:

- 1 Creation of customer order
- 2 Acknowledgement of the resultant activity orders
- 3 Shipment notification
- 4 Invoice notification
- 5 Delivery notification

The goals set for this service are:

- 1 Reduce the size of test workflows required to test a business functionality.
- 2 Provide a framework to verify the results of a test.
- 3 Provide a framework for reporting the results of tests.
- 4 Enable a high degree of reusability of test workflow code.
- 5 Provide a mechanism to instantiate test request data with as little manual intervention as possible.

Provide a common test infrastructure to execute unit tests, regression tests and PSR tests

Deployment Support

The Teradata MDM Deployment Manager provides the ability to create a runtime deployment package from the MDM Studio application project and move or deploy the code (including all appropriate xml files) to the ‘target environment. The target environment may be a Quality

Test system, a Pre-Production system, a Production system, or whatever target the implementation calls for.

MDM Deployment Manager addresses the deployment of the MDM application from MDM Studio to other 'servers'. It does not address the installation of MDM Studio, Server, or Database. The Teradata MDM Installation Guides provide installation instructions for MDM Studio and MDM Server. Within the reference guides the database definition via Schema Generation is provided.

MDM Deployment Manager consists of two basic functions:

- A function within MDM Studio to package directories and files created in Studio for a MDM application. This 'package' is converted into a CLOB and then inserted into the MDM Deployment database and table(s).
- An InstallAnywhere executable that retrieves the MDM application 'package' from the MDM Deployment database and 'deploys' it to an existing MDM installation on another system. The InstallAnywhere provides a GUI for ease-of-use installation on Windows, AIX, or Linux in Web Logic or WebSphere, appropriately.

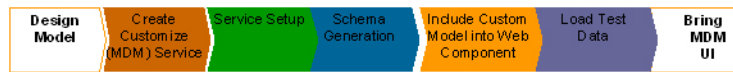
Sample Application Support

The TD MDM solution provides the capability to manage any kind of System of Reference (Master) data or System of Record data. Based on that, it has been used to solve a variety of business problems related to information management. We have chosen Customer Data Integration (CDI) as the Business Problem to be solved with a MDM Application. The primary purpose of the Sample Application is to provide real-world examples of key features and functions of the Teradata MDM product. The various features and functions are exemplified in the context of a common business problem regarding Customer Information.

In this Sample Application (CDI) explains about the Customer consolidation, identification of the duplicates and makes a new unique or update of existing records based on the selection. The CDI Process is modeled here. It starts with a Dashboard view of all Customer related information. From the dashboard, the user can deal with duplicate resolution (merging/survivorship), drill into customer details, and get into a customer enrichment process. For survivorship, the user can study the incoming record and compare it against the existing MASTER record. The user can then interactively select one or more fields from one record and one or more the other record to form the winning record that will become the MASTER record. And for Search will show all active available customers and drill into customer details for view or Enrich. New Customer is used for introducing New Customer and become active this new created customer has go to some approval process and finally this new customer active and available for further activities. For enriching the customer details is based on the role/user group and concern eligible/ authorized person will modify/add the details to a particular selected customer. In the MDM UI activities are visible to user based on the user group/role.

Before the CDI process, It explains about the Sample Application Model design and related MDM services, workflows and rules which are created using the Teradata Studio.

Figure 33: Sample Application Creation process



See [“Section B—Sample Application”](#) for more details.

List of Key Services

- Timer Service
- Publication Service
- Incremental Schema Generation

Timer Service

The MDM Timer Service provides the capability to perform specified actions at a specified time. In addition, the duration at which the timer should repeat actions can be specified. The Timer Service operates in two phases:

- Phase I provides for the setup of the timer in the system with a specified callback time. This activity is handled by the MDM server and it also manages adding timer records to the database.
- Phase II allows the Timer Sink to run as a service on the MDM server and manages the specified callback actions on the services.

Publication Service

Publication Services is a new feature in MDM 2.0 that allows the Data Steward to control the usage and flow of Master Data to end users or to consuming processes or applications. Publication Services allows any data within the scope of MDM to be published in multiple formats, to meet the needs and requirements of consuming application or processes. Consuming applications or processes can retrieve data directly from the Publication database, or they can have data pushed to them from a JMS Provider Queue table via JMS Messaging. Additionally, data can be extracted from the database into file format, and emailed directly to business users. Along the way, an audit trail of each publication request is preserved, including (optionally) the actual data published with each request.

Publication Objects

A Publication Object is defined as any collection of information that can be published to a downstream application, process, or end user. A Publication Object is comprised of a Publication Key, and the Publication MetaData. In the context of publishing Master Data, the Publication Meta Data specifies the composition of the data (tables and columns) that will be published to the downstream consuming application, process, or user, and is represented by one or more XDocuments, and their respective properties. XDocuments and Properties are used by Teradata MDM processes to denote underlying Tables and Columns respectively. The Publication Key is used to create a singular reference to this collection of data.

Publication Methods

The Publication method is used in a Publication Workflow Node the method (or format) that the data will take when it is published in a workflow. Data can be published to the following formats:

Internal Formats – data remains within Teradata

- Database Tables (based on the X-Documents being published) – the data will be published into user-specified tables, ready for consumption by downstream applications and/or processes.
- JMS Provider Queues – this is a specially formatted Teradata Queue table. The data is converted into XML-based JSM Payloads, and can then be pushed to J2ee applications via the Teradata JMS Provider.

External Formats – data is exported from Teradata

- Excel – data can be published into Excel 2003 compliant worksheets
- Comma Separated Value (CSV) text files – data can be published into one or more text files

Publication Workflow Nodes

The Publication Node is a new MDM Workflow node that has been added in MDM 2.0 to support Publication Services. Publication Workflow Nodes can be added to any workflow in which data should be pushed to a consuming process or application, or directly to a business user. The Publication Workflow Node is used to specify the data being published (via the Publication Object), and the method (or format) the published data will take.

CHAPTER 4 Customer Service

What's In This Chapter

This chapter provides information about Teradata customer service.

Topics include:

- [World Class Support](#)
- [Web Access](#)

World Class Support

Teradata provides World Class Customer Support in which the contracted service level is documented in the “Teradata Master Data Manager Support Plan”. This document can be obtained from the Teradata Customer Service Consultant.

The Teradata Service Focus Team is a user group of Teradata customers which drive the support processes and procedures in order to achieve the highest customer satisfaction. Every year the Teradata Global Support Center achieves high scores in customer satisfaction which is evaluated from numerous customer feedback mechanisms. Every year the Teradata Global Support Center also rates very high in the “Service Capability & Performance (SCP) Support Standard”.

Here is a list of the core features of MDM support. Additional service level options are available.

- Remote Problem resolution assistance to correct Software Problems
- Local business coverage Monday through Friday 8AM to 5PM Pacific Time.
- 7x24 unlimited phone and Web access to report software problems and to search for known problem resolutions
- Customer assigned call priority as predefined
- Timely resolution for reported problems
- Three Customer Authorized Service Requesters
- Support on the current distributed Major/Minor release and the prior Major or Minor release
- Maintenance on the latest Maintenance Release of the current Major/Minor release.

The Support Plan also includes “Call Priority Definitions”, “Technical Escalation Guidelines” and “Support Team Contacts” (including the management team).

Web Access

Web based access is available through the Teradata@YourService Website. Below are the key benefits of Teradata@YourService:

- **Search:** Knowledge repositories to quickly get answers to questions or problems. Several knowledge repositories are searched with one query, including the online documentation. This search engine can also be used as a great training tool.
- **Software & Patches Access:** Download selected Teradata drivers, software and patches to fix known problems
- **Collaborate:** 24 x 7 with thousands of Teradata users around the world for an answer, post a question or answer other users' questions in the Teradata discussion forum community
- **Incident Management:** Create, View and update incidents online. This allows the customer to define the incident exactly as they want. Teradata@YourService provides the ability to upload attachments, view updates and link to other documents referenced in the incident.
- **Reporting:** Reports are available on Teradata@YourService which reflect how the "Service Level" and "Resolution Within Guidelines" are being met.
- **ERs/RFCs** – Requests for new MDM functionality can be opened through Teradata@YourService and will be routed directly to Product Management.

Teradata@YourService can be accessed under www.teradata.com under "Support Services".

CHAPTER 5 Training

What's In This Chapter

This chapter provides information about Teradata training.

Topics include:

- [Training Information](#)

Training Information

Teradata Training provides a full curriculum of MDM classes to meet the needs of all audiences. The individual courses in Teradata Training's MDM curriculum include:

- Teradata MDM Solution Overview (Teradata University Course # 38651)
 - 2-Hour Web-Based Training
 - This course covers the main concepts and business benefits of Master Data Manager. It also reviews the key components of the Teradata MDM solution, the current MDM marketplace, and the competitive landscape.
 - Audience: Teradata Internal Only
- Teradata MDM Application Workshop (Teradata University Course # 39360)
 - 3-Day Instructor Led Training
 - This course provides detailed, hands-on training to accompany the base Teradata MDM application. Its purpose is to make the students proficient with the concept, purpose, and operation of the Teradata MDM solution.
 - Audience: Teradata Solution Architects and Customer Data Stewards
- Teradata MDM Developers Workshop (Teradata University Course # 39361)
 - 3-Day Instructor Led Training
 - This course provides detailed, hands-on training to cover the Teradata MDM Studio application development environment. Its purpose is to make the students proficient in the development of MDM applications.
 - Audience: MDM Application Developers (Teradata PS & Customers)

For additional information on Teradata MDM training you can access the Teradata University Web site or contact your Teradata Sales Associate.

Please note that if third party partner software or other Teradata software is to be utilized (such as Trillium, SAP, FirstLogic, D&B, or Teradata Warehouse Miner) training for those products must be attained through each of those products training offerings.

SECTION B —Sample Application

Section B provides link to the various chapters on Sample Application.

- [Chapter 6: “Introduction.”](#)
- [Chapter 7: “Custom Application.”](#)
- [Chapter 8: “Custom Models.”](#)
- [Chapter 9: “Define Web Component.”](#)
- [Chapter 10: ”Sample Application Process”](#)
- [Chapter 11: ”MDM Features”](#)
- [Appendix B: “Configuration of Trillium Client.”](#)
- [Appendix C: “MDM Custom Web Services.”](#)

CHAPTER 6 Introduction

What's In This Chapter

This chapter provides information about MDM and its key features and MDM sample application.

Topics include:

- [Introduction](#)
- [MDM Key Features](#)
- [Studio Sample Projects](#)
- [MDM—Sample Application \(CDI\) Solution](#)
- [MDM—Sample Application \(CDI\) Solution with Trillium Software](#)

Introduction

The important principle of Master Data Manager is to manage the enterprise resource (data) from an enterprise perspective. Teradata MDM is a class of enterprise applications designed to support the management of data and metadata across heterogeneous enterprise applications. Teradata MDM is built on a model driven, open standards, service-oriented architecture and enables the enterprise to manage master data via a virtual data dictionary using easy to use, visual, business process and data modeling tools. Business and data workflows are defined and executed using a powerful Business Process Execution engine (BPE) and data validation framework. These tools enable business users to create, update, and document processes and workflows while simultaneously empowering IT departments to rapidly extend existing data models to support the changing business needs.

Teradata MDM, at its core, consists of an enterprise data solution supporting the following business and IT problems:

- 1 Data Management
 - a Define, create, update, and delete master data (soft delete, that is data would not be visible to user but will be available in the system).
 - b Maintain enterprise metadata.
 - c Define business process and data validation rules.
 - d Provide mass update capability to systems.
 - e Track changes in data for audit purposes.
- 2 Data Synchronization

- a Define workflows for data flow across systems.
 - b Store intermediate data from multiple systems.
 - c Validate data against business processes and rules.
 - d Provide clean data to all systems.
 - e Aggregate data for asset management.
 - 3 Data Quality Measurement
 - a Create exception and informational alerts.
 - 4 Design environment for maintaining business processes
 - a Visually create workflows, data models and business rules that define a business process.
 - b Easily update and document the business processes.
 - 5 Rapidly create/update UI screens and workflows for Infrastructure Services
 - a Enable role-based security for workflows and data for users and user groups.
 - b Integrate easily with EAI and ETL tools.
 - c Provide real time alerts for exception handling.
 - d Enable authentication by third-party systems such as Directory Services and so on.
- MDM solutions are applicable in the following industries:
- i Manufacturing and Hi-Tech
 - ii Consumer Goods and Retail
 - iii Telecommunications
 - iv Financial Services
 - v Insurance
 - vi Pharmaceuticals and Life Sciences
 - vii Healthcare

MDM Key Features

- 1 Table Editors
- 2 Manage Lookup Data
- 3 Hierarchy Management
- 4 Publishing
- 5 Data Load
- 6 User Management
- 7 Managing Master Data

- 8 Business Rule Builder
- 9 Data Quality
- 10 Manage Authorization
- 11 Manage Query
- 12 Manage Logger
- 13 Web Services
- 14 3rd party Application Integration - Excel
- 15 Alert

Studio Sample Projects

- 1 PGL Examples
- 2 Project Teachers
- 3 Workflow Monitor
- 4 Approvals

MDM is a Web based application and is accessible through the Web browser. Contact your system administrator for the appropriate link to the MDM login page.

MDM Sample Application

The primary purpose of the sample application is to provide real-world examples of key features and functions of the Teradata MDM product.

The Teradata MDM solution provides the capability to manage any kind of System of Reference (Master) data or System of Record data. Based on that, it has been used to solve a variety of business problems related to information management. For sample application demo process, the Customer Data Integration (CDI) is selected as the business problem to be solved using MDM Application. The various features and functions are exemplified in the context of a common business problem relating to customer information.

Custom Application

You can create a custom application on top of an existing MDM installation. Your custom application should be an MDM based application and should reference the MDM Toolkit models, services, workflows and rules, as well as contain its own service with a model, workflows and rules. Once your application has been setup, you can include your custom application related workflows, business rules and validations. For details, see [Chapter 7: “Custom Application”](#)

MDM—Sample Application (CDI) Solution

The CDI sample application provides solution for problems of customer consolidation, identification of duplicates and the addition of new unique records or update of existing records based on selection. For the overall CDI process flow, see [“Sample Application CDI Process Flow”](#).

MDM—Sample Application (CDI) Solution with Trillium Software

The primary purpose of the sample application with Trillium is to provide real-world examples of key features and functions of the Teradata MDM product, as well as external application usage support. In the sample application (CDI) with Trillium, Teradata MDM solves the problems of customer consolidation, identification of duplicates, and the addition of new unique records or update of existing records based on the selection. For the overall CDI solution with Trillium software, see [“Sample Application with Trillium Process Flow”](#).

CHAPTER 7 Custom Application

What's In This Chapter

This chapter provides details on creating custom application on top of existing MDM Platform.

Topics include:

- [Introduction](#)
- [Custom Application Creation](#)
- [Sample Application Setup](#)
- [Custom Application Folder Structure](#)

Introduction

After installing MDM base, you have two options for setting up the sample application.

- Create custom application ([Custom Application Creation](#)).
- Use the existing custom application (sample application) provided in your base MDM ([Sample Application Setup](#)).

Custom Application Creation

This section describes the creation of a custom application on top of an existing MDM installation. Your custom application must be a MDM based application and it should reference the MDM Toolkit models, services, workflows and rules, as well as contain its own services with a models, workflows and rules. The custom application should be created in the MDM default location with a folder structure as `<MDM_Install_Directory>/custom/<customApplication Name>`. This folder contains only the files and sub folders required for custom application and can later be used for schema generation, application, and other processes.

Note: Currently, the MDM sample application supports only one level down (custom application name) in the custom folder structure. For details on custom application's folder structure, see [“Custom Application Folder Structure”](#)

To create a custom application:

1 Create a custom application location

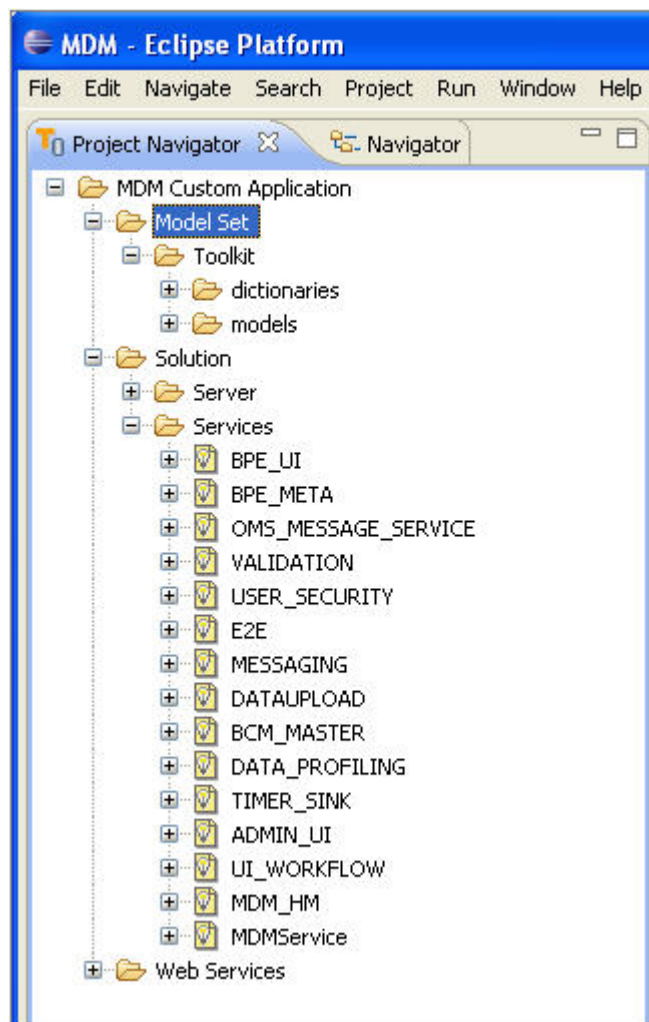
For detailed steps on creating new custom application location, refer to *section Creating Projects in Chapter 2 Getting Started of MDM Platform Studio User Guide.pdf*.

Note: For successful creation of a new project in MDM Studio with MDM Installation on WebSphere application server, the mdmclient war file should be extracted at `<MDM_Install_Directory>/web` folder, as it refers to the files from this directory.

2 Create models

The **Project Navigator** pane on the Studio displays the custom application as in [Figure 34](#).

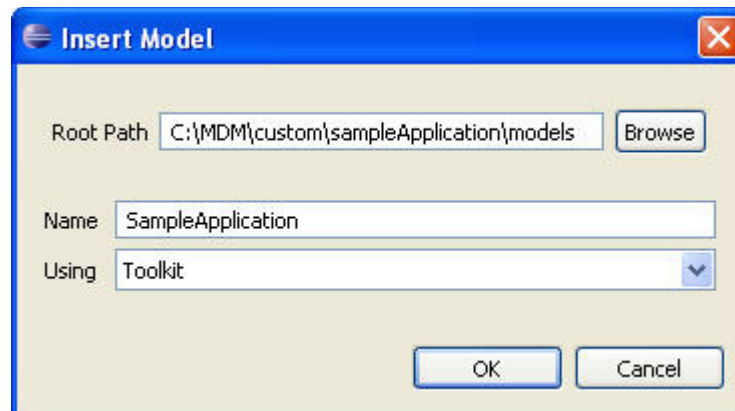
Figure 34: Custom Application—Studio Project Files



- e On the **Project Navigator** pane ([Figure 34](#)), right-click on **Model Set** and select **Insert Model** to create a custom model at the new custom location.

The **Insert Model** dialog box ([Figure 35](#)) is displayed.

Figure 35: Custom Application—Insert Model



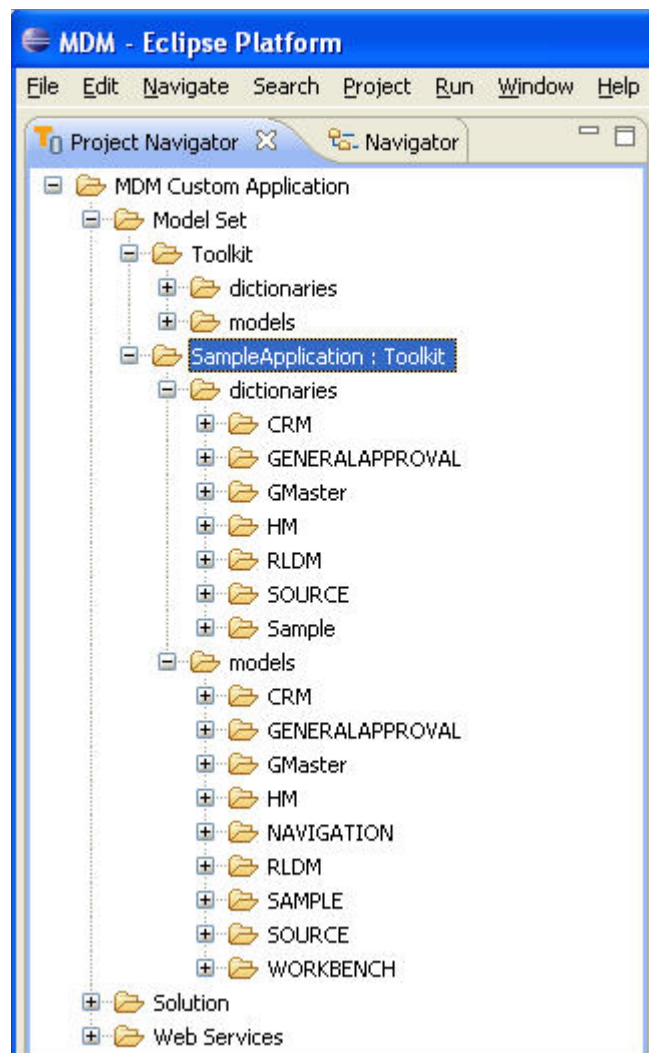
- f** On the **Insert Model** dialog box (Figure 35), enter the following:

 - Root Path: < MDM_Install_Directory>/custom/
<CUSTOM_APPLICATION_NAME>/models
 - Name: The name of the custom application
 - Using: Select “Toolkit” if the model’s base is the MDM toolkit, otherwise leave blank. It is recommended that you use “Toolkit” as the base model and click **Ok**. The SampleApplication model set is created.
- g** Create the dictionary and models for the custom application at the above location. You can create models using any of the following.

 - i** Import from XDOC
 - ii** Import from Erwin Model
 - iii** Import from Relational database
 - iv** Use Studio to create documents and dictionaries.

For details, see Chapter 8: “Custom Models”. The custom application models will be displayed as in Figure 36.

Figure 36: Custom Application—Models and Dictionaries



3 Create a custom application service

Execute the following steps to create custom application service:

- a On the **Project Navigator** pane (Figure 34), right-click on BCM_MASTER under Services and select **Customize Service**.

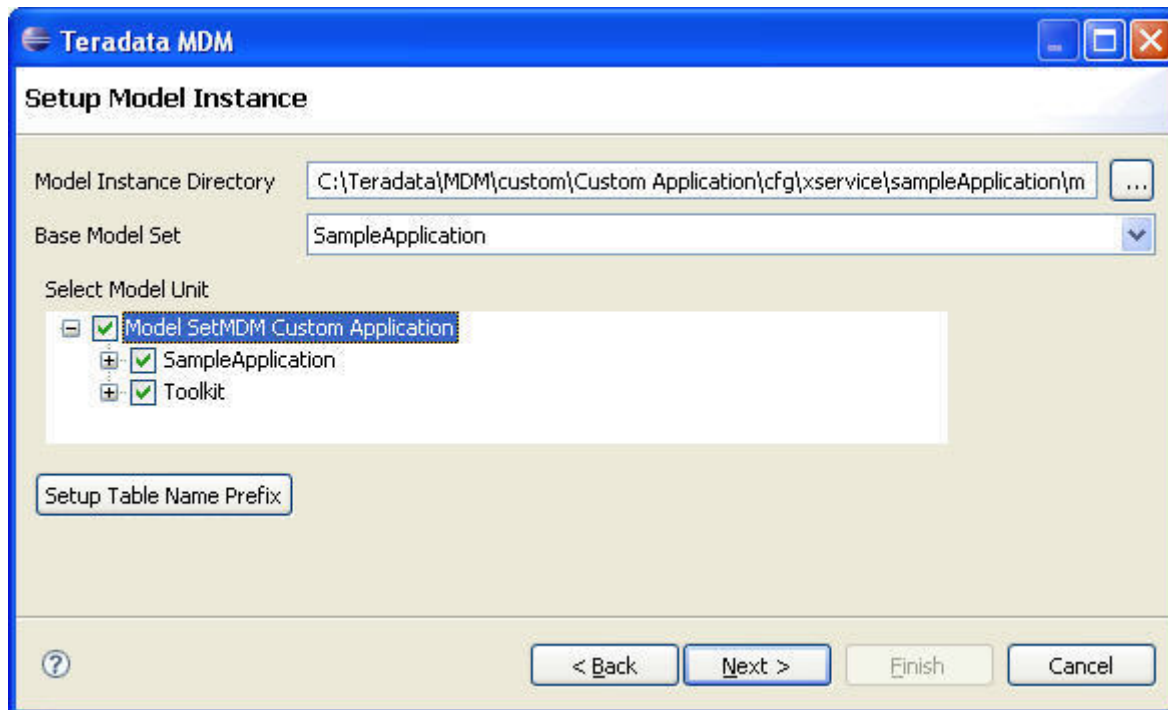
The **Setup Nature** page (Figure 37) is displayed. By default, the **MDM Based** option is selected.

Figure 37: Service Setup—Setup Nature

The screenshot shows the 'Teradata MDM' window titled 'Service Configuration Document'. The 'Setup Nature' section has a dropdown menu set to 'MDM Based'. The 'New Service File Information' section has three radio buttons: 'New Service' (selected), 'Existing Service', and 'Archived Service'. Below these are text fields for 'Service Name' (containing 'BCM_MASTER'), 'Service Path/Archive File Path' (containing 'C:\MDM\cfg\properties\custom_toolkit.xml'), and a 'Select Service' dropdown menu. At the bottom are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

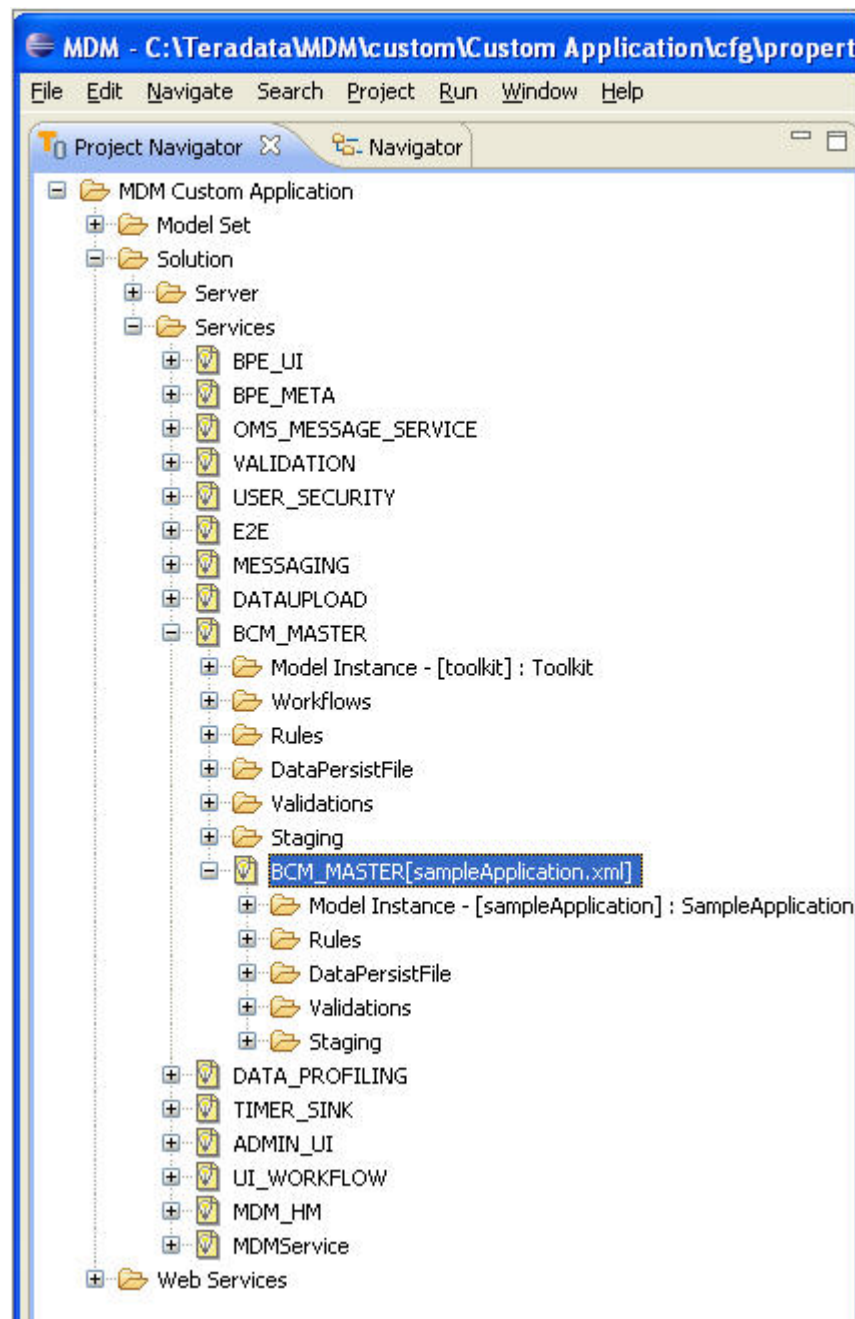
- b On the **Setup Nature** page (Figure 37), select **New Service** option and click **Next**.
 Toolkit.xml is the master service of the MDM base and its location must be `<MDM_Install_Directory>/custom/<CUSTOM-APPLICATION-NAME>/cfg/properties`. It is recommended to use a service file name that is the same as the custom application name. For example, if the custom application name is "sampleApplication" then the service file must be `sampleApplication.xml`.
 The **Setup Model Instance** page (Figure 38) is displayed.

Figure 38: Service Setup-Setup Model Instance



- c On the **Setup Model Instance** page (Figure 38), select the **Base Model Set** as SampleApplication and in the **Select Model Unit** pane, select the **Model Set** checkbox and click **Next**.
The **Setup Options** page will be displayed.
- d On the **Setup Options** page, click **Finish**.
The **Project Navigator** pane displays the custom application service as in Figure 39.

Figure 39: Custom Application Service



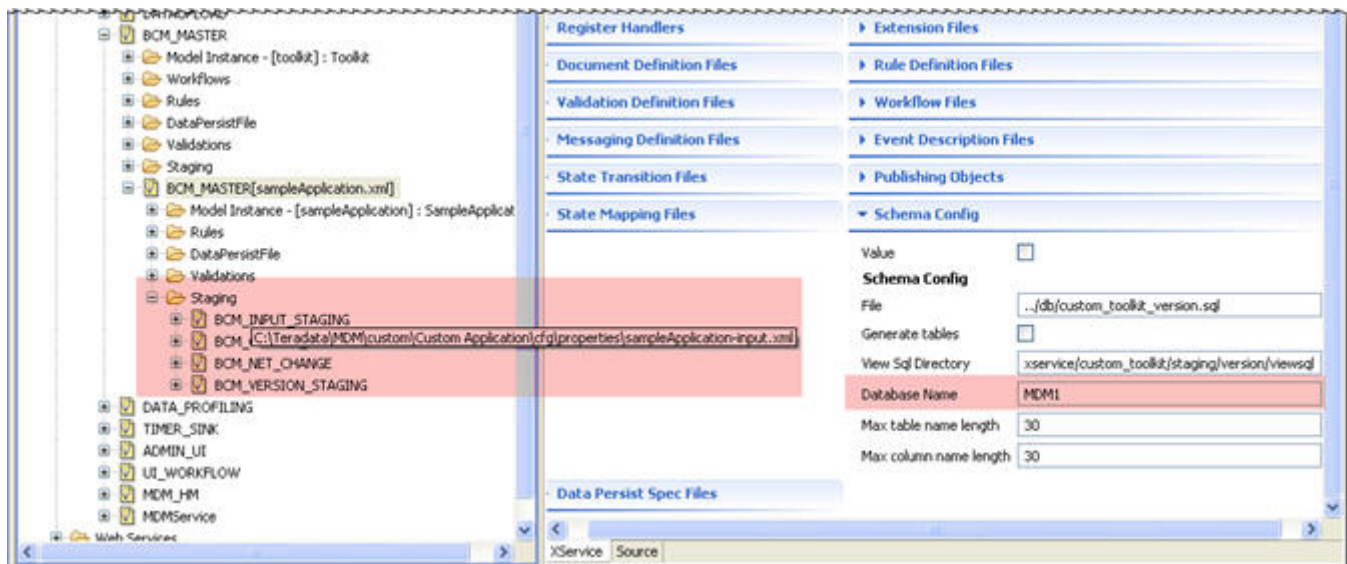
- e Add customer specific related workflows to the respective service file.

It is recommended that you add all workflows, rules and validations to the custom application service. For details on adding workflows, rules and validations, refer *Chapter 4: Process Modelling in MDM Platform Studio User Guide.pdf*.

Some manual changes are required for the custom service files. If the MDM base model uses the MDM database topology mechanism (separate databases for the services), then the custom application must be modified as in the below step g (otherwise leave these fields blank):

- f On the **Project Navigator** pane (Figure 39), double-click *sampleApplication.xml*. On the right pane, scroll down till **Schema Config** and in the **Database Name** field, enter the respective MDM staging database name.
- g On the **Project Navigator** pane, expand the *Staging* folder under *sampleApplication.xml* and repeat the step f for *sampleApplication-input.xml*, *sampleApplication-output.xml*, *sampleApplication-netchange.xml* and *sampleApplication-version.xml* as in Figure 40.

Figure 40: Enter Database Name in Schema Config



4 Modify the required files for the custom application schema generation

Execute the following steps to modify the required files for schema generation of the custom application.

- a If the custom service has any stored procedures or constraints to create then perform the following steps:
 - i Copy compile_MDM_SP.bat/.sh from <MDM_Install_Directory>/bin to custom location bin folder.
 - ii Open the copied file and change the customMDMSPList.txt name as per your requirement.
 - iii Create a file with the same name at <customApplication>/bin folder and add the list of stored procedures you require to compile at SG/ISG.

You can compile the stored procedures in the following two ways:

- Perform step i in the custom location as per your requirement.
Open <MDM_Install_Directory>/custom/<CUSTOM_APPLICATION_NAME>/bin/custom_gen_schema.bat/sh or incr_custom_gen_schema.bat/sh and add the following line:
echo compiling Custom Stored Procedure and Views

```
%JAVA_HOME%\bin\java %JAVA_OPTIONS%
com.teradata.xcore.util.StoredProcCompiler xserver.xml
<CUSTOMMDMSPLIST.TXT >
```

- Stored procedures gets compiled during Incremental schema generation or schema generation.

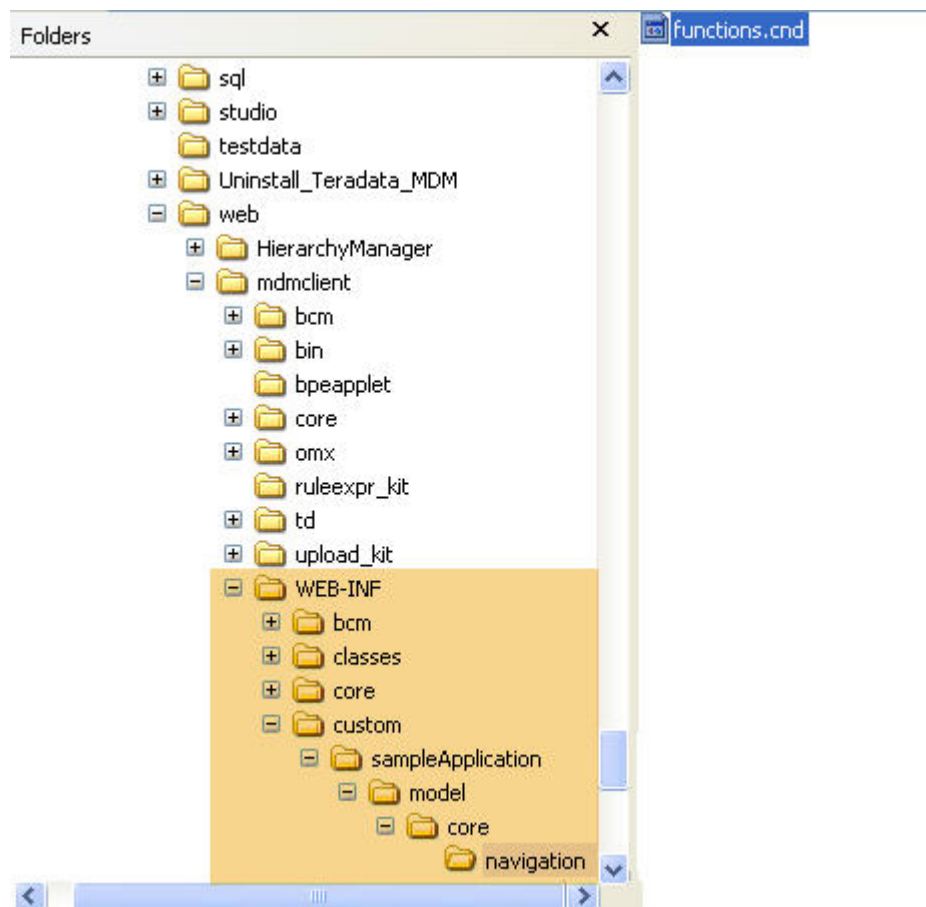
- b If the custom service has any IDGEN creation then include those entries as given below:

```
<idgen>
<document Name="SystemProperties" prefix="SYS" delimiter="-"
sequenceName="SYS_SEQ"
handler="com.teradata.xcore.idgen.BlockTableIdGenerator"
blockSize="100"/>
</idgen>
```

5 Create a Web UI component for the custom application

If the custom application has workflows and if the workflows are to be called from the MDM Web UI left navigation pane (UINavigation.xml pad-item structure), then you need to create a folder structure as in [Figure 41](#) at `<MDM_Install_Directory>/web/mdmclient/WEB-INF`.

Figure 41: Folder Structure



- a On the **Navigation** folder created, include a file *functions.cnd* with the below coding:

```
<?xml version="1.0" encoding="UTF-8"?>
<x2:commands xmlns:x2="http://www.td.com/x2" xmlns:bcm="http://
www.td.com/bcm" xmlns:core="http://www.td.com/core"
xmlns:xcore="http://www.td.com/xcore" xmlns:xcore-db="http://
www.td.com/xcore-db" xmlns:xapl="http://www.td.com/xapl"
xmlns:web="http://www.td.com/web" xmlns:docutil="http://
www.td.com/docutil" xmlns:util="http://www.td.com/util"
xmlns:FUNCTION="http://www.td.com/FUNCTION" xmlns:REMOVE="http://
www.td.com/REMOVE" xmlns:xudl="http://www.td.com/xudl"
xmlns:cfg="http://www.td.com/configurability" xmlns:perm="http://
www.td.com/permisibility" xmlns:identity="http://www.td.com/
identity" xmlns:il8n="http://www.td.com/il8n" description="_">
  <!--
  ****
  ** -->
  <!--
  ****
  ** -->

  <x2:command public="yes" description="_"
name="getPadItemExtension">
    <x2:param name="selectedActivity" select="$this/ACTIVITY/@Value"/
    >
      <xapl:print>
        ***** CALLING CDI PAD ITEMS.... *****
      </xapl:print>
      <xapl:return>
        <x2:execute
command="core.navigation.functions:getCDIStudioExtensions">
          <x2:with-param name="selectedActivity"
select="$selectedActivity"/>
        </x2:execute>

      </xapl:return>
    </x2:command>

    <x2:command public="yes" description="_"
name="getCDIStudioExtensions">
      <x2:param name="selectedActivity" select="$this/ACTIVITY/
@Value"/>
      <xapl:variable name="studioNavigation">
        <xcore:execute-method name="//BCMMasterService/
getNavigation" entireResponse="false">
          <user_id Value="{identity-user-id()}" />
          <user_name Value="{identity-username()}" />
        </xcore:execute-method>
      </xapl:variable>
      <xapl:return>
        <xapl:if test="count($pad/PAD_ITEM) >0">
          <xapl:value-of select="$pad"/>
        </xapl:if>
        <xapl:value-of select="$studioNavigation/*"/>
      </xapl:return>
    </x2:command>
  </x2:commands>
```

Note: The model folder would be created by default in the custom application folder. You can create the sub folders (core and navigation folders) as per your requirement, but the function.cnd should be in the model folder as in [Figure 41](#).

- b Based on the custom application requirements, you will need to change the above file.

- c Navigate to `<MDM_Install_Directory>/web/mdmclient/WEB-INF/classes` and open the `x2.properties` file and add the line `/WEB-INF/custom/<CUSTOM-APPLICATION-NAME>/model;` into the model path used by MDM UI as in [Figure 42](#).

Figure 42: x2.properties file

```
LOG.PRIORITY=WARN
workflow-id=system\ProcessRequest
model-attribute.compile-trace=off
model-attribute.validate-classes=on
model-attribute.validate-stop-on-error=off
model-attribute.modelpath=/WEB-INF/custom/sampleapplication/model:/WEB-INF/bcm/model;
/WEB-INF/core/model;/WEB-INF/system;/WEB-INF/upload_kit/model
#model-class=com.teradata.x2.model.xmlfile.XmlFileModel
model-class=com.teradata.x2.model.sax.XmlFileModel
model-attribute.load-trace=off
model-attribute.valid-extensions=xml;def;x2;cfg;cmp;cnd;wfl;pgl
CONTEXT.MODELPATH=/WEB-INF/system/context/xml/resolvers.xml
#LOGGER_CLASS=com.teradata.x2.auditlogging.StdOutLogger
LOGGER_CLASS=com.teradata.x2.util.logger.DynamicLogger
LOGGER_CONFIG_PATH=/WEB-INF/system/LoggerSettings.xml
LOGGER_DYNAMIC_CONFIG=on
```

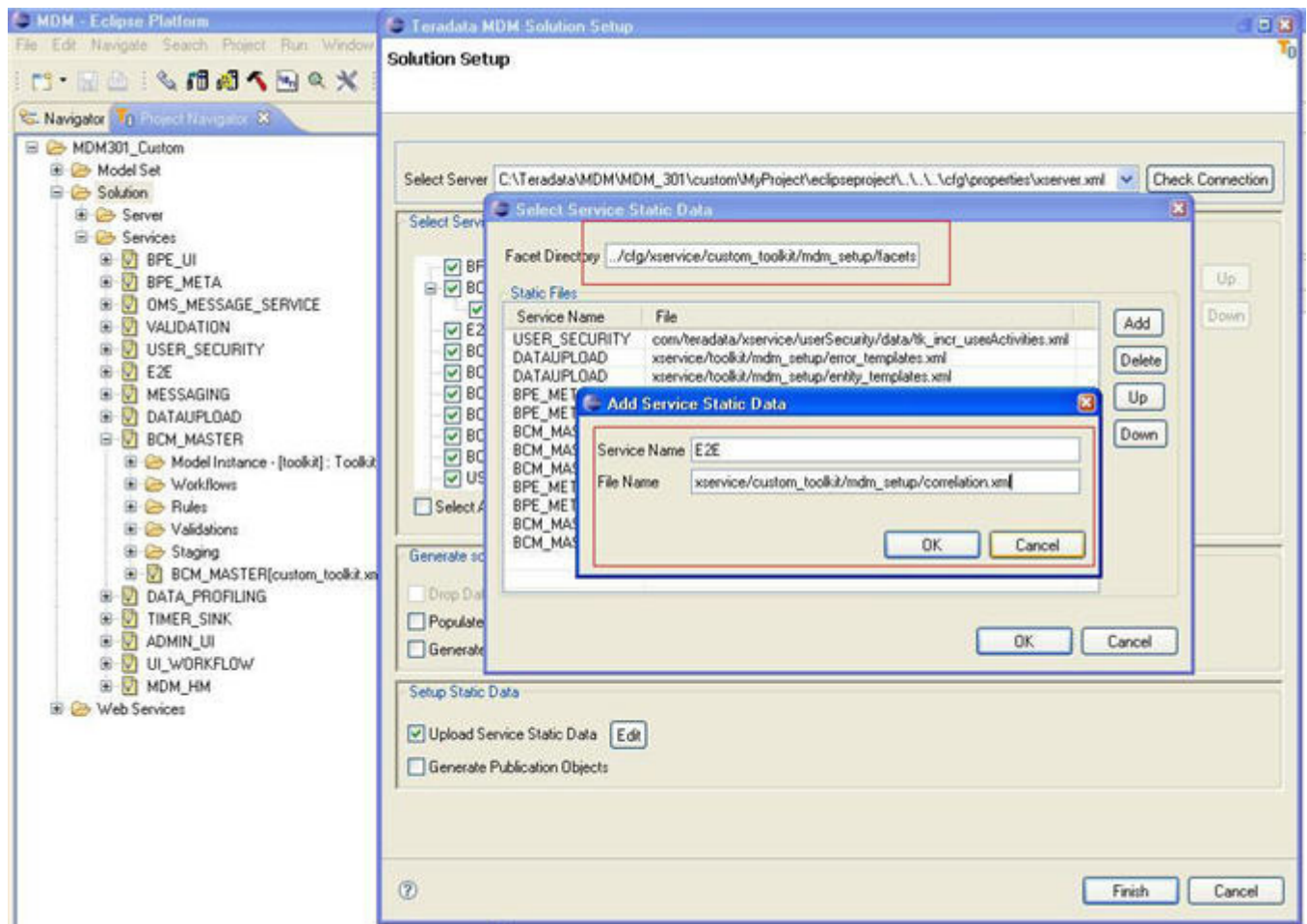
6 Execute the schema generation process

The schema generation process is performed by one of the two following ways: complete schema generation or incremental schema generation.

Note: During the installation of MDM, if schema generation was not performed, then choose the complete schema generation process else you can choose incremental schema generation (ISG).

- If running schema generation using solution setup from MDM Studio, perform the following:
 - Click on Edit button for Static Data and delete correlation.xml entry.
 - Add a new entry for custom correlation.xml as in [Figure 43](#).

Figure 43: Solution Setup



- Update facet path
- Perform Solution Setup.
- Complete schema generation.

The complete schema generation process generates the entire toolkit model, as well as the custom models. This method expects a clean MDM database (no MDM system or model tables already exist in the database). If you want to create a complete and full toolkit and custom model then run the schema generation file *call_custom_gendb.bat* (or *call_custom_gendb.sh* on UNIX) located at `<MDM_Install_Directory>/custom/<CUSTOM-APPLICATION-NAME>/bin`.

- Incremental schema generation

The incremental schema generation process generates SQL scripts based on the schema differences between an existing schema in a database and schema documents in one or more MDM Services. The inputs to this script are essentially the Server configuration file (that contains the database information whose schema forms the baseline for comparison) and the service configuration file(s) (that have references to the MDM XDocuments that will be compared with the baseline). If you only want to create the newly added model elements or modify existing elements, then run the schema generation file *call_custom_incr_gendb.bat* (or *call_custom_incr_gendb.sh*

on UNIX) location at `<MDM_Install_Directory>/custom/<CUSTOM-APPLICATION-NAME>/bin`. This will perform the incremental model creation for toolkit and the custom model.

The custom schema generation's log file `custom_gendb.log` is located at `<MDM_Install_Directory>/custom/<CUSTOM-APPLICATION-NAME>/log`.

Note: If you wish to deploy the application in a collapsed tier environment, refer *Appendix Collapsed Tier Setup in MDM Platform Studio User Guide.pdf*.

7 Load pre-defined data

If the custom application requires any pre-defined data or static data, you can create and load data to database at this point.

8 Bring up MDM UI

In order to bring up the MDM UI with the custom application, the application must be have been deployed into the database via schema generation (step 5). Use the batch files (located at `<MDM_Install_Directory>/custom/sampleApplication/bin`) created during the process of creating custom application location (step1).

To start the MDM server, run “startAll.bat”. This will start up the MDM locator and MDM server. Once both are up and running, start your Web server pointing it to the MDM default installation.

Sample Application Setup

Note: Read the README document available at `<MDM_Install_Directory>` before proceeding with the below steps.

Sample Application setup involves the following steps:

- Installation
- Load pre-defined data
- Configuring Sample Application Project in Eclipse
- Enable Web Service

After performing the above steps, launch sample application MDM UI and perform the “Load Data Process”.

Installation

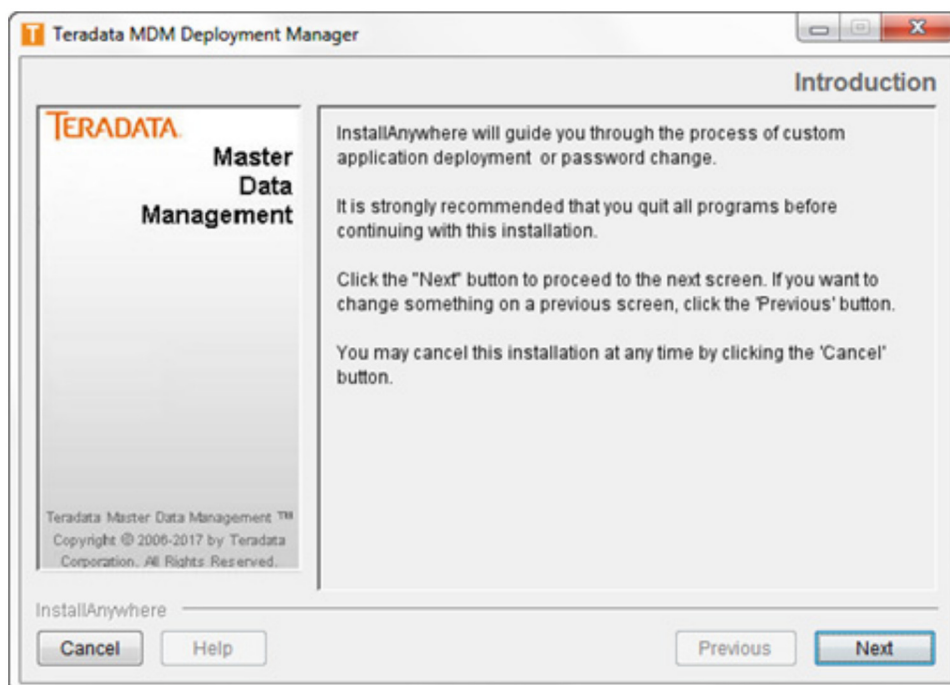
Pre requisite: install MDM before installing sample application.

Perform the following steps for MDM sample application setup:

- 1 Run DeploymentMgr.exe from `<MDM_Install_Directory>\bin`

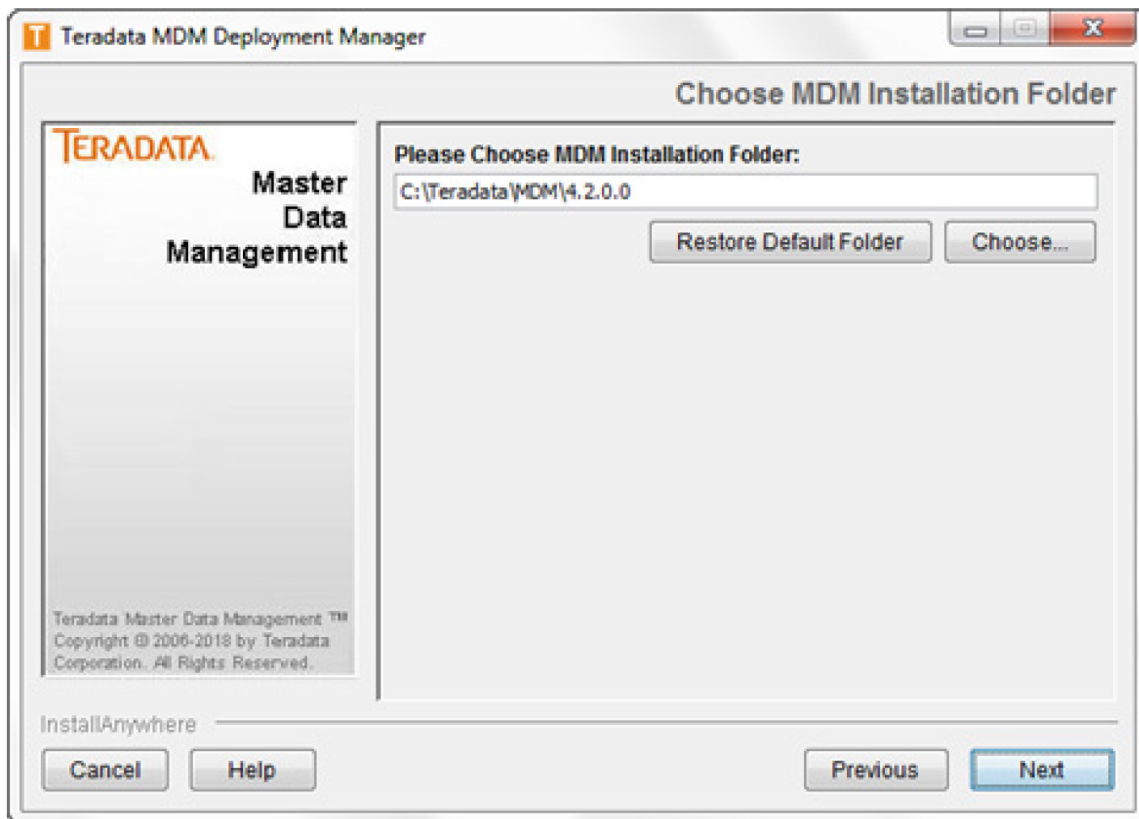
The **MDM Deployment Manager—Introduction** window (Figure 44) is displayed.

Figure 44: Introduction



- 2 On the **Introduction** window (Figure 44), read the information and click **Next**. The **Choose MDM Installation Folder** window (Figure 45) is displayed.

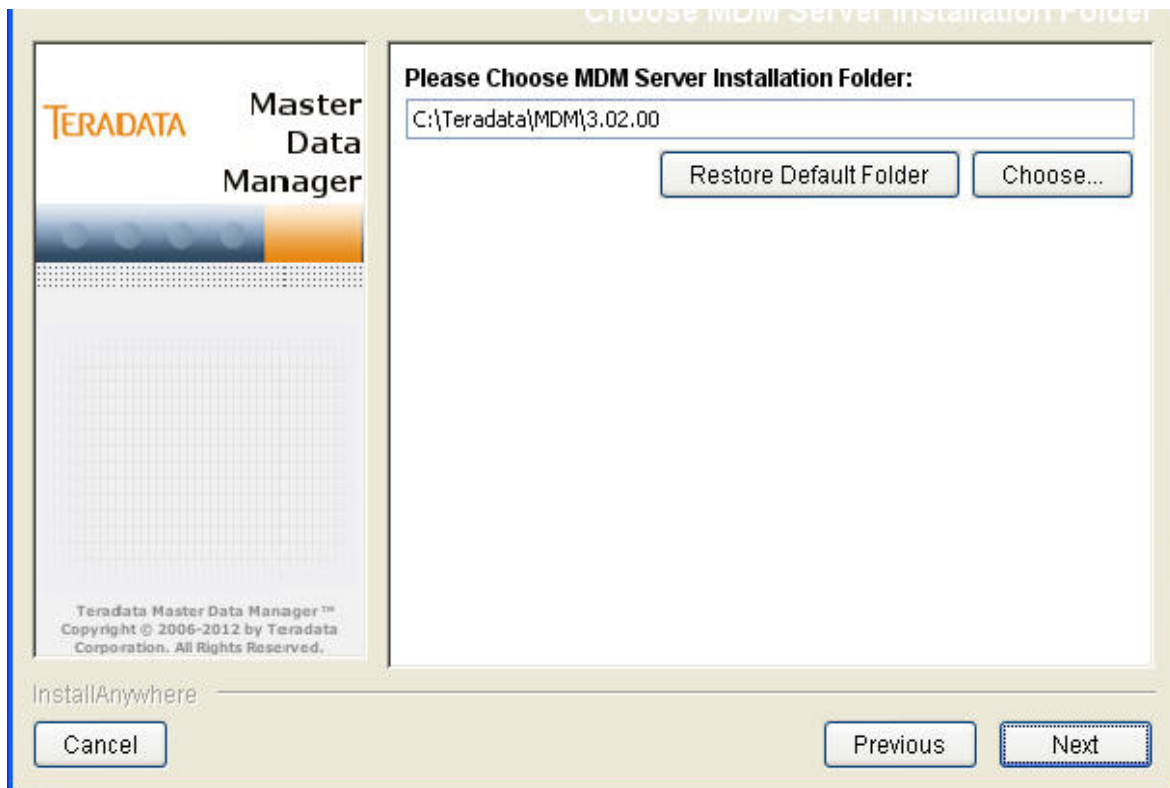
Figure 45: Choose MDM Installation Folder



- 3 On the **Choose MDM Server Installation Folder** window (Figure 45), select the location of the MDM installation and click **Next**.

The **Deploy Custom Application/Change Password** window (Figure 46) is displayed.

Figure 46: Deploy Custom Application/Change Password



- 4 On the **Deploy Custom Application/Change Password** window (Figure 46), select the option **Deploy Custom Application**.

For the **Source of Deployment** option, by default **Database** option is selected. If Database option is selected, provide the database details from where the deployment manager can fetch the stored custom project details. If File option is selected, specify the custom jar file location to be used for deployment.

Note: Select the deployment option as specified in Studio while creating the custom application.

- 5 On the **Deploy Custom Application/Change Password** window (Figure 46), with **Database** option selected as in Figure 46, click **Next**.

The **Database Setting** window (Figure 47) is displayed.

Figure 47: Database Settings

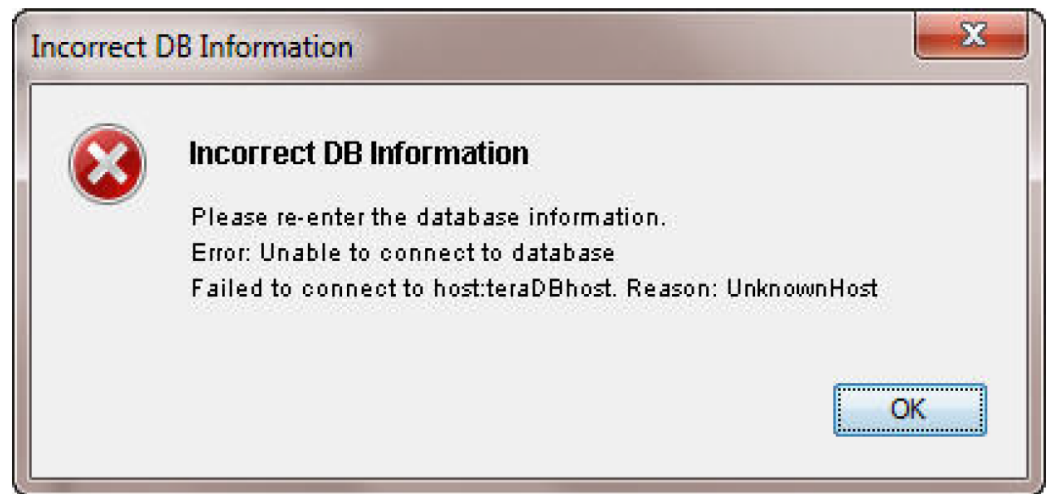
6 On the **Database Setting** window, enter the required details for the deployment database and click **Next**.

- Teradata Data System Name -> refers to the Teradata Database system hostname (mdmSystem/mdm etc.) where your custom projects exists (using MDM Studio).
- Database Name -> refers to the database (within MDM User) where your custom project exists (using MDM Studio). Deployment Manager refers to this database to fetch the list of archived/stored MDM custom Projects.
- Database User and Password -> refers to the database user and password where your MDM custom projects are saved.

Note: You can have your custom projects under different Databases (can be staging database as well). Deployment Manager refers to this database to fetch archived/stored (using MDM Studio) MDM custom Projects.

Note: If the entered details on the **Database Setting** window are not correct, the connection fails and the **Incorrect DB Information** window is displayed as in [Figure 48](#). Re-enter the details and try again to establish the connection.

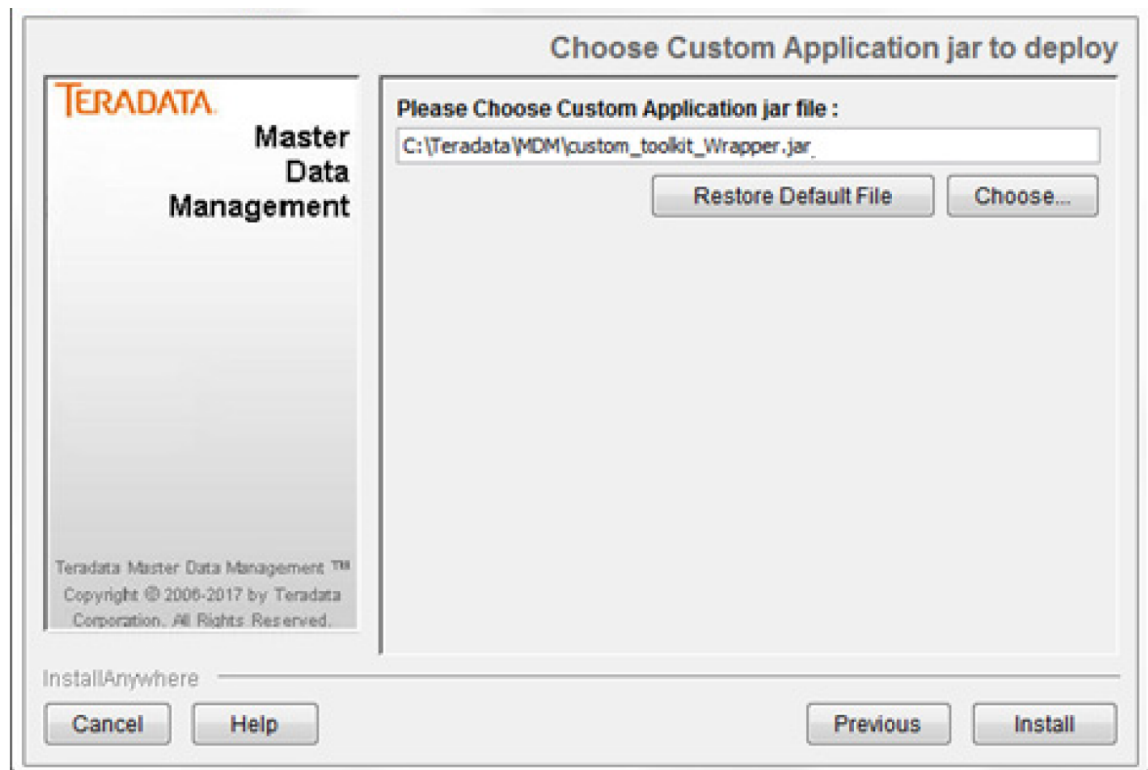
Figure 48: Incorrect DB Information



The **Enter Project ID & Your Name** window (Figure 50) is displayed.

On the **Deploy Custom Application/Change Password** window, if the **Source of Deployment** option is selected as **File System**, the **Choose Custom Application Jar to Deploy** window (Figure 49) is displayed..

Figure 49: Choose Custom Application jar to Deploy



- 7 On the **Choose Custom Application Jar to Deploy** window (Figure 49), specify the custom / sample application jar location and click **Next**.

Note: If any changes are done in .jar file, the file will be corrupted and cannot be re-used.

The **Enter Project ID & Your Name** window (Figure 50) is displayed. Displays the list of all the versions of all the projects that have been deployed in the Deployment database.

Figure 50: Enter Project ID & Your Name

Enter Project ID & Your Name

Enter Your Name:

Enter Project ID to Deploy from the below details:

ID	MODIFIED DATE	VERSION	PROJECT NAME
13	2017-12-04 16:54:10.0	1	CRDM
12	2017-10-31 11:37:49.0	1	CRDM
11	2016-07-15 12:46:06.0	1	CRDM
10	2016-03-23 08:51:12.0	1	mdmsample
9	2016-03-23 08:42:02.0	1	CRDM
8	2015-07-22 13:58:09.0	1	mdmsample
7	2015-07-22 13:52:42.0	1	CRDM
6	2015-05-22 15:14:25.0	1	mdm sample1
5	2015-02-17 15:40:27.0	1	mdmsample
4	2014-11-03 12:13:00.0	1	mdmsample
3	2014-11-03 12:11:00.0	1	CRDM
2	2014-11-03 10:00:28.0	1	mdmsample
1	2014-11-03 09:49:29.0	1	CRDM

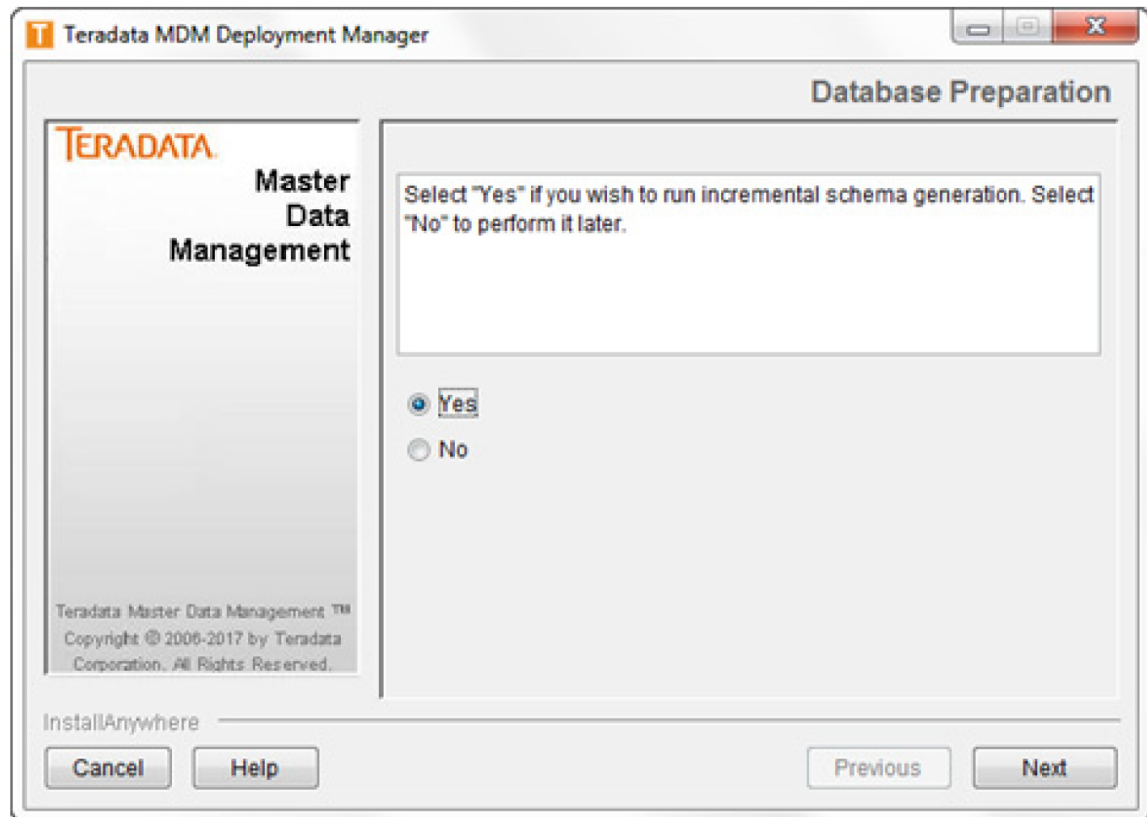
Cancel Help Previous Install

- On the **Enter Project ID & Your Name** window (Figure 50), enter your name and enter the ID of the project that needs to be deployed onto the target system and click **Next**.

The name is required for auditing purpose as this information will be logged under DEPLOY_LOG table to audit who deployed which application etc.

The **Database Preparation** window (Figure 51) is displayed.

Figure 51: Database Preparation



- 9 On the **Database Preparation** window (Figure 51), select the required option and click **Next**.

If **Yes** option is selected, schema generation changes will be done and if **No** option is selected, the schema generation changes will not be updated.

If Collapsed mode was selected during base MDM installation, by default collapsed mode is selected and the message is displayed as in Figure 52.

The **Please Choose Mode of Deployment** window (Figure 53) is displayed if Co-located mode was selected during base MDM installation.

Note:

- For re-deployment, use the same existing deployment mode. If the custom application is deployed in co-located mode, it can still be re-deployed in collapsed mode. But if the custom application is deployed in collapsed mode, it cannot be redeployed in co-located mode.
- For collapsed mode, the following jars have to be explicitly added to the application server (WebSphere/Weblogic/Tomcat) classpath in the same order.
 1. `<MDM_Install_Directory>/web/mdmclient/WEB-INF/lib/customcoloc.jar` (in case of custom/sample application, this has to be added first).
 2. `<MDM_Install_Directory>/web/mdmclient/WEB-INF/lib/coloc.jar`

Figure 52: Default Mode of Deployment

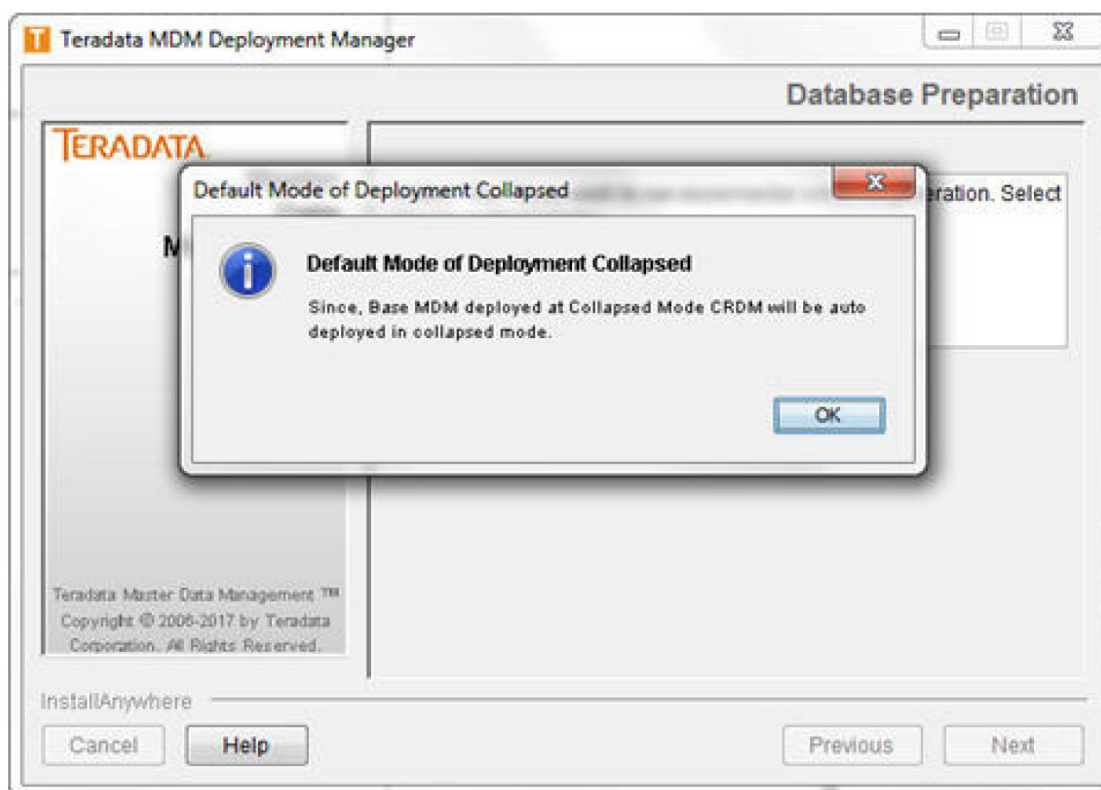
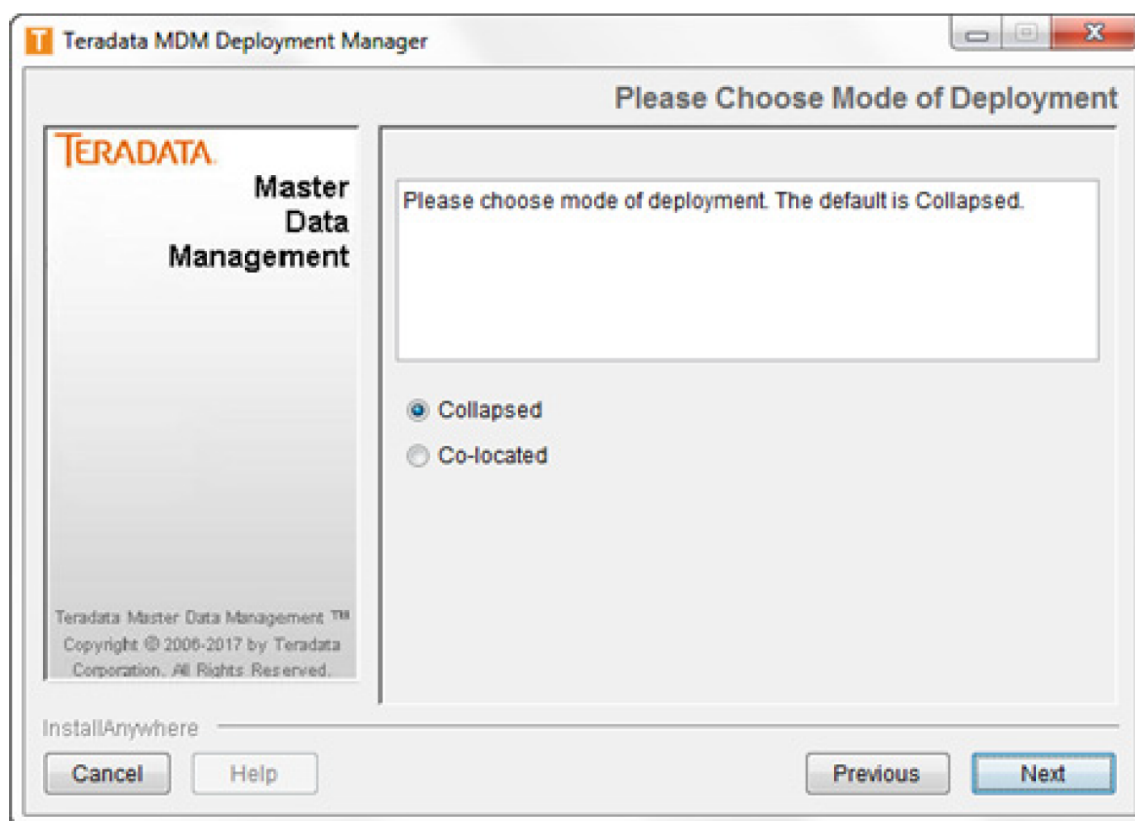


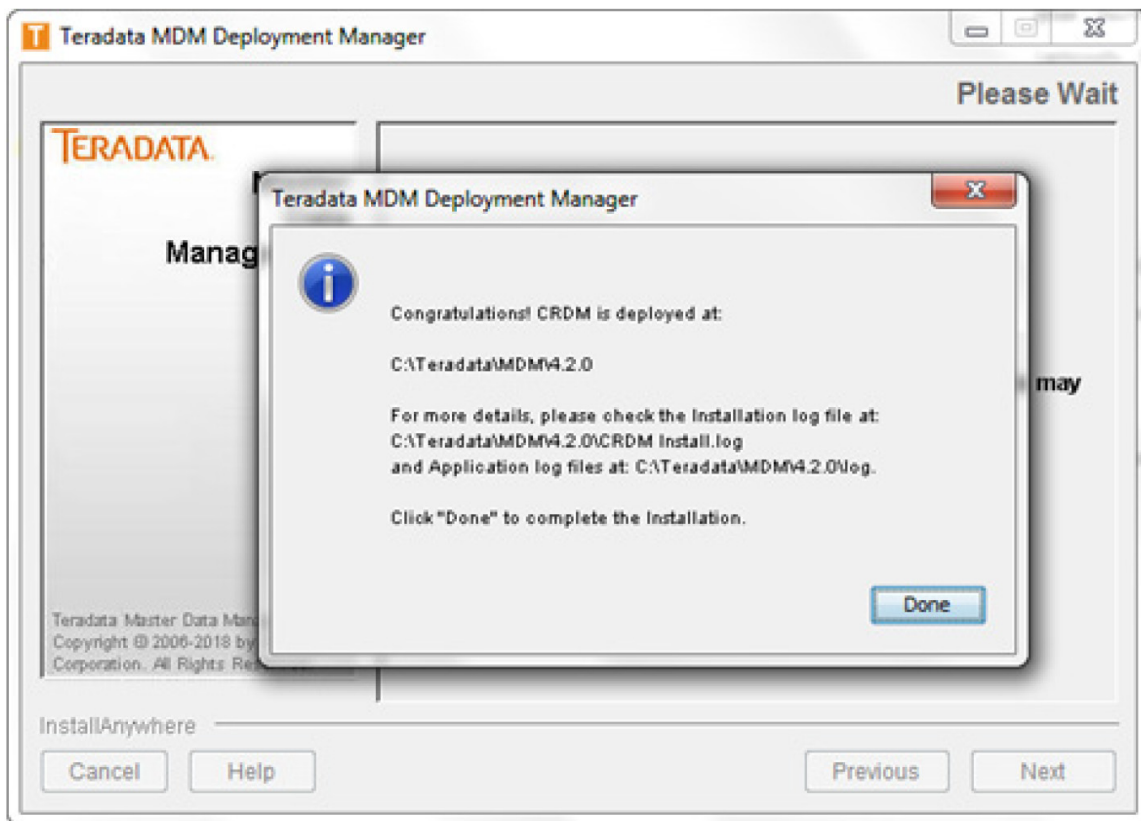
Figure 53: Please Choose Mode of Deployment



- 10 On the **Please Choose Mode of Deployment** window (Figure 53), select the appropriate deployment mode and click **Next**.

The **Install Complete** pop-up (Figure 54) is displayed.

Figure 54: Install Complete



- 11 On the **Install Complete** pop-up, click **Done**.

Load Pre-defined Data

To load pre-defined data, navigate to
`<MDM_Install_Directory>\custom\sampleApplication\testdata` and run
`Call_populate_sampleApplicationData.bat/sh`

On successful completion of the script, the predefined data are loaded into database.

Configuring Sample Application Project in Eclipse

Perform the following steps in MDM studio:

- 1 Launch Eclipse MDM studio. For detailed steps, refer to *section Opening an Custom Application Project of chapter 2 Getting Started in MDM Platform Studio User Guide.pdf*

Enable Web Services (Optional Step)

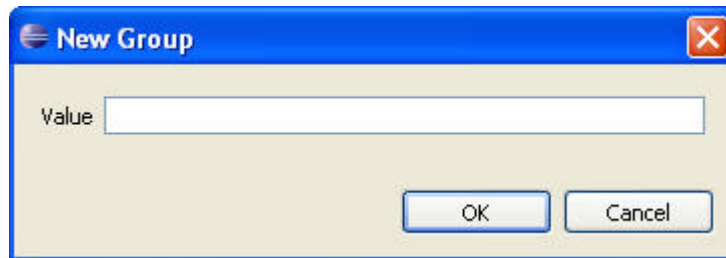
For detailed steps on how to enable web services, refer to *Chapter Web Services Implementation in in MDM Platform Studio User Guide.pdf*

Perform the following steps to enable custom web service:

- 1 Copy the BCM_MASTER.aar file from the location
<MDM_Install_Directory>\custom\sampleApplication\testdata\WSDL\customWebServices to the location<MDM_Install_directory>\web\mdmclient\WEB-INF\services
- 2 Launch MDM Studio. For detailed steps, refer to *chapter 2 Getting Started in MDM Platform Studio User Guide.pdf*.
- 3 On the **MDM Studio**, in the **Project Navigator** pane, right-click **Web Services** and select **Insert New Group**.

The **New Group** dialog box (Figure 55) is displayed.

Figure 55: New Group



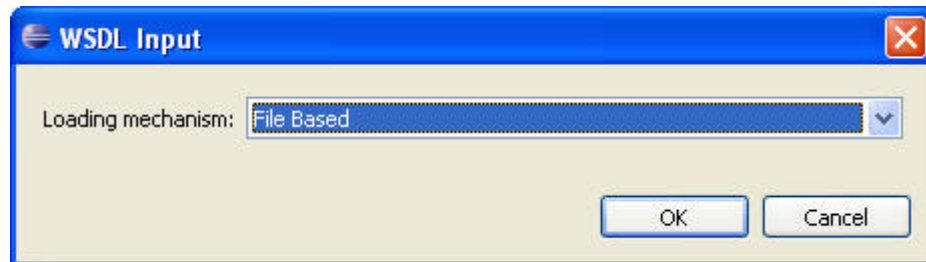
- 4 On the **New Group** dialog box (Figure 55), in the **Value** field, enter the group name and click **Ok**.

The group is added to the Web services node in the **Project Navigator** pane.

- 5 Right-click on the newly added group folder and select **Insert WSDL**.

The **WSDL Input** dialog box (Figure 56) is displayed.

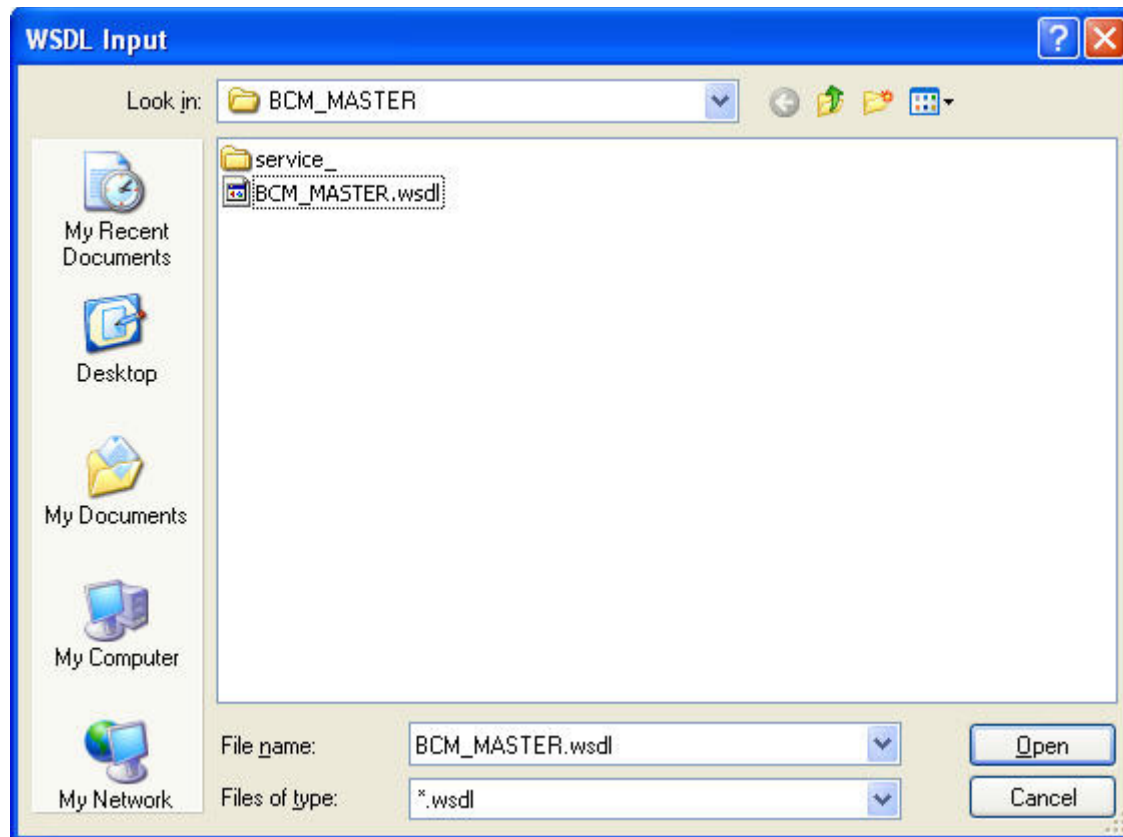
Figure 56: WSDL Input



- 6 On the **WSDL Input** dialog box (Figure 56), from the **Loading Mechanism** drop-down list, select the loading mechanism as **File Based** and click **OK**.

The **WSDL Input** dialog box (Figure 57) is displayed.

Figure 57: WSDL Input Dialog Box



- 7 On the **WSDL Input** dialog box (Figure 57), select the WSDL file to import and click **Open**.
The WSDL file gets added to the group in the **Project Navigator** pane.
- 8 Right click imported WSDL and Select **Activate**. (MDM need to be restarted if it is already running).

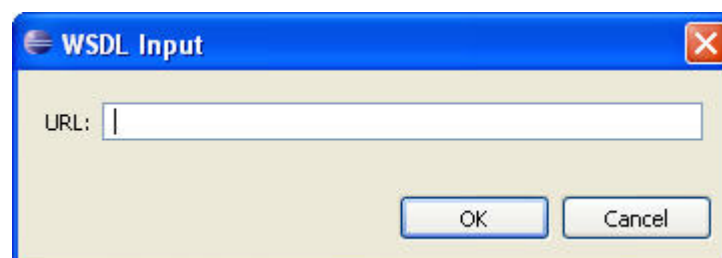
Sample Web Service

To enable Yahoo Web service:

- 1 Perform step 1 to 5 as in section “[Enable Web Services \(Optional Step\)](#)”.
- 2 On the **WSDL Input** dialog box (Figure 56), from the **Loading Mechanism** drop-down list, select the loading mechanism as **HTTP Based** and click **OK**.

The **WSDL Input** dialog box (Figure 58) is displayed.

Figure 58: WSDL Input



- 3 On the **WSDL Input** dialog box (Figure 58), in the **URL** field, enter the URL of the file to import and click **OK**.
Inset URL from
<MDM_BASE>\custom\sampleApplication\testdata\WSDL\YahooWorkflowWSDLURL.txt FILE.
- 4 Repeat the steps for remaining two URL's available in the above text file.
- 5 Right click imported WSDL and Select Activate (MDM need to be restarted if it is already running).
- 6 Register with xignite service using the URL <https://xignite.com/MyAccount/Register.aspx>
- 7 Open yahooWorkflow.xml in MDM Studio and click open Get Stock Quote1 WSDL node and add username and password registered in WSDL Transform Header Section.

Custom Application Folder Structure

MDM server side folder structures for custom application is <MDM_Installed_Directory>/Custom/<Custom Application Name>. The below table describes the folders available in custom application folder.

Folder Name	Required for	Description
batch	It is required if the custom application's register load and prepare loads are defined during schema generation process/testdata.	This folder and its subfolders are required to have custom E2E process like register load and prepare load process.
bin	It is required for custom application schema process and to start the services.	This folder has all the files required for custom application schema generation and to start the MDM services.
cfg	It is required for custom application schema process and to start the services.	This folder and its subfolders holds all custom application's properties files and service related files like workflows, rules, model instances and so on.
log	It is required for custom application schema process and to start the services.	This folder and its subfolders holds all custom application's service and base MDM's service log files.
models	It is required for custom application schema process and to start the services.	This folder and its subfolders have all custom application's models.
db	This folder gets created during schema generation process.	This folder has a SQL sentence structure of custom application.

Folder Name	Required for	Description
testdata	It is created during the schema generation process. It is required only if any stored procedure's or other custom schema process related files are located under this folders.	This folder have all custom application's predefined data or stored procedures.

Web server side folder structures for custom application is

Web\mdmclient\WEB-INF\custom\<Custom Application>\model\function.cnd

Or

Custom\<Custom Application>\model\core\navigation\function.cnd

The below table describes the folder available at: *Web\mdmclient\WEB-INF\custom\<Custom Application>*

Folder Name	Required for	Description
model	It is used by Web application server for left navigation.	Custom application's navigation can be defined.

The below table describes the file available at: *Web\mdmclient\WEB-INF\class*

Folder Name	Required for	Description
X2.properties	It is used by Web application server for left navigation.	Defines custom application's model path.

CHAPTER 8 Custom Models

What's In This Chapter

This chapter provides information about the powerful data modeling environment provided by MDM Studio. You can use the data modeling environment for defining the data models within an enterprise. MDM Studio provides a design environment for operating on metadata of the business models and it allows managing all aspects of the model.

Topics include:

- [Creating a Model using Studio](#)
- [Importing a Model into MDM Studio](#)

Creating a Model using Studio

Navigate to `<MDM_Install_Directory>/custom/sampleApplication/bin` and open the `sampleApplication.owb` file in Studio. For details on creating model using studio, refer to section *Create Models in Chapter 3 Data Modelling of MDM Platform Studio User Guide.pdf*

For the CDI sample application, the following Dictionary and Model files are created:

- Dictionary
 - SOURCE
 - `srdictionary.mdt`
- Model
 - SOURCE
 - i `CUSTOMER.mtt`
 - ii `LEGACY_CUSTOMER.mtt`
 - iii `PARTY_XREF_CUST.mtt`

Note: From MDM 3.01.01 release, `Legacy_Customer` is changed to `Customer_Source`.

For detailed information, refer to *MDM Platform Studio User Guide*.

Importing a Model into MDM Studio

You can use any of the following to import a model into MDM studio:

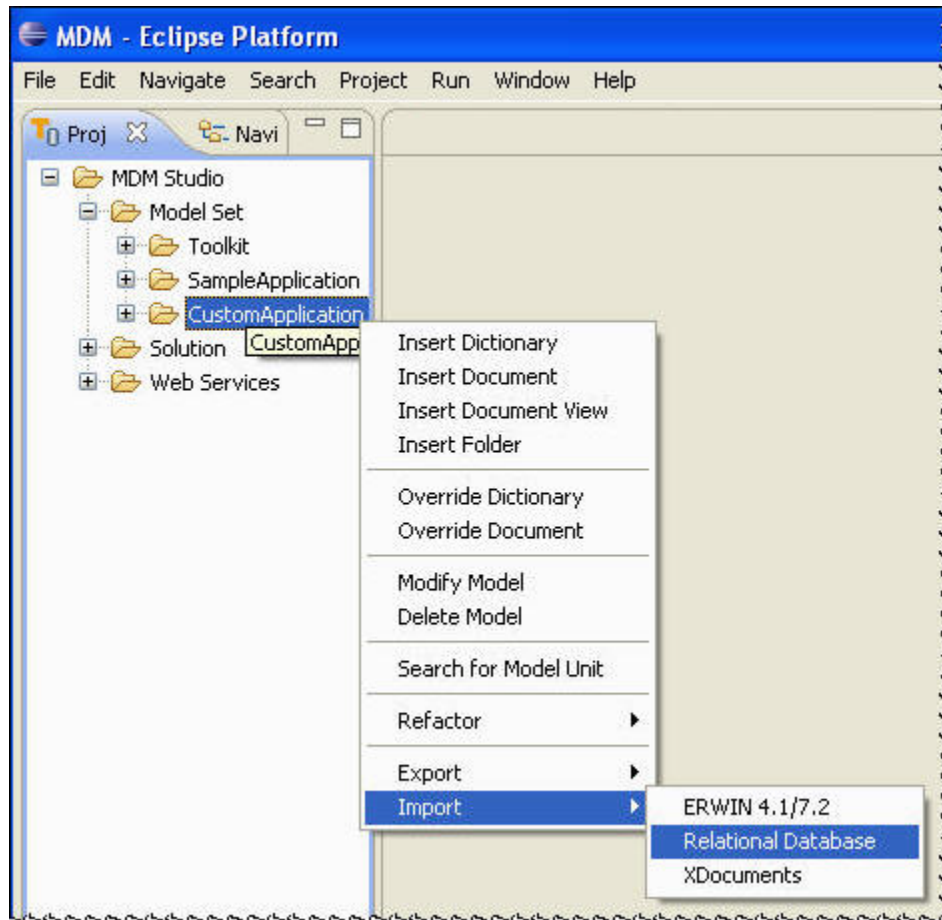
- Import from Relational Database

- Import from Erwin
- Import from XDocs

Import from Relational Database

- 1 On the MDM Studio window, in the Navigator pane, right-click on the custom model (Custom Application Model), point to Import and then select Relational Database as in [Figure 59](#).

Figure 59: Import from Relational Database



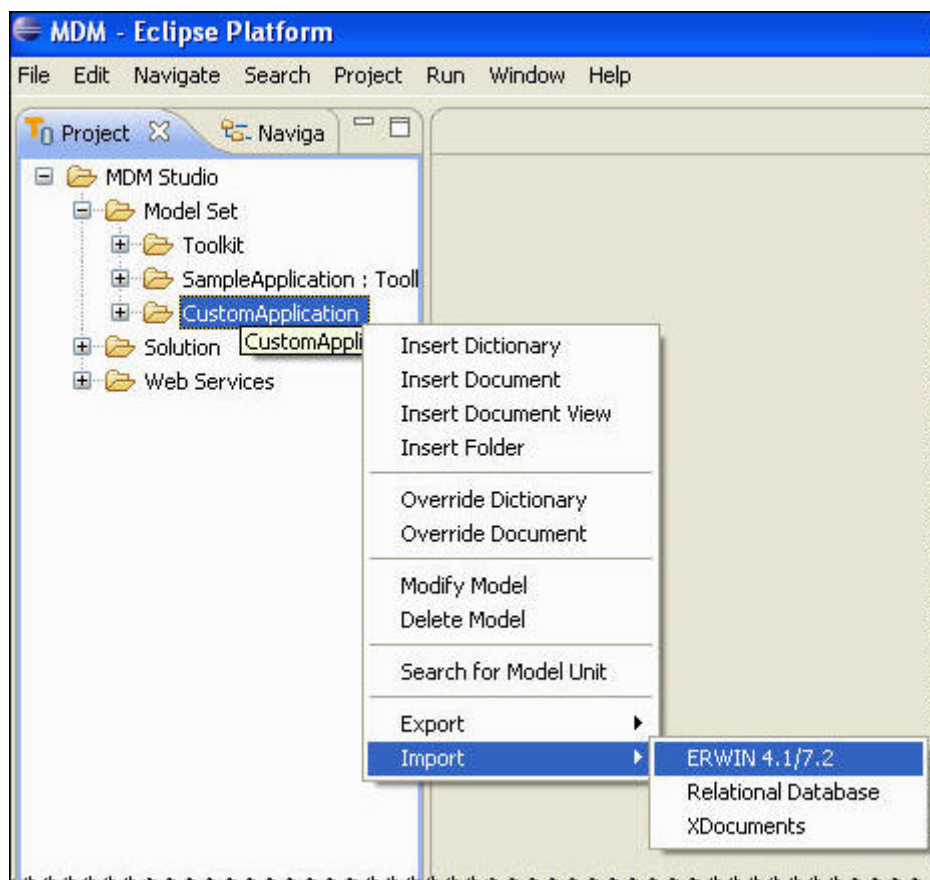
- 2 Follow the remaining steps as explained in section *Importing from Relational Database* in chapter 3 *Data Modeling* of the *MDM Platform Studio User Guide* along with the following steps.
- 3 On the Categorize Field Types page, in the Categorize Field Types pane, select the dictionary folder from Sample App Model and create a folder called CRM.
- 4 In the CRM folder, create a crmdictionary file and move all the field types from the Field Type pane to the crmdictionary file and click Next. The Categorize Tables page is displayed.
- 5 On the Categorize Tables page

- In the Categorized Tables pane, select the models folder from Sample App Model and create a folder called CRM.
 - From the Tables pane, select the following tables (ACCOUNT_MASTER, HOUSEHOLD_ACTIVITY_SUMMARY, and INDIVIDUAL_MASTER) and move it to the CRM folder.
 - Click Next. The Reconcile Model Set page is displayed.
- 6 On the Reconcile Model Set page,
- From the Action drop-down list, select the option Add All to Target and click Go.
 - Click OK to finish the database import process.

Import from ERwin

- 1 On the MDM Studio window, in the Navigator pane, right-click on the custom model (Custom Application), point to Import and then select Erwin 4.1 or Erwin 7.2 as in Figure 60.

Figure 60: Import from ERwin



- 2 Follow the remaining steps as explained in section *Importing from Erwin* in chapter 3 *Data Modeling* of the *MDM Platform Studio User Guide* along with the following steps.
- 3 On the Select ERwin File page, in the ERwin Source File field, select the Address file.

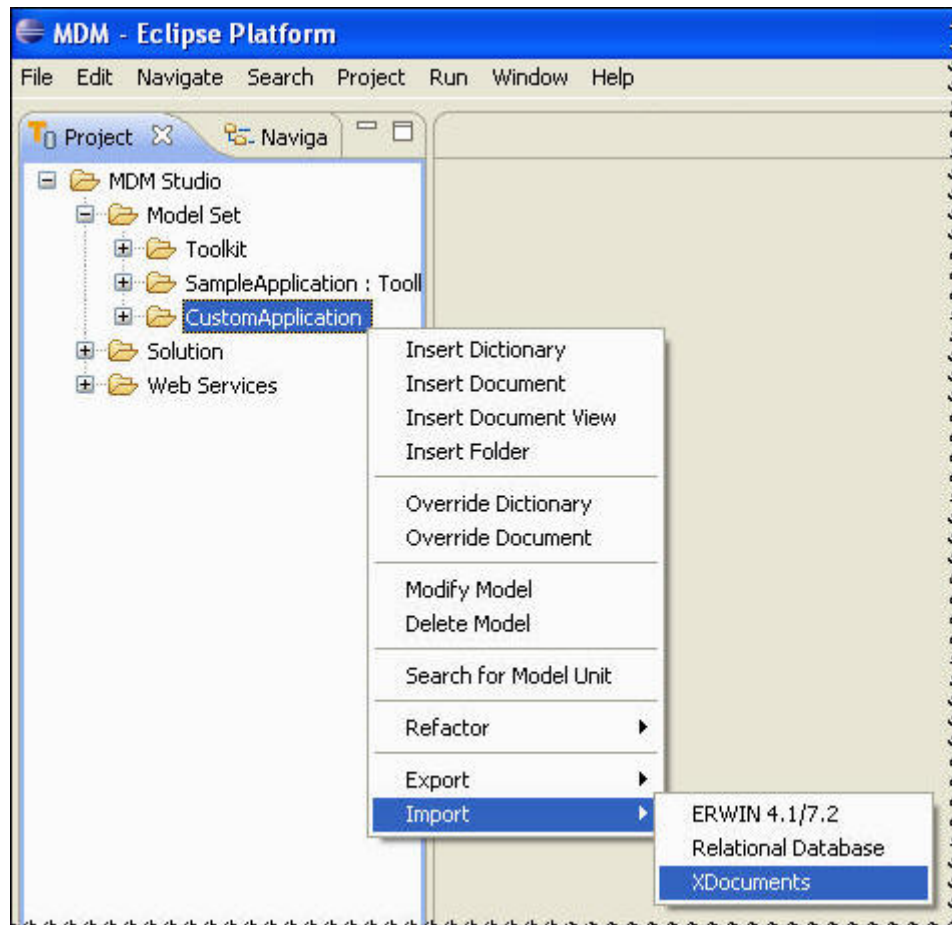
Note: The following sample ERwin files are available: Address, Party, Payment Account, and Vendor. But for the CDI Sample Application, only the Address file is selected.

- 4 On the Categorize Field Types page, in the Categorize Field Types pane, select the dictionary folder from Sample App Model and create a folder called RLDM.
- 5 In the RLDM folder, create an rldmdictionary file and move all the field types from the Field Type pane to the rldmdictionary file and click Next. The Categorize Tables page is displayed.
- 6 On the Categorize Tables page,
 - In the Categorized Tables pane, select the models folder from Sample App Model and create a folder called RLDM.
 - In the RLDM folder, create a folder called ADDRESS.
 - From the Tables pane, select the tables and move it to the ADDRESS folder.
 - Click Next. The Reconcile Model Set page is displayed.
- 7 On the Reconcile Model Set page,
 - From the Action drop-down list, select the option Add All to Target and click Go.
 - Click OK to finish the database import process.

Import from XDocs

- 1 On the MDM Studio window, in the Navigator pane, right-click on the custom model (Sample App Model), point to Import and then select Xdocs as in [Figure 61](#).

Figure 61: Import from XDocs



- 2 Follow the remaining steps as explained in section *Importing from Xdocs in chapter3 Data Modeling* of the *MDM Platform Studio User Guide* along with the following steps.
- 3 On the Categorize Field Types page, move all the field types from the Field Type pane to the existing sredictionary file and click Next.
- 4 On the Categorize Tables page, from the Tables pane, select the tables and move it to the existing SOURCE model folder and click Next. The Reconcile Model Set page is displayed.
- 5 On the Reconcile Model Set page,
 - From the Action drop-down list, select the option Apply Reconcile Indent and click Go.
 - Click OK to finish the database import process.

CHAPTER 9 Define Web Component

What's In This Chapter

This chapter provides information about the web component definition. MDM Studio provides a design environment for defining the navigation model structure, as well as managing all aspects of the model within the defined service.

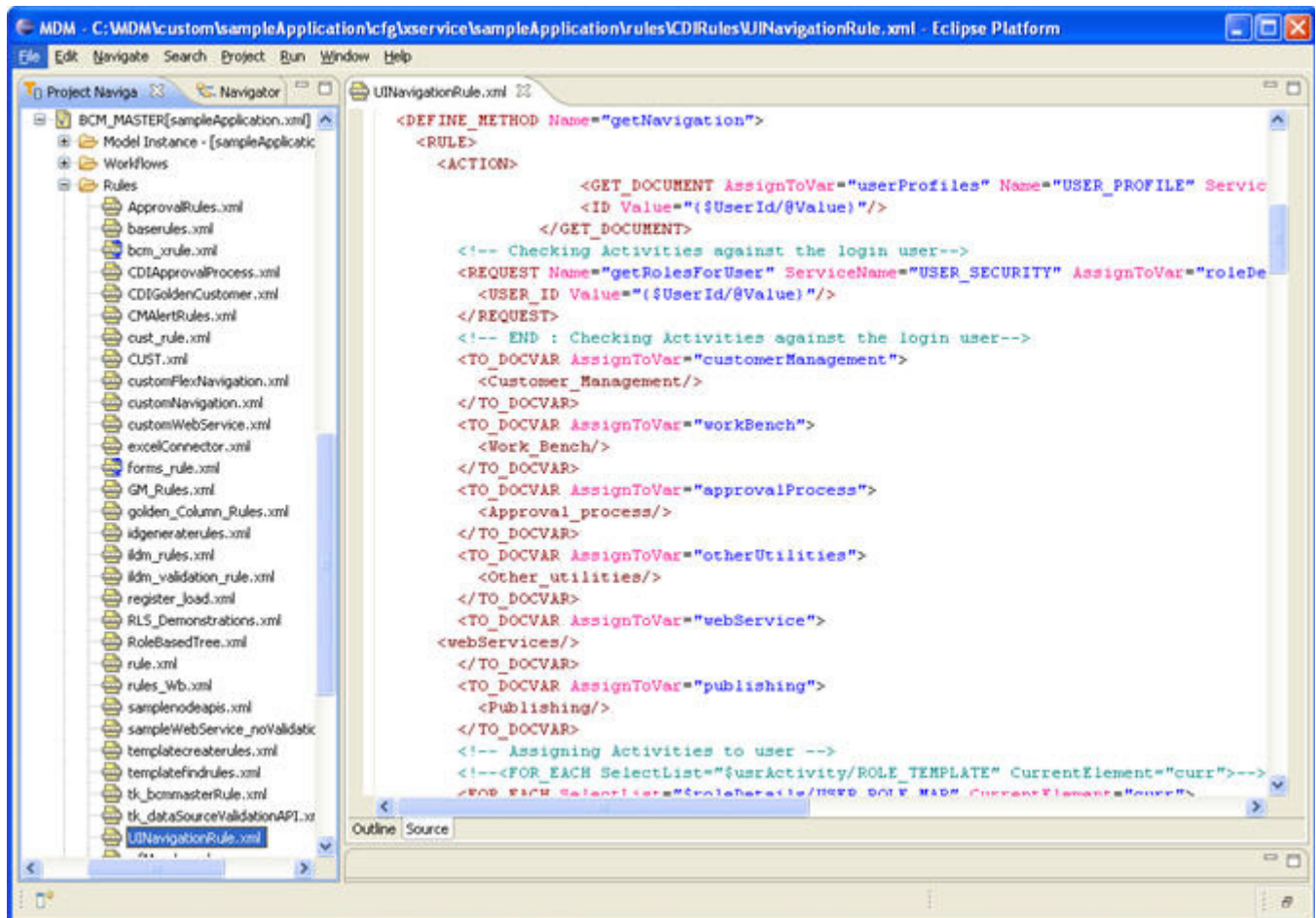
Topics include:

- [Define UI Navigation Structure](#)
- [Include into Web Component](#)

Define UI Navigation Structure

Navigation structure is defined in a workflow activity specifying how you want to expose your workflows in the MDM UI. For that you have to define the structure in the `UINavigation.xml` which is part of the service rule folder. While creating the new service, add from an existing service which has the `UINavigationRule.xml` rule file and then modify the structure according to your requirements.

Figure 62: Define UI Navigation Structure



In this rule, define a method that is called by the web component while the web service is running. Here you can define how you want to call the workflow activities. For example in this demo we have defined the activity as follows:

- Customer Data -> Level 1
- Customer Management – Sub level 2

Define the workflow

```
<TO_DOCVAR AssignToVar="CMTempXml">
<PAD_ITEM DisplayText="Customer Management">
<PAD_ITEM OnClick="/mdm/
start.x2ps?START_WORKFLOW=ApprovalInbox&SERVICE_NAME=BCM_MASTE
R" DisplayText="Inbox"/>
<PAD_ITEM OnClick="/mdm/
start.x2ps?START_WORKFLOW=SearchCustomer&SERVICE_NAME=BCM_MAST
ER" DisplayText="Search Customers"/>
<PAD_ITEM OnClick="/mdm/
start.x2ps?START_WORKFLOW=NewCustomerIntroduction&SERVICE_NAME
=BCM_MASTER" DisplayText="New Customer Introduction"/>
<PAD_ITEM OnClick="/mdm/
start.x2ps?START_WORKFLOW=CustomerDashboard&SERVICE_NAME=BCM_M
ASTER" DisplayText="Customer Dashboard"/>
</PAD_ITEM>
</TO_DOCVAR>
```

- Other Utilities – Sub level 2

```
<TO_DOCVAR AssignToVar="CPTempUtilXml">
<PAD_ITEM DisplayText="Other Utilities">
<PAD_ITEM OnClick="/mdm/
start.x2ps?START_WORKFLOW=alert_utilWF&SERVICE_NAME=BCM_MASTER
" DisplayText="Send an Alert"/>
<PAD_ITEM OnClick="/mdm/
start.x2ps?START_WORKFLOW=DataValidationError&SERVICE_NAME=BCM_MASTER" DisplayText="Batch Data Validation"/>
<PAD_ITEM OnClick="/mdm/
start.x2ps?START_WORKFLOW=load_data&SERVICE_NAME=BCM_MASTER" DisplayText="Load Data "/>
<PAD_ITEM OnClick="/mdm/
start.x2ps?START_WORKFLOW=reset_data&SERVICE_NAME=BCM_MASTER" DisplayText="Reset CDI Data"/>
<PAD_ITEM OnClick="/mdm/
start.x2ps?START_WORKFLOW=dirty_read&SERVICE_NAME=BCM_MASTER" DisplayText="Dirty Read"/>
<PAD_ITEM OnClick="/mdm/
start.x2ps?START_WORKFLOW=customerPublication&SERVICE_NAME=BCM_MASTER" DisplayText="Publishing Objects"/>
</PAD_ITEM>
</TO_DOCVAR>
```

- Level 1 Definition

```
<TO_DOCVAR AssignToVar="navigationXml">
<PAD Name="Studio" DisplayText="Customer Mgmt" Scrollable="yes">
<PAD_ITEM DisplayText="Customer Data">
<TO_XML SelectList="$CMTempXml"/>
<TO_XML SelectList="$CPTempUtilXml"/>
</PAD_ITEM>
</PAD>
</TO_DOCVAR>
<ADD_CHILDREN DocVar="thisReturn" FromSelect="$navigationXml"/>
```

This allows you to design your web workflow activity navigation.

Include into Web Component

After defining the workflow activity, include the structure file into the Web component located at *<MDM_Install_Directory>/web/mdmclient/WEB-INF*. It is recommended not to use the existing files and create an additional folder structure like below with a “cnd” file.

Custom/SampleApplication/model/core/navigation/function.cnd. For the sample cnd file, see step 4 in section “Custom Application Creation” of Chapter 7: “Custom Application”.

Once this is defined, add the custom model location into the model properties files.

Navigate to *<MDM_Install_Directory>/web/mdmclient/WEB-INF/classes* and open the *x2.properties* file and add the line */WEB-INF/custom/<CUSTOM-APPLICATION-NAME>/model;* into the model path used by MDM UI as below.

```
model-attribute.modelpath=/WEB-INF/custom/sampleApplication/model;/WEB-INF/bcm/
model;/WEB-INF/core/model;/WEB-INF/system;/WEB-INF/upload_kit/model
```

For more details, see step 4 in section “Custom Application Creation” of Chapter 7: “Custom Application”.

CHAPTER 10 Sample Application Process

What's In This Chapter

This chapter provides information about the sample application process.

Topics include:

- [Introduction](#)
- [Sample Application Data Model](#)
- [Sample Application CDI Process Flow](#)
- [Sample Application with Trillium Process Flow](#)

Introduction

The unstructured and unclean bulk data fed into the MDM Inbound staging area via an ETL tool or as a flat file. This bulk data is taken as input to compute the net change in data from the previous load. The net change data is then persisted into the MDM master tables.

Data is moved into the Legacy system of reference and then the Teradata MDM application is used to enrich the data, cleanse the data and match customer records. Based on the Teradata MDM application's matcher response, customer records are split into customer master records, cross reference and potential records. Using the data steward algorithm approach, all impending records are moved to a new customer or updated within an existing customer. Finally, the customer data is ready to publish to the outside world.

Sample Application Data Model

The data model utilized within this demonstration (loaded into MDM during installation) is based upon Teradata's logical data models. Within the Teradata data models, the primary subject area known as 'Party' supports the CDI sample application. 'Party' is a 'Subject Area' that is common across all the industry logical data models. For this version of the CDI sample application, the CDI data requirements do require additional data that is directly related to 'Party', but is more industry specific, such as Address, Payment Account, and Vendor (Supplier), from Retail LDM (due to familiarity only) as well as a few from the CRM model.

- CRM Model—It can be tailored more specifically to the particular needs of an organization and holds the customer's related data.
- PARTY—it has the flexibility in grouping entities and holds the customer's related data.
- ADDRESS—it holds the customer's address related data model and data.

In addition to this model, three more customer related tables (Legacy Customer, Customer and Party Customer Cross reference) similar to customer's consolidated model are required.

- Legacy Customer—holds the source system data or incoming data.

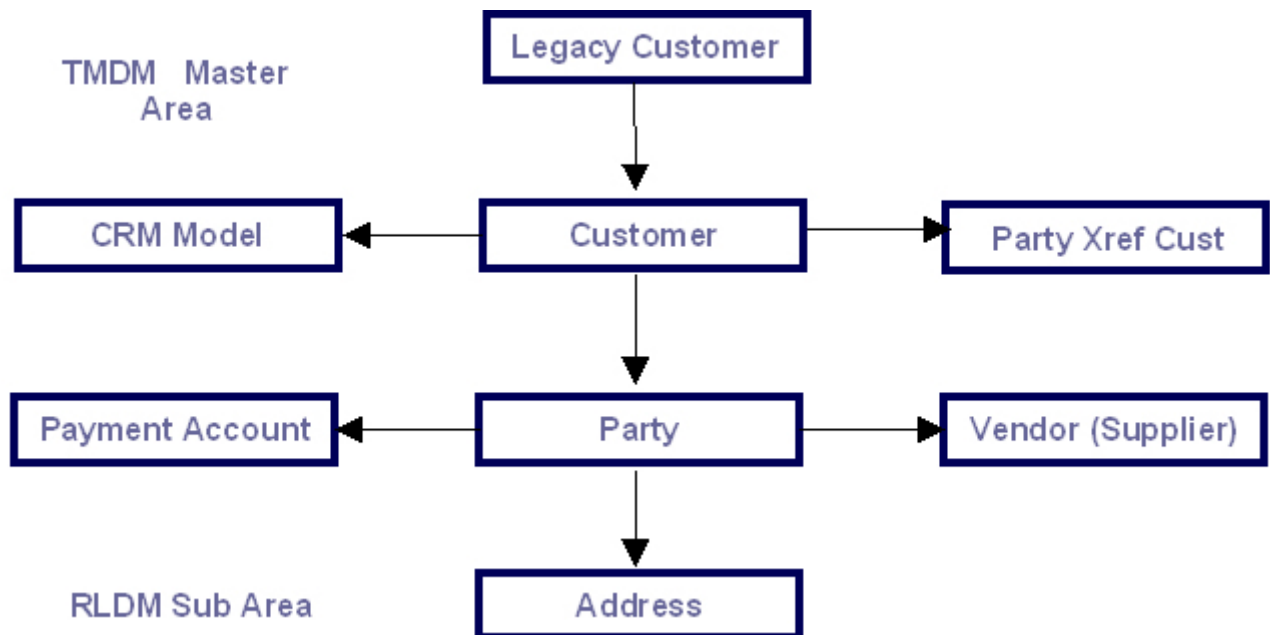
Note: From MDM 3.01.01 release, Legacy_Customer is changed to Customer_Source.

- Customer—holds the unique customer data.
- Party Customer Cross Reference—holds the customer related data.

For the demo purpose, other tables that exist in the database are not used.

Figure 63 displays the RLDM model and customer model relationship.

Figure 63: RLDM Model and Customer Model relationship



Sample Application CDI Process Flow

The CDI process starts with a Dashboard view of all Customer related information. From the dashboard, you can deal with duplicate resolution (merging/ survivorship), drill into customer details, and get into a customer enrichment process. For survivorship, the user can study the incoming record and compare it against the existing MASTER record. You can then interactively select one or more fields from one record and one or more from another record to form the winning record that will become the MASTER record. In addition, Search option will show all active available customers and drill into customer details for viewing or enrichment. You can also introduce new customer. To make the new customer active, this newly created customer must go through an approval process before it is available for further activities. The ability to enrich customer details is based on the role/user group of the user. In addition, the MDM UI activities are visible to a user based on the user group/role.

CDI Process Flow

CDI is the process of consolidating and managing customer information from all available sources, including contact details, customer valuation data and information gathered through interactions such as direct marketing. Properly conducted, CDI ensures that all relevant departments in the company have constant access to the most current and complete view of customer information available. As such, CDI is an essential element of customer relationship management.

Figure 64 displays the sample application CDI process flow.

Figure 64: Sample Application process flow



Figure 65: Overall CDI Process Flow

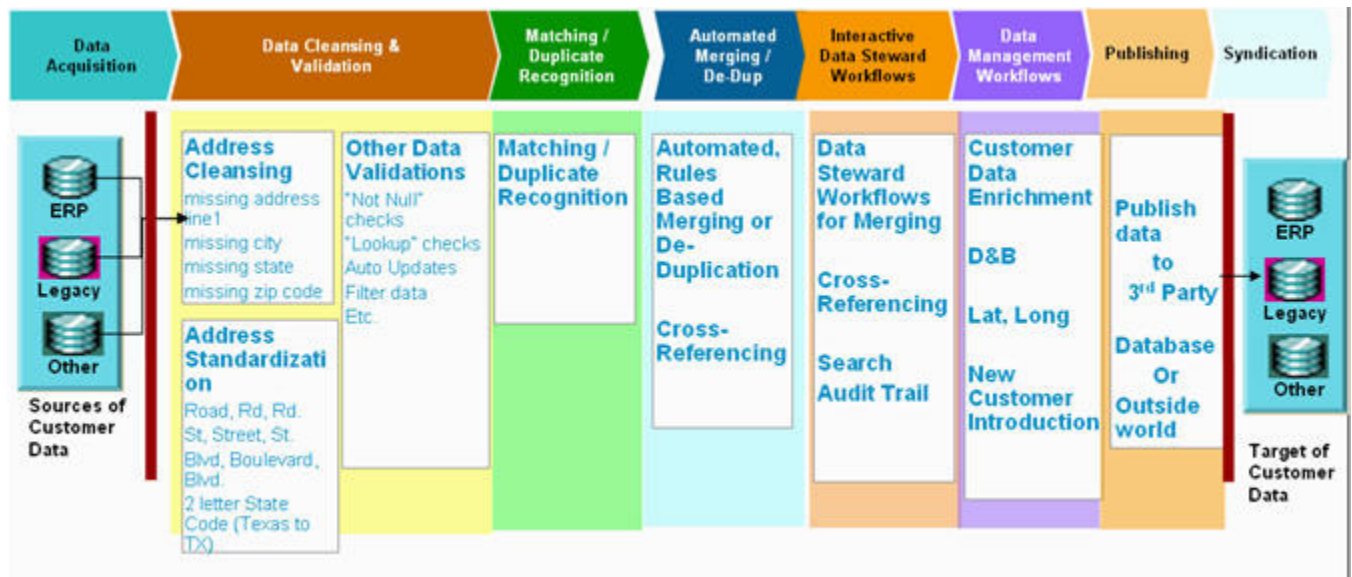


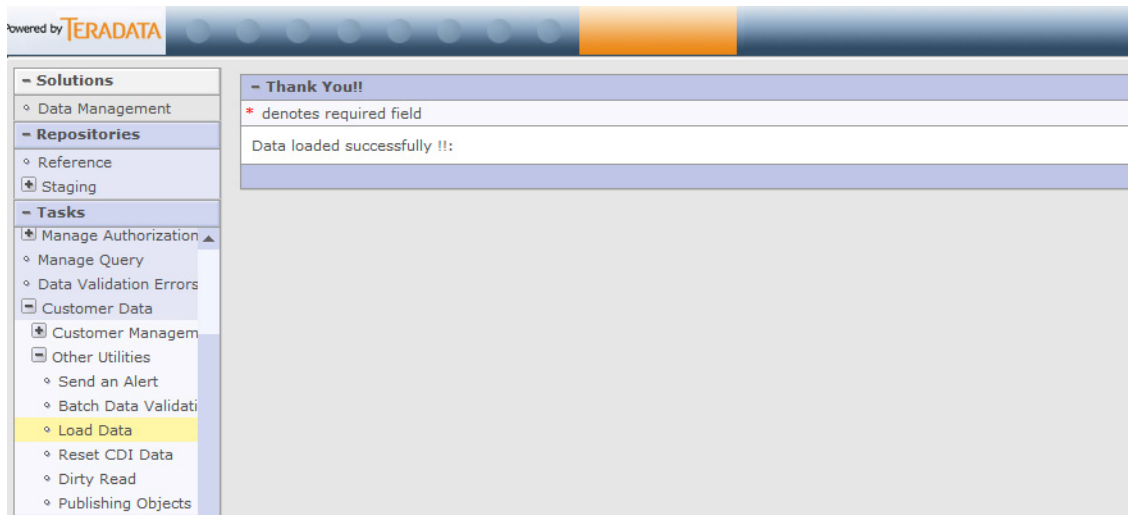
Figure 65 displays the overall CDI process flow using Teradata MDM. MDM handles data acquisition from the external source system or 3rd party legacy system, business validation, basic data type validation, other validations, data cleansing, data enrichment, matching and duplicate recognition, and publishing. Teradata MDM also handles processes like identifying the cross reference customer, interactive data steward UI process, customer management process and data reconstruction and syndication.

Data Acquisition

Data acquisition is a process of extracting, transforming, and transporting data from the source systems and external data sources to the data warehouse database objects. The data acquisition process can be achieved via ETL (Extraction Transformation and Load) or EII (Enterprise Information Integration) technologies.

For the CDI sample application, a workflow called “Load Data” loads the sample demo data into the Teradata MDM inbound staging area. The LEGACY_CUSTOMER (IN_LEGACY_CUSTOMER) table gets involved during the data acquisition process.

Figure 66: Load Data



Data Validation and Enrichment

Using Business rule builder, you can define business rules for data validation, rule based data update/insert, rule based filtering of records etc. You can define SQL validation, update, insert and filter rules. The following section explains the different types SQL rules.

Rule Type: SQL_Filter

The sql filter rule will filter the records satisfying the defined filter rules in the inbound staging itself during E2E. Before netchange is computed, all such records are removed from the input tables that participate in the specific E2E load.

For example, in CDI demo, it filters records which belong to COUNTRY as "India" from source table (LEGACY_CUSTOMER) as displayed in [Figure 67](#).

Figure 67: Rule Type: SQL_Filter



Rule Type: SQL_Validation

You can define validation rules of three different severities - WARNING, ERROR and SEVER ERROR. In CDI demo, the WARNING and ERROR types are demonstrated.

During Do_Db_Persist, the validation rules are executed one by one and for each error it appends the error code to the SYS_ERR_CODE column of the records that fail the validation.

Figure 68: Rule Type: SQL_Validation

The screenshot shows a software interface with a left-hand tree view and a main content area. The tree view includes items like ItemMainFrame, JobFunction, JobLevel, and LEGACY_CUSTOMER (which is highlighted). The main content area is titled 'Search Rules' and contains a 'Search Criteria' section with dropdowns for 'Rule Type' (set to 'sql_validation') and 'Status' (set to 'ACTIVE'). A 'Search' button is present. Below this is a 'Search Results' table with the following data:

Rule Name	Rule Type	Status	Description
FAMILY MEMBER MA	sql_validatio	ACTIVE	FAMILY MEMBER MAY NOT BE MORE THAN THREE
CREDIT_RISK_CLA	sql_validatio	ACTIVE	CREDIT_RISK_CLASS

SQL_Validation Rule with Severity WARNING

Validation rules with severity WARNING are classified as SOFT ERROR rules. When a record fails such a validation, it is considered as erroneous but not a severe error and will not be restricted to move to master. When a record fails a validation of severity WARNING, the SYS_ERR_SVRTY for such record is marked as WARNING and at the end of the validation process these records are copied to the corresponding Master and Error tables unless the same record fails another validation of higher severity (ERROR or SEVERE ERROR).

In CDI demo, in the LEGACY_CUSTOMER table, the records that have FAMILY_MEMBER values as more than 3 are marked as records having SOFT ERRORS (severity WARNING).

Figure 69: Rule Type: SQL_Validation with Severity WARNING

The screenshot shows the 'Rule Details' interface. The left-hand tree view is similar to Figure 68, with LEGACY_CUSTOMER highlighted. The main content area shows details for a specific rule:

- Rule Name: FAMILY MEMBER MAY NOT BE MORE
- Status: ACTIVE
- Description: FAMILY MEMBER MAY NOT BE MORE THAN THREE
- Rule Type: sql_validation
- Start Date(mm/dd/yyyy): //
- End Date(mm/dd/yyyy): //

 Below these fields are two tabs: 'Expression Builder' and 'Expression Viewer'. The 'Expression Viewer' is active and displays the following SQL expression:


```
update T1 from MST_LEGACY_CUSTOMER T1
SET SYS_ERR_CODE = (COALESCE(T1.SYS_ERR_CODE,"") ||";FAMILY MEMBER MAY NOT BE MORE THAN THREE") , SYS_ERR_SVRTY = "W"
where T1.FAMILY_MEMBERS > 3
```

SQL_Validation Rule with Severity ERROR

Validation rules with severity ERROR or SEVERE ERROR are classified as HARD ERROR rules. A record that fails such a validation is erroneous and must be restricted from moving to master. When a record fails a validation of severity ERROR or SEVERE ERROR, the

SYS_ERR_SVRTY for such record is updated with appropriate error and at the end of the validation process these records are moved to the corresponding error table.

In CDI demo, in the LEGACY_CUSTOMER table, records that have FAMILY_MEMBER values as more than 3 must be marked as ERRORS.

Figure 70: Rule Type: SQL_Validation with Severity Error

Rule Details

Rule Name: Rule Type:

Status:

Description:

Start Date(mm/dd/yyyy):

End Date(mm/dd/yyyy):

Expression Viewer

```
update T1 from MST_LEGACY_CUSTOMER T1
SET SYS_ERR_CODE = (COALESCE(T1.SYS_ERR_CODE,"") || 'CREDIT_RISK_CLASS_DATA_IS_ZERO_ERROR') , SYS_ERR_SVRTY = 'ERR
where T1.CREDIT_RISK_CLASS = 0
or T1.CREDIT_RISK_CLASS IS NULL
```

Note: While creating the business rules of type validation, all rules with severity WARNING must be kept above the rules of the other two severities. This is later used by the Data Quality Soft error reports.

Rule Type: SQL_INSERT

Insert rule allows data enrichment on the input data. It can be defined using Business Rule Builder capability available on MDM Web UI. It can only perform INSERT operation on the same table/other table using the input record values as per defined criteria. Scenarios like inserting child records in the dependent entities can easily be accomplished through this type of business rule.

During DO_DB_Persist processing, these rules get executed by default after input record is persisted successfully into master table. The insert rules are action based, i.e., at the time of definition one can declare action type (DO_DB_Persist API action) upon which given insert rule should get executed.

Note: Sql_Insert rules do not execute if a custom xrule, defined through data persist config spec, is executed as part of DO_DB_Persist action API. For more details, refer to *Chapter Business Rules in MDM Platform Server Guide*.

In CDI demo, whenever record is inserted into the LEGACY_CUSTOMER table of Master area, the same record in the same table has to be updated with CUST_STATE='INCOMING' for identifying the new incoming records.

Figure 71: Rule Type: SQL_Insert

Rule Details

Rule Name: * Rule Type:

Status: *

Description:

Start Date(mm/d/...

End Date(mm/dd/...

Expression Builder **Expression Viewer**

```
update T1 from MST_LEGACY_CUSTOMER T1
set CUST_STATE = 'INCOMING',
    SYS_ENT_STATE = 'ACTIVE',
    SYS_LAST_MODIFIED_BY = T1.SYS_LAST_MODIFIED_BY,
    SYS_LAST_MODIFIED_DATE = current_timestamp(0),
    SYS_SOURCE = T1.SYS_SOURCE
where T1.ORG_ID = T1.ORG_ID
    and T1.COUNTRY <> 'INDIA'
    and T1.CUST_ID = T1.CUST_ID
    and T1.CUST_STATE = "
```

Rule Type: SQL_UPDATE

Similar to Insert rule, update rule also allows data enrichment on the the input data. It can be defined using Business Rule Builder capability available on MDM Web UI. However, unlike insert rule, Update rule can only perform UPDATE operation on the same table/other table using the input record values as per defined criteria. During DoDbPersist processing, these rules get executed by default after input record is persisted successfully into master table.

The update rules are action based, i.e., at the time of definition one can declare action type (DO_DB_Persist API action) upon which given update rule should get executed.

Note: Sql_Update rules do not execute if a custom xrule, defined through data persist config spec, is executed as part of DDP action API. For more details, refer to *Chapter Business Rules in MDM Platform Server Guide*.

The following rules are created for the CDI sample application:

The Inbound service and LEGACY_CUSTOMER table.

SQL_FILTER - Type

LEGACY_CUST_FILTER_CTY_INDIA

The Master service and LEGACY_CUSTOMER table.

SQL-VALIDATION - Type

FAMILY MEMBER MAY NOT BE MORE THAN THREE - WARNING

CREDIT_RISK_CLASS_SQLValidation - ERROR

SQL-INSERT - Type

Modify_CustState_To_INCOMING - Query Type - Update

To load the input staging data into the master tables, call the Data Load Workflow (E2E) that is a part of Inbound staging service. For detailed information on E2E, refer *MDM Platform Server Guide*.

Data Cleansing

The data cleansing process identifies and isolates specific parts of mixed data and standardizes data based on information stored in the database. In the CDI sample application, data cleansing is performed for Address attributes. You can achieve this by defining a rule, which is part of the Master Service and is called by the E2E process's "Pre Custom Action".

This workflow node is located at the beginning of the E2E data load process workflow. The data cleansing rule works mainly on the inbound staging tables and by default called by the E2E process, hence include the data cleansing rule in the customize rule file, which is part of the E2E service.

The below xcore request will execute the data standardization rule.

```
<REQUEST AssignToVar="custStateIncomingData" Name="enrichIncomingData"
ServiceName="BCM_MASTER"/>
```

The data standardization rules cleanse and standardize the data before moving it to the reference (Master) area table. The method is defined in the customRules.xml of the E2E service and it is part of RULE.

```
<REQUEST AssignToVar="modBLVD" ServiceName="BCM_MASTER" Name="Modify_BOULEVARD_to_BLVD"/>
<REQUEST AssignToVar="modDR" ServiceName="BCM_MASTER" Name="Modify_DRIVE_to_DR"/>
<REQUEST AssignToVar="modRD" ServiceName="BCM_MASTER" Name="Modify_ROAD_to_RD"/>
<REQUEST AssignToVar="modST" ServiceName="BCM_MASTER" Name="Modify_STREET_to_ST"/>
<REQUEST AssignToVar="modUSA" ServiceName="BCM_MASTER" Name="Modify_UnitedStates_to_USA"/>
```

The details of the data standardization rules are given below:

Rule Modify_BOULEVARD_to_BLVD is modify the word "BOULEVARD" into "BLVD" in the Address column values of the In bound Legacy_Customer Table or Document

Rule Modify_DRIVE_to_DR is modify the word "DRIVE" into "DR" in the Address column values of the In bound Legacy_Customer Table or Document

Rule Modify_STREET_to_ST is modify the word “STREET” into “ST” in the Address column values of the In bound Legacy_Customer Table or Document

Rule Modify_ROAD_to_RD is modify the word “ROAD” into “RD” in the Address column values of the In bound Legacy_Customer Table or Document

Rule Modify_UnitedStates_to_USA is modify the word “UnitedStates” into “USA” in the Country column values of the In bound Legacy_Customer Table or Document

After the E2E process runs successfully, all unique customer records are moved to the master area of the Legacy Customer table. Now that table has standardized customer data. If the E2E process finds any error record then those records are moved to the respective error table of the master service. During this process, the following physical table is gets involved:

LEGACY_CUSTOMER (MST_LEGACY_CUSTOMER)

After execution of the above rules, LEGACY_CUSTOMER (MST_LEGACY_CUSTOMER) table will have all cleansed records. All of the records are cleansed, enriched and the status has been updated with CUST_STATE set to ‘INCOMING’.

Figure 72: LEGACY_CUSTOMER

CUST_ID	MATCH_ID	ORIG_ID	CONTACT	ADDRESS1	ADDRESS2	CITY	STATE	ZIP
251753	2116	BU-MRO	MIKE BRADY	1209,SE FEDERAL HWY		STUART	FL	34994
251579	2069	BU-WYW	ANTHONY MATHON	2520,SE WILLOUGHBY BLVD		STUART	FL	34991
251576	2065	BU-MRO	JOHN MACGAIN	1655,EE WALTON RD		LOUL GAIN LUCIE	FL	34952
201992	2100	BU-HWT	STEFAN L ANDON	111, WINTER AVE		LEEDSBURG	FL	34743
251720	2103	BU-USAB	MARIA CILIC	1215, GLENWOOD, DR		STUART	FL	34994
252023	2186	BU-MRO	AGIOS NIKOLAOS	7600W, BRONSON MEMORIAL HWY		KISSIMEE	FL	34717
251564	2061	BU-MRO	BRAU HOCU	3415,EE CUBA WAY		STUART	FL	34992
201002	2125	BU-ELECTRICAL	MKT BRADY	13000, RAINBOW LN		CLERMONT	FL	34715
251942	2167	BU-WYW	RONALD MICHAEL	7601, SW LOST RIVER RD		STUART	FL	34997

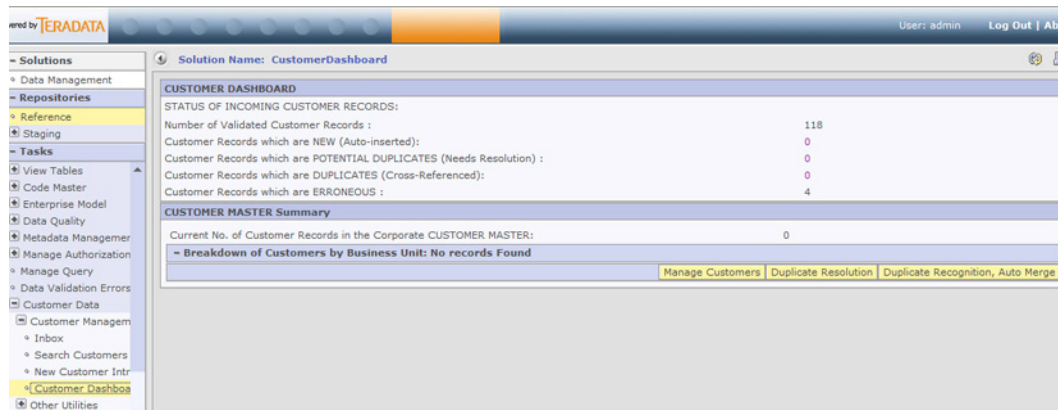
In the Address1 field data were Standardized
Like Road → RD

Matching/Duplicate Recognition & Automated Cross-reference

This process identifies matched and duplicate records and displays a single customer view. The matching and duplicate recognition requires the Customer Dashboard workflow, which is part of the sample application's master service.

Launch the **Customer Dashboard** page on the MDM UI. On the MDM UI, expand **Tasks** bar, expand **Customer Data**, expand **Customer Management** and click **Customer Dashboard** as in [Figure 73](#).

Figure 73: Customer Dashboard



On the **Customer Dashboard** page, you can perform the match, duplicate recognition, resolve the duplicate, and manage the customer functions using the following buttons.

- Manage Customers
- Duplicate Resolution
- Duplicate Recognition, Auto Merge

Click on **Duplicate Recognition, Auto Merge** button for executing matching & de-duplication process. This action extracts all records from MST_LEGACY_CUSTOMER with CUST_STATE equal to 'INCOMING'. The match process checks to see if all required fields in MST_LEGACY_CUSTOMER match against the MST_CUSTOMER. If MST_CUSTOMER matches MST_LEGACY_CUSTOMER then this record is an exact match and is called a FULL MATCH record. If it is not the same as MST_LEGACY_CUSTOMER then this process assumes that it might be a potential duplicate or suspect match.

For the suspect or potential duplicate match process, all required fields in MST_LEGACY_CUSTOMER are checked against the MST_CUSTOMER. If either one of the match returns true then the application assumes that the current record is a Suspect Match or Potential duplicate record. If it is a suspect or potential match record then update the MST_LEGACY_CUSTOMER's MATCH_ID equal to MST_CUSTOMER's PARTY_ID value. If it does not return any match or Suspect match then the current record is a New Customer record.

[Figure 74](#) displays the results of match process.

Figure 74: Customer Dashboard—Summary

Solution Name: CustomerDashboard	
CUSTOMER DASHBOARD	
STATUS OF INCOMING CUSTOMER RECORDS:	
Number of Validated Customer Records :	0
Customer Records which are NEW (Auto-inserted):	116
Customer Records which are POTENTIAL DUPLICATES (Needs Resolution) :	1
Customer Records which are DUPLICATES (Cross-Referenced):	1
Customer Records which are ERRONEOUS :	4
CUSTOMER MASTER Summary	
Current No. of Customer Records in the Corporate CUSTOMER MASTER:	116
- Breakdown of Customers by Business Unit	
Business Unit	No. of Customers
BU-APEX	1
BU-BUILDING MATERIAL	3
BU-CORPORATE	2
BU-ELECTRICAL	10
BU-MECHANICAL INDUST	1
BU-MRO	55
BU-NWW	27
BU-PLUMBING	10
BU-USABB	2
BU-WATER And SEWER	5
BU-WC	1
Manage Customers Duplicate Resolution Duplicate Recognition, Auto Merge	

In the Duplicate Recognition, Auto Merge process the following tables are involved:

- MST_LEGACY_CUSTOMER
- MST_CUSTOMER
- MST_PARTY_CUST_XREF
- MST_PARTY
- MST_PARTY_ADDRESS

Interactive Data Steward

The interactive data steward workflows for de-duping, merging, and cross-referencing are used in cases where automation does not yield the expected results because of a duplicate match or suspect match. To execute the interactive data steward workflow, click **Duplicate Resolution** button on the **Customer Dashboard** page (Figure 74). The **Survivorship** page (Figure 75) is displayed.

Figure 75: MDM Survivorship UI

Solution Name: CustomerDashboard > Survivorship_PGL

Survivorship

- Potential Duplicates in the Incoming Customer Records: Page 1 of 1

	Business Unit	Source CUST_ID	Customer Name	SITE_DUNS	ADDRESS1	ADDRESS2	CITY	STATE	ZIP	PHONE
<div><div></div><div></div></div>	BU-PLUMBING	381679	WOODLANDS CHURCH LAKE LLC	148661481	1231,SPRING,CREEK		GROVELAND		34736	3524294192
<div><div><div><div></div><div></div><div></div><div></div></div><div>Jump</div></div><div>Ignore Match, Create as New</div></div>										

- Potential Match in Customer Master

	Corporate CUST_ID	Customer Name	SITE_DUNS	ADDRESS1	ADDRESS2	CITY	STATE	ZIP	PHONE
	119	WOODLANDS CHURCH	148661481	1231,SPRING,CREEK		GROVELAND	FL	34737	3524294192
<div>Interactive Merge</div>									<div>Done</div>

This **Survivorship** page (Figure 75) displays all potential duplicate matches in the incoming customer records. On the **Survivorship** page (Figure 75), you can either “Ignore the Suspect or Duplicate Match” and create a New Customer record or choose “Interactive merge”. For Interactive merge, both the Master Customer record and the Incoming record are displayed for comparison. Based on this comparison, you can select the relevant information from the both records and then click ‘SAVE MERGE’ as in Figure 76. This will update the existing master with selected information from both records.

Figure 76: Interactive Merge Workflow UI

The screenshot shows a web application interface for the 'Survivorship' workflow. The breadcrumb trail at the top reads: 'Solution Name: CustomerDashboard > Survivorship_PGL > Survivorship'. The main section is titled 'Customer Details' and is divided into two columns: 'CUSTOMER MASTER:' and 'Incoming Customer Record:'. Each column contains a series of input fields for various customer attributes, with radio buttons next to each field to select which record's data to use for the merge. The attributes being compared are: Corporate Customer ID, Source Customer ID, Business Unit, Customer Name, DUNS Number, Address1, Address2, City, State, Zip Code, and Phone. The 'Save Merge' button is located at the bottom right of the form.

Field	CUSTOMER MASTER:	Incoming Customer Record:
Corporate Customer ID:	119	
Source Customer ID:		381679
Business Unit:		BU-PLUMBING
Customer Name:	<input checked="" type="radio"/> WOODLANDS CHURCH	<input type="radio"/> WOODLANDS CHURCH LAKE LLC
DUNS Number:	<input checked="" type="radio"/> 148661481	<input type="radio"/> 148661481
Address1:	<input checked="" type="radio"/> 1231,SPRING,CREEK	<input type="radio"/> 1231,SPRING,CREEK
Address2:	<input checked="" type="radio"/>	<input type="radio"/>
City:	<input checked="" type="radio"/> GROVELAND	<input type="radio"/> GROVELAND
State:	<input checked="" type="radio"/> FL	<input type="radio"/> FL
Zip Code:	<input checked="" type="radio"/> 34737	<input type="radio"/> 34736
Phone:	<input checked="" type="radio"/> 3524294192	<input type="radio"/> 3524294192

Save Merge

In the above process the following tables are involved:

- MST_LEGACY_CUSTOMER
- MST_CUSTOMER
- MST_PARTY_CUST_XREF
- MST_PARTY
- MST_PARTY_ADDRESS

Manage Customer

On the Customer Dashboard page, clicking on the Manage Customer button launches another workflow called Data Management workflows, which will be used to create a New Customer, New Customer Introduction or manage the existing customers. On the **Customer Dashboard** page (Figure 74), click **Manage Customer**. The **Manage Customer** page is displayed. On the **Manage Customer** page, you can modify or enrich existing master customer records. Based on the authorization of the user logged in, they are allowed to either modify the record or just view the details.

Figure 77: Manage Customer workflow UI

Solution Name: CustomerDashboard > Manage Customers

- Search Customer Master

Corp Customer ID: Address 1:

Customer Name: City:

Duns Number: State:

Credit Risk Class: Zip:

[Export Search](#) [Search](#)

- Search Results: Page 1 of 6

Corp Customer ID	Customer Name	Address 1	City	State	Zip	Duns Number	Credit Risk Class
208	WILLISAMS MIKE	101,LE GRAND, BLVD	LEESBURG	FL	34748	119171895	2
144	WILLISAMS COOL	611,S 13TH,ST	FORT PIERCE	FL	34950	120349019	3
156	WAUGH STEVE	5000,STARIS,AVENUE	KISSIMMEE	FL	34746	877257170	3
143	TOMAS BERDYCH	7900,SUS,HIGHWAY	PORT SAINT LUCI	FL	34952	836235101	1
121	TOM VINY	4577,MIGHTY GREY, RD	FORT PIERCE	FL	34947	606724156	4
163	TOM SUN	110,CLINTON, DR	KISSIMMEE	FL	34742	167198592	3
137	TOM SON	3793,NE OCEAN,BLVD	JENSEN BEACH	FL	34957	77283547	3
198	TOM POUL	15838,ORANGE,AVE	FORT PIERCE	FL	34945	800077448	2
110	TOM MATHEW	821,LAKE ELLARD	FRUITLAND PARK	FL	34731	197018880	3
145	TOM COOL	700,S 29TH,ST	FORT PIERCE	FL	34947	51127686	4
199	TOM CHRIS	4755,SE MANATEE,TER	STUART	FL	34997	831001201	1

[View Details](#) [View Cross Reference](#) [Enrich Customer](#)

In the above process the following tables are involved:

- MST_LEGACY_CUSTOMER
- MST_CUSTOMER
- MST_PARTY_CUST_XREF
- MST_PARTY
- MST_PARTY_ADDRESS

Manage an existing customer activities include:

- Search the Customer using the workflow “Search Customer” which is located in the Navigation “Customer Data -> Customer Management -> Search Customer”.
- Select the desired customer and modify the existing details, add more details, view the details or get more information about the cross reference details.
- BILLING_INVOICE_GRP user group can either modify or add more billing information for a selected customer.
- SHIPPING_DELIVERY_GRP user group can either modify or add more shipping information for a selected customer.
- CUSTOMER_SERVICE_GRP user group can either modify or add more customer profile information for a selected customer.

In the above process the following tables are affected:

- MST_CUSTOMER
- MST_PARTY_CUST_XREF
- MST_PARTY
- MST_PARTY_ADDRESS
- plus RLDM model tables DUSN and BUSINESS

Authentication services represent the initial login security of the solution. Authentication will be based on an organizational model for the enterprise. The intent is to provide a single logon security context.

In CDI sample application, there are three different user group:

- Task
 - Modify or enrich the customer profile.
 - Add, modify or enrich the billing information.
 - Add, modify or enrich the shipping information.
- User Group
 - CUSTOMER_SERVICE_GRP
 - BILLING_INVOICE_GRP
 - SHIPPING_DELIVERY_GRP

Modification, Addition or Enrichment permissions:

- If login user belongs to CUSTOMER_SERVICE_GRP then they are allowed to modify the customer profile information. Other details can be viewed.
- If login user belongs to BILLING_INVOICE_GRP then they are allowed to modify or add the customer's billing information. Other details can be viewed.
- If login user belongs to SHIPPING_DELIVERY_GRP then they are allowed to modify or add the customer's shipping information. Other details can be viewed.
- If login user belongs to Administrator then that person can modify, add or enrich all information.

A user who belongs to the CUSTOMER_SERVICE_GRP or ADMIN user group will be able to create a new customer. Figure 78 displays the New Customer Introduction page. Once a new customer is created then this record's status is inactive. An Administration group user has to approve the new customer request, otherwise this record will stay in the inactive status. Once approved by an admin user, the new customer record becomes active and is available for modification or enrichment by other users.

Figure 78: New Customer Introduction UI

New Customer with Corporate Customer ID 217 and Effective Start Date of 02/02/2008 has been sent for approval

Customer Information	
Corp Customer Name:*	Steve Bob
Contact Name:*	Steve Bob
Customer Type:*	Individual
Belongs to Organization:*	BU-CORPORATE
Eff. Start Date:*	2/2/2008
Address Line 1:*	854 Maria Lane
Address Line 2:	
City:*	Sunnyvale
State:*	CA
Country:*	USA
Zip:*	98046
Phone:*	4081237689
Business Name:	
Industry:	
DUNS Number:*	123456789
Additional Information	
Geo Unit Name:	
Div Unit Name:	
Credit Risk Class:*	2
Financial Stress Score:	

Cancel

Publishing

Publication services is a new feature in MDM 2.0 that allows the Data Steward to control the usage and flow of Master Data to end users or to consuming processes or applications. Publication Services allows any data within the scope of MDM to be published in multiple formats to meet the needs and requirements of consuming application or processes. Consuming applications or processes can retrieve data directly from the publication database, or they can have data pushed to them from a JMS Provider Queue table via JMS Messaging. Additionally, data can be extracted from the database into file format, and emailed directly to business users. Along the way, an audit trail of each publication request is preserved, including (optionally) the actual data published with each request.

Four types of publishing are supported:

- Database Table (Teradata)
- Java Message service (JMS) Queue Table
- Excel
- Text—Comma Separated File

The CDI sample application demonstrates only publishing to a Database Table method. The publishing object is defined during design time (before bringing up the MDM server). Once defined, the publishing facilities can be used at any point in a workflow to publish any data.

For detailed information, refer the *MDM Platform Studio User Guide*.

Define the service file (sampleApplication.xml – BCM_MASTER) as in [Figure 79](#). [Figure 80](#) displays the workflow to publish data.

Figure 79: Define Service File

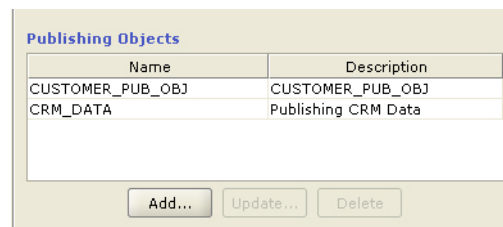


Figure 80: Workflow to Publish Data



Data Syndication

Data Syndication exposes the master data to the outside world, 3rd party system or back to the source system. Using ETL or EAI, you can access the data from published object type of format.

Sample Application with Trillium Process Flow

Summary

The sample application with Trillium (CDI) is the same as sample application with CDI except it uses the Trillium Cleanse and Match functionality to identify duplicates and also while introducing the new customers.

Trillium Process Flow

Figure 81 and Figure 82 displays the sample application with Trillium process flow.

Figure 81: CDI Process Flow (Trillium)

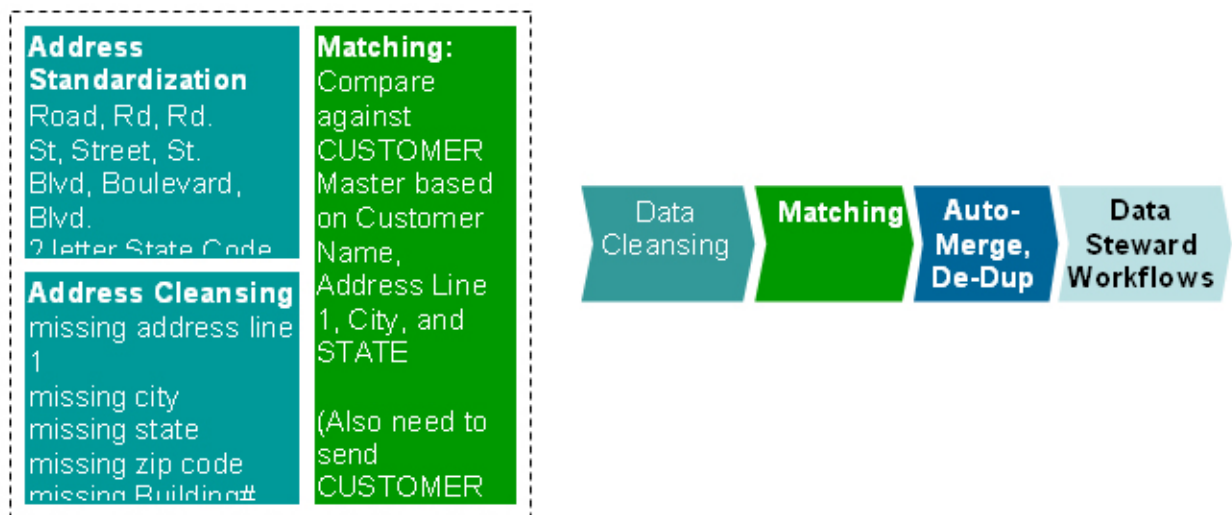


Figure 82: Trillium CDI Process Flow

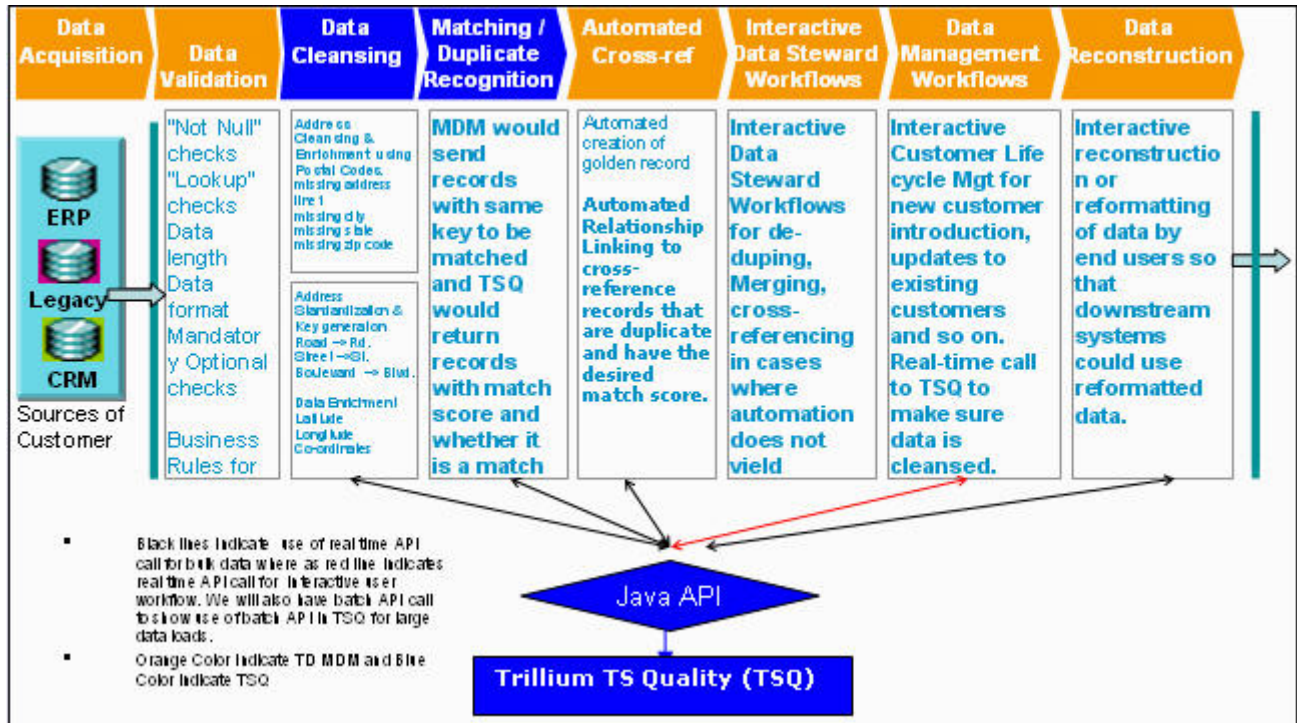


Figure 82 represents the overall Trillium CDI process flow using Teradata MDM. Teradata MDM handles data acquisition from the external source system or 3rd party legacy system, business validation, basic data type validation, other validations, data cleansing, data enrichment, matching and duplicate recognition, and publishing. Teradata MDM also handles processes like identifying the cross reference customer, interactive data steward UI process, customer management process and data reconstruction and syndication.

Data Acquisition

See "Data Acquisition" in "Sample Application CDI Process Flow".

Data Validation

See "Data Validation and Enrichment" in "Sample Application CDI Process Flow".

Data Cleansing

See "Data Cleansing" in "Sample Application CDI Process Flow". This section describes the first level of cleansing using MDM capabilities.

Trillium Data Cleansing

In this process, additional customer data cleansing capabilities are described when utilizing the Trillium Cleanser. These capabilities are provided when the Data Cleanse and Match workflow is called. The Data Cleanse and Match workflow is part of the master service and you can access the Data Cleanse and Match workflow on the MDM UI (expand Tasks bar, Sample CDI demo->Customer Data -> Enhanced Data Cleansing and Matching -> Data Cleansing and Matching).

For more details about Trillium Cleanser, refer to the Trillium Directors Guide.

The Data Cleanse and Match workflow will fetch all incoming records from MST_LEGACY_CUSTOMER and through the appropriate user interaction, the Trillium Cleanser will be called internally. Within the call, Teradata MDM passes the Complete Name, Complete Street Address, City, State, Postal code and two character Country Code to Trillium. In turn, the Trillium Cleanser returns the cleansed/corrected variables, as well as, the enriched variables like Latitudes and Longitudes. Based on these variables, MDM will update the MST_LEGACY_CUSTOMER with the 'Window Key'. The 'Window key' is the value used to identify duplicates and is used in the matching algorithm. In this workflow, the EXECUTE_TRILLIUM tag is used to connect to the Trillium Cleanser. For the cleansing function to be utilized, Trillium Cleanser configuration setup must occur. Regarding the Trillium Cleanser Configuration setup, refer to *Utility Operations section in MDM Platform Reference Guide.pdf*.

Below is an example of using the EXECUTE_TRILLIUM tag in MDM for cleansing:

```
<EXECUTE_TRILLIUM Type="Cleanse" SYS_ID="G" serverName="Cleanser"
RootTag="CustomerData/PrimaryCustomer/Address"
AssignToVar="TrilliumXMLData">
<CustomerData>
  <REQUESTTYPE>Cleanse</REQUESTTYPE>
  <PrimaryCustomer>
    <!-- Input Variables -->
    <Name>ALAN WOLFSON</Name>
    <Address>
      <!-- Input Variables -->
      <StreetLine1>25 Linnell Circle</StreetLine1>
      <StreetLine2/>
      <City>Billerica</City>
      <State>MA</State>
      <Country>US</Country>
      <ZipCode>01821</ZipCode>
    <!-- output Variables -->
    <WINDOW_KEY_01></WINDOW_KEY_01>
    <NEWADDR1></NEWADDR1>
    <PR_NAME_GENDER_01></PR_NAME_GENDER_01>
    <US_GOUT_STREET_SUFFIX></US_GOUT_STREET_SUFFIX>
    <US_CEN_RESLV_LONGITUDE></US_CEN_RESLV_LONGITUDE>
    <US_CEN_RESLV_LATITUDE></US_CEN_RESLV_LATITUDE>
    <US_CEN_RESLV_COORD_LEVEL></US_CEN_RESLV_COORD_LEVEL>
    <US_GOUT_HOUSE_NUMBER></US_GOUT_HOUSE_NUMBER>
    <US_GOUT_STREET_NAME></US_GOUT_STREET_NAME>
    <US_GOUT_POSTAL_CITY_NAME></US_GOUT_POSTAL_CITY_NAME>
    <US_GOUT_STATE_NAME></US_GOUT_STATE_NAME>
    <US_GOUT_POSTAL_CODE></US_GOUT_POSTAL_CODE>
    <US_GOUT_MATCH_LEVEL></US_GOUT_MATCH_LEVEL>
    <US_GOUT_SECONDARY_NUMBER></US_GOUT_SECONDARY_NUMBER>
    <US_GOUT_RECORD_TYPE></US_GOUT_RECORD_TYPE>
    <LEV2_MATCHED_PATTERN></LEV2_MATCHED_PATTERN>
    <NEWADDR2></NEWADDR2>
    <PR_NAME_SUFFIX_RECODED_01></PR_NAME_SUFFIX_RECODED_01>
    <PR_GIVEN_NAME1_RECODED_01></PR_GIVEN_NAME1_RECODED_01>
    <PR_SURNAME1_RECODED_01></PR_SURNAME1_RECODED_01>
  </Address>
</PrimaryCustomer>
```

```
</CustomerData>
</EXECUTE_TRILLIUM>
```

Trillium Matching/Duplicate Recognition & Automated Cross-reference

In this process we identify ‘matched’ and ‘duplicate’ records in order to enforce a single customer view. These capabilities are provided when the Data Cleanse and Match workflow is called. In this workflow, once the cleansing functions occur, the Matching portion takes place in the Trillium Matcher. Here we need to select all records and send ‘matched’ records and incoming records into the Trillium Matcher. The Trillium Matcher will identify the ‘100 percent’ closest records and return with the ‘match’ or a pattern score. Based upon the returned value a complete match or potential match can be determined. Our EXECUTE_TRILLIUM Tag expects two sets of inputs in order to identify the match record. The first set of inputs is the ‘match’ record, that is, which record are we looking to match. The second set of inputs is the master records, that is, which records are we looking to compare. Based on these inputs, the Trillium Matcher performs matching functionality and returns the input records with a pattern score against each master record. Based upon that, TMDM can identify which are duplicates or potential duplicates using our business functionality. For the Matching function to be utilized, the Trillium Matching configuration setup must occur. Regarding the Trillium Matching Configuration setup, refer *Utility Operations section in MDM Platform Reference Guide.pdf*. Based on that only Trillium returns the matched output to the caller.

During the matching process, Teradata MDM extracts all records from MST_LEGACY_CUSTOMER with CUST_STATE equal to ‘INCOMING’. If MST_CUSTOMER’s ‘Window Key’ matches the MST_LEGACY_CUSTOMER’s ‘Window Key’ then this record is considered as an exact Match. MDM then passes these MST_CUSTOMER records, as candidates, and the MST_LEGACY_CUSTOMER record as Primary records to the Trillium Matcher. Based on the return value of pattern code, MDM will identify the ‘FULL MATCH’ record or ‘Potential Duplicate’ or ‘Suspect Match’. If it is a suspect or potential match record then update the MST_LEGACY_CUSTOMER’s MATCH_ID equal to MST_CUSTOMER’s PARTY_ID value. If it does not return any ‘Suspect Match’ or ‘Potential Match’ then the current record is considered a New Customer record.

Below is an example of using the EXECUTE_TRILLIUM tag in MDM for matching:

```
<EXECUTE_TRILLIUM Type="MatchSuspect" SYS_ID="G" serverName="RMatcher"
RootTag="CustomerData/PrimaryCustomer"
AssignToVar="TrilliumXMLMatchData">
<CustomerData>
<REQUESTTYPE>RMatch</REQUESTTYPE>
<!-- Going Match - Incoming record data ?
  <PrimaryCustomer>
    <Name>John Smith</Name>
    <PartyID>1234</PartyID>
    <Address>
      <StreetLine1>170 Lexington Road</StreetLine1>
      <StreetLine2/>
      <City>Billerica</City>
      <State>MA</State>
```

```

        <Country>US</Country>
        <ZipCode>01821</ZipCode>
    </Address>
</PrimaryCustomer>
<!--Masters record data ?

    <Candidate>
        <Name>John Smith</Name>
        <PartyID>1235</PartyID>
        <Address>
            <StreetLine1>170 Lexington Road</StreetLine1>
            <StreetLine2/>
            <City>Billerica</City>
            <State>MA</State>
            <Country>US</Country>
            <ZipCode>01821</ZipCode>
        </Address>
    </Candidate>
    <Candidate>
<Name>Carl Smith</Name>
        <PartyID>1236</PartyID>
        <Address>
            <StreetLine1>170 Lexington Street</StreetLine1>
            <StreetLine2/>
            <City>Billerica</City>
            <State>MA</State>
            <Country>US</Country>
            <ZipCode>01821</ZipCode>
        </Address>
    </Candidate>
</CustomerData>
</EXECUTE_TRILLIUM>

```

In the process the following tables are involved:

- MST_LEGACY_CUSTOMER
- MST_CUSTOMER
- MST_PARTY_CUST_XREF
- MST_PARTY
- MST_PARTY_ADDRESS

Interactive Data Steward

See [“Interactive Data Steward”](#) in [“Sample Application CDI Process Flow”](#).

Manage Customer

See [“Manage Customer”](#) in [“Sample Application CDI Process Flow”](#). The Introducing New customer section utilizes the Trillium Cleanse and Match feature to identify the Duplicate records. If the record is determined to be a duplicate, then it shows with the matched records and an accompanying warning message. Based on the user selection, MDM will create a new customer.

Publishing

See [“Publishing”](#) in [“Sample Application CDI Process Flow”](#).

Data Syndication

See [“Data Syndication”](#) in [“Sample Application CDI Process Flow”](#).

APPENDIX A **Publication Services**

What's In This Appendix

This appendix provides information on Publication Services.

Topics include:

- [Introduction](#)
- [Logical Data Model](#)

Introduction

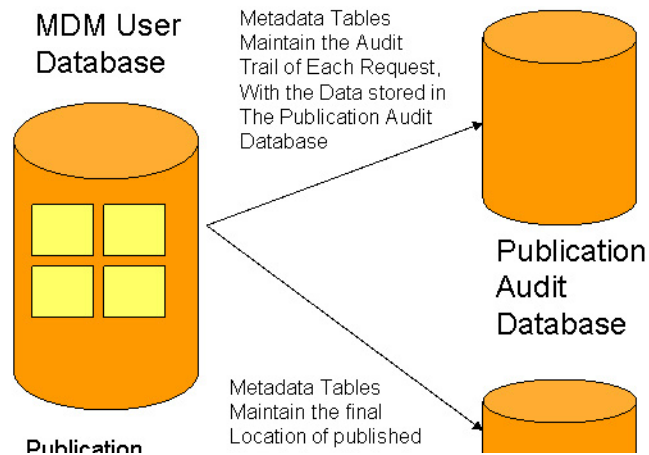
Publication Services is a new feature in MDM 2.0 that allows the Data Steward to control the usage and flow of Master Data to end users or to consuming processes or applications.

Publication Services allows any data within the scope of MDM to be published in multiple formats, to meet the needs and requirements of consuming application or processes.

Consuming applications or processes can retrieve data directly from the Publication database, or they can have data pushed to them from a JMS Provider Queue table via JMS Messaging. Additionally, data can be extracted from the database into file format, and emailed directly to business users. Along the way, an audit trail of each publication request is preserved, including (optionally) the actual data published with each request.

Figure 83: Publication Services

Publication Services Databases



To accomplish this goal, Publication Services have been integrated directly into the MDM (MDM) Workflow engine. This means that there is both a design time and a run time component to Publication Services. At design-time, the Data Steward can create workflows containing Publication Nodes. Publication Nodes are used to push MDM managed data to consuming processes or applications, or directly to business users. When MDM is running, the Data Steward can explicitly execute these workflows through the MDM user interface to effectively prepare and publish the data for a specific application, process, or business user. The MDM user interface is generally one that has been created explicitly for the Data Steward at design time, using MDM Studio, and it includes all of the user interfaces and workflows that can be used by the Data Steward to control the flow of data.

Through Publishing Services, customers can use Workflows to Publish data. Data can be published for a variety of reasons:

- Error records can be published directly to the Data Steward as part of a Data Quality workflow
- New records can be pushed to consuming applications
- Critical updates to the master data can be published to consuming applications

To support the Publishing Services framework, several new conceptual objects are going to be defined. These include:

- Publication Object
- Publication Method
- Publication Workflow Node

This section will describe these new conceptual objects, and show how they will all work together in Publication Services.

Publication Object

A Publication Object is defined as any collection of information that can be published to a downstream application, process, or end user. A Publication Object is comprised of a Publication Key, and the Publication MetaData. In the context of publishing Master Data, the Publication Meta Data specifies the composition of the data (tables and columns) that will be published to the downstream consuming application, process, or user. In the context of Teradata MDM, Publication MetaData is represented by one or more XDocuments, and their respective properties. XDocuments and Properties are used by Teradata MDM processes to denote underlying Tables and Columns respectively. The Publication Key is used to create a singular reference to this collection of data.

It should also be noted that a Publication Object can be referenced directly in a Workflow node through its Publication Key. For example, referring to a “Customer” Key will result in publishing all of the underlying data structures that have been mapped to “Customer”.

Publication Method

The Publication Method is defined as the manner in which the data will be published. There are a variety of mechanisms for actually publishing data to another application, process, or to an end user. These methods include:

- Publishing data to a separate set of Tables – this option will publish MDM controlled data into a set of database tables that reside in the Publication database.
- Publishing to a JMS Provider Queue table – this option will publish the MDM controlled data into a Teradata Queue table, such that the data can be pushed out via the Teradata JMS Provider utility. In this option, the target table will be created as Teradata Queue table through MDM publishing and data will be moved to underlying Queue table with first column QITS of datatype Timestamp (6) and rest of the columns same as the source table columns, for easy transfer to consuming applications.
- Publishing data to Excel Spreadsheet Format – in this format, the MDM controlled data will be exported into a single Microsoft 2003 compliant spreadsheet. Within this spreadsheet, there will be one worksheet per XDocument (a Publication Object may consist of more than one XDocument). In MDM 2.0, this file will be emailed to a single user, as specified in the Publication Workflow node.
- Publishing data to a text file, in Comma-Separated-Value (CSV) format – in this format, the MDM controlled data will be exported into one or more text files. Each text file contains the data – in a comma separated format – of a single XDocument. These text file(s) are emailed to a single user, as specified in the Publication Workflow node.

Publication Node

A Publication Node is a new processing node that can be inserted into any MDM workflow. The Publication Node is used the data that should be published, and the method of publication.

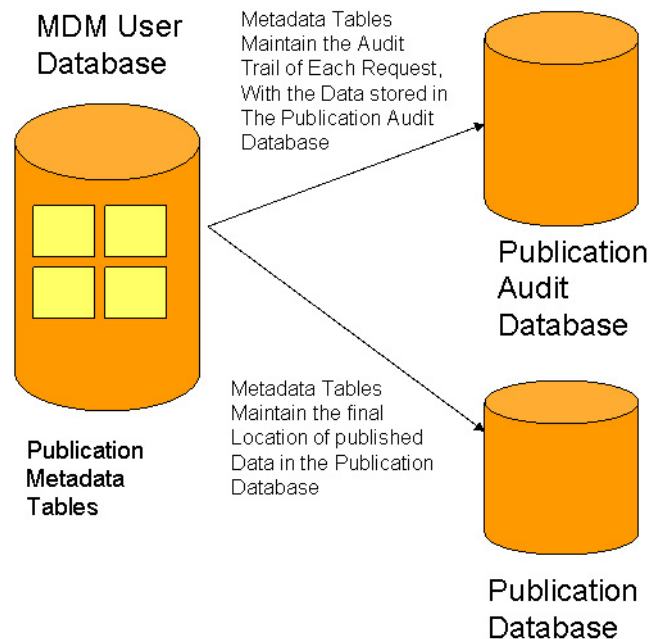
Logical Data Model

At a high level, Publication Services maintains three sets of data in the MDM databases:

- **Publication Audit Database** – this database contains snapshots of the data as it was published during publication requests.
- **Publication Database** – this database contains published data, ready for consumption by other applications and processes. Data published to Database Tables, or to JMS Provider Queues will reside in tables in this database.
- **Metadata Tables** – these metadata tables reside in the MDM User database. Among other functions, the metadata tables maintain pointers to the audit trail of every publication request, and they maintain pointers to the published data (if it was published to a database table or to a JMS Provider Queue)

Figure 84: Publication Services Databases

Publication Services Databases



During the installation process, the user is prompted to enter the names for several MDM databases, including both the Publication Audit database, and the Publication database. These are logical databases. The Publication database must be a physically separate data from the rest of the MDM databases, however, the Publication Audit database can be combined with other MDM databases (it does not have to be a distinct physical database).

The remainder of this section will describe the metadata tables, and will show how they can be used to point to the audit trail information and data contained in the Publication Audit Database, as well as how they point to data published to the Publication database.

Metadata Tables

The Publication Services metadata tables can be broken down into four sets of tables:

- Publication Object Definition tables
- Publication Object Audit Tables
- Publication Request Tables
- Publication Request Audit tables

Publication Object Definition Tables

When Publication Objects are deployed into the database, they are converted from an XML format into an internal metadata format that is stored in the database tables shown below:

Figure 85: Database Tables

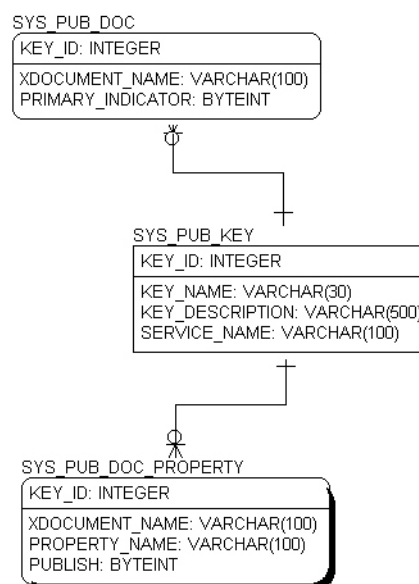


Table Description:

- **SYS_PUB_KEY** – this is the main table for the Publication Object metadata. It stores a list of all Publication Objects, and the X-Service to which they are attached. Note that the **KEY_ID** of this field is generated by the system.
- **SYS_PUB_DOC** – A Publication Object consists of one or more X-Documents. This table lists the X-Documents that comprise a Publication Object.
- **SYS_PUB_DOC_PROPERTY** – this table is reserved for future expansion. It contains a list of the fields or properties of a table that will be published. In MDM 2.0, all properties of a table are published.

Note: Publication Objects will have a **KEY_ID** generated by the system anytime they are deployed into the MDM database. This **KEY_ID** is later referenced in the audit trail

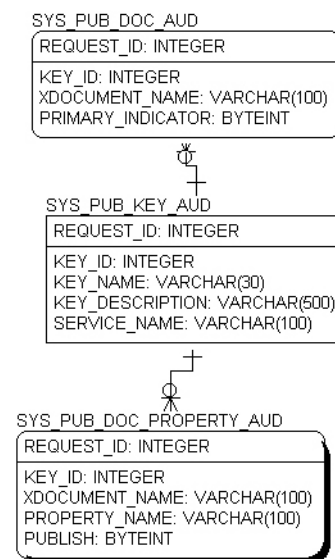
information: Whenever a publication request publishes an object, an entire copy of the publication object is copied into the audit tables, all referenced by the same KEY_ID.

When Publication Objects are re-deployed, they are assigned a new system generated key – making it a new publication object. The original definition of the Publication Object is maintained in the audit trails, but the newly deployed Publication Object will be identified as a separate object, as it will have a new key_id.

Publication Object Audit Tables

When a publication object is published in the context of a specific Publication Request, a snapshot of the Publication Object is stored in the Publication Object Audit Tables.

Figure 86: Publication Object Audit Tables



These tables are all joined by the REQUEST_ID, which is the runtime ID of the Publication Request. This ID is a system generated ID, and is created when a new Publication Request is issued.

Other than being linked by a common REQUEST_ID, these tables mirror in structure the Publication Object Definition Tables described above.

Publication Request Tables

The Publication Request Tables store the runtime definition of the Publication Request as it will be processed. These tables are joined together by a REQUEST_ID, which is generated when the request is processed.

Figure 87: Publication Request Tables

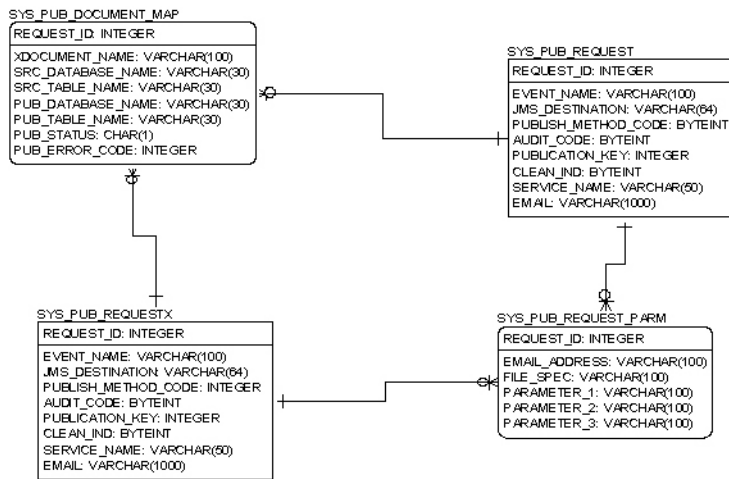


Table Description:

- **SYS_PUB_REQUEST** – this is the main publication request table. This table stores the Publication Request, which includes:
 - Publication Request (“Event”) Name – name of the publication request as it is entered in the Publication Node at design time.
 - Publication Object Key – this specifies which object is being published
 - Publication Method – the method or format in which the data will be published
 - Audit_Code – this specifies whether or not the data will be audited with the request (The request itself is always audited)
 - Service_name – name of the service to which this publication request belongs
 - Request-specific parameters, including the email address and the JMS Destination (used when publishing to a JMS Provider Queue)

The key to this table – and all of the request tables – is the REQUEST_ID, which is automatically generated when the request is processed.

- **SYS_PUB_REQUESTTX** – when a request cannot be processed entirely within the database – such as when data is being exported into an external Excel or CSV format – the request is first audited, then moved into this table for asynchronous processing by the MDM Server.
- **SYS_PUB_DOCUMENT_MAP** – this table contains a document map for each request, which will show – for each X-Document being published – what the source table was, what the publication database and table name are (if it is being published to the database), and the final status and error code (if an error occurs).

Note: This is an intermediate table – the final publication document map for a publication request can be found in the AUDIT version of this table (SYS_PUB_DOCUMENT_MAP_AUD).

Publication Request Audit Tables

The Publication Request Audit Tables store an audit copy of the publication request, after it has been processed. In addition to maintaining an audit trail, these tables also contain a full listing of the request, the outcome, and error code (if there was an error), and a mapping of where the data was published (if it was published in the database) and where the audit copy of the data resides. These tables are joined together via the REQUEST_ID.

Figure 88: Publication Request Audit Tables

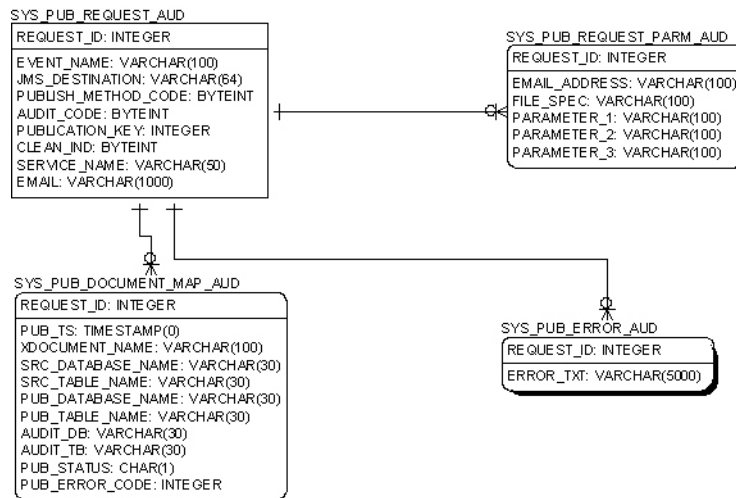


Table Description:

- **SYS_PUB_REQUEST_AUD** – this table contains audit copies of the publication requests as they were processed. It is a mirror image of SYS_PUB_REQUEST.
- **SYS_PUB_ERROR_AUD** – this table contains error messages if there was an error during the publishing of data (normally there will not be any errors in this table)
- **SYS_PUB_REQUEST_PARM_AUD** – this table is not used in MDM 2.0, but is reserved for future use.
- **SYS_PUB_DOCUMENT_MAP_AUD** – this table is the most important table in the audit trail, as it maintains for each Publication Request detailed information about:
 - When the data was published
 - Where it was published (if it was published in the database)
 - Where the source data for this table was drawn from
 - If an audit copy of the data was preserved, this table will point to that data for each X-document published
 - Publication Status – S=Success, F=Failure
 - Error Code – if the publication failed, this column will contain an explanation of why it failed

Following an Audit Trail of a Publication Request

This section will show how to follow an audit trail of a Publication Request, by following the data in the SYS_PUB_REQUEST_AUD and SYS_PUB_DOCUMENT_MAP_AUD tables. For this exercise, we will audit a Publication Request that was published as follows:

Publication Object contained 2 X-Documents: Customer and Order

Publication Method is set to publish the records to the Publication database

Audit is set, so that both the Publication Request and the data being published are audited.

The **REQUEST_ID** for this table when it was published is “10”.

Here’s what the SYS_PUB_REQUEST_AUD table contained for this record:

- REQUEST_ID: 10
- EVENT_NAME: “test1:PublishCustomerOrders”. This field will always contain a combination of the <workflowname>:<publication node name>
- Publish Method Code:
 - 0 = Publish to the Database. Other possible codes are:
 - 1 = Publish to a JMS Provider Queue
 - 5 = Publish to a Comma Separated Values (CSV) text file
 - 6 = Publish to an Excel 2003 compliant spreadsheet
- Audit Code: 1 (1 = audit the data being published, 0 = no audit of the data (request-only))
- Publication Key: 4 --- in this case, when the Publication Objects were deployed, the “Customer Order” Publication Object was assigned an ID of 4.
- Clean_Ind – this field is not used
- Service Name: this will contain the name of the X-Service, in this case, “Customer”
- Email: This field will contain an email address if the user is publishing via Excel or CSV formats. This will specify to whom the file(s) were emailed. In this example, this field is empty.
- JMS_DESTINATION: This field will contain the name of a JMS Queue (used only when publishing to a JMS Provider Queue). In this example, this field is empty.

The SYS_PUB_DOCUMENT_MAP_AUD table contains a detailed listing of exactly where the Published data originated, and where it resides, and it contains a listing for each X-Document being published. In this example, the “Customer Order” Publication Object contains two X-Documents, so there are two rows of data for this request. Here’s what this table contains for this example (note that these results are retrieved by using the REQUEST_ID (“10”) above):

Row 1: “Order” X-Document

- REQUEST_ID: 10
- PUB_TS: timestamp specifying when the data was published
- XDOCUMENT_NAME: Order (this is the name of the X-Document)
- SRC_DATABASE_NAME: mdm_mst (in this example, the Order data was being drawn from the MASTER database of MDM).

- SRC_TABLE_NAME: order_pad. These two fields combined mean that the source table for this Publication Request is MDM_MST.ORDER_PAD
- PUB_DATABASE_NAME: mdm_pub. During installation, this was the name given for the MDM Publication Database. All data published in a database-format is published to this database.
- PUB_TABLE_NAME: WeeklyOrders. This value was specified in the Publication Node
- AUDIT_DB: mdm_pub_audit. This value was specified during installation as the name of the Publication Audit Database
- AUDIT_TB: PUB_10_1. The name of this table includes the ID of the request, and the sequence of the X-Document in the Publication Object. These two fields combined mean that an audit copy of the data being published can be found in MDM_PUB_AUDIT.PUB_10_1. It is useful to know how to derive this, as there will likely be a case in which a user needs to validate exactly what data was published with a given request.
- PUB_STATUS: "S" (S=Success, F=Failure)
- PUB_ERROR_CODE: empty, as there was no error

Row 2: "Customer" X-Document

- REQUEST_ID: 10
- PUB_TS: timestamp specifying when the data was published
- XDOCUMENT_NAME: Customer (this is the name of the X-Document)
- SRC_DATABASE_NAME: mdm_mst (in this example, the Customer data was being drawn from the MASTER database of MDM.
- SRC_TABLE_NAME: CUSTOMER. These two fields combined mean that the source table for this Publication Request is MDM_MST.CUSTOMER
- PUB_DATABASE_NAME: mdm_pub. During installation, this was the name given for the MDM Publication Database. All data published in a database-format is published to this database.
- PUB_TABLE_NAME: WeeklyCustomers. This value was specified in the Publication Node
- AUDIT_DB: mdm_pub_audit. This value was specified during installation as the name of the Publication Audit Database
- AUDIT_TB: PUB_10_2. The name of this table includes the ID of the request, and the sequence of the X-Document in the Publication Object. These two fields combined mean that an audit copy of the data being published can be found in MDM_PUB_AUDIT.PUB_10_2. It is useful to know how to derive this, as there will likely be a case in which a user needs to validate exactly what data was published with a given request.
- PUB_STATUS: "S" (S=Success, F=Failure)
- PUB_ERROR_CODE: empty, as there was no error

One general comment about the Pub Audit tables: The names of the tables – as shown above – are always PUB_<Request_id>_<Sequence_id> (as in PUB_10_2), however each table

also contains a COMMENT, which in turn contains the name of the source table as well as the Timestamp of the publishing event. This is another way to retrieve this information, although looking at SYS_PUB_DOCUMENT_MAP_AUD provides this information as well, as shown above.

The comments can be found in the DBC.Tables for each table.

APPENDIX B Configuration of Trillium Client

What's In This Appendix

This appendix provides information on configuring Trillium client.

Topics include:

- [Trillium Client Setup in MDM](#)

Trillium Client Setup in MDM

Execute the following steps to setup Trillium client in MDM:

If the Trillium server and MDM server are running on the same system, execute from step 3 else if Trillium server and MDM server are running on different systems, execute from step 1 and skip step 3.

- 1 Configure Trillium client on MDM server system.

Copy the entire bin folder from `<Trillium_Installed_Location>\tsq11r0s\Software\bin` to the MDM client machine and set the TRILLDIRPORT and TRILLDIRADDR environment variables.

For example, on windows, the bin folder (`c:\Trillium\tsq11r0s\Software\bin`) will have all the required files and if the remote system Trillium IP address is 255.255.255.255 and Trillium port number is 4444, then the environment variables setting would be as shown below:

```
set PATH = c:\Trillium\tsq11r0s\Software\bin;%PATH%.
```

```
set TRILLDIRADDR = 255.255.255.255
```

```
set TRILLDIRPORT = 4444
```

Note: The static ports set for Cleanser and Matcher should be open on the firewall.

Refer to Trillium Director manual for detailed information on configuring Trillium client.

- 2 On Trillium client side, environment variable TRILLCONFIG should be referring the TrilXML.cfg file. Because Teradata MDM supports only XML based cleanse and Match functionality of Trillium.

Copy the file *TrilXML.cfg* from location

`<TrilliumSoftware_Install_Location>\tsq11r0s\<project_name>\settings\TrilXML.cfg` from the server's project setting location and paste into client.

- 3 On Trillium Server side, environment variable TRILLCONFIG should be referring the TrilXML.cfg file. Because Teradata MDM supports only XML based cleanse and Match functionality of Trillium.


```
set TRILLCONFIG=..\TrilliumSoftwareHH\tsq11r0s\<project  
_name>\settings\TrilXML.cfg
```

Note: The above Trillium configuration steps are certified on the Windows operating system only.

- 4 If you want to use Teradata Sample application using Trillium cleanse and match, then execute the below listed steps else you can skip the below steps:

If existing and specific project's TrilXML file does not have below listed variables then include these variables in to the files.

For Cleanser:

[XML<CustomerData/PrimaryCustomer/Address>] ## Map data for cleanse/enrich

LINE_01 = ../Name ###Replace

(Existing Variables)

StreetAddress = StreetLine1
City = City
State = State
Country= Country
PostalCode = ZipCode
ObjectType = XMLAddress

[XMLAddress]

../Name = LINE_01 ###Replace

###Existing Variables

StreetLine1 = dr_address
City = dr_city_name
State = dr_region_name
Country= dr_country_name

###Add these variables

WINDOW_KEY_01 = WINDOW_KEY_01

NEWADDRL1 = NEWADDRL1
NEWADDRL2 = NEWADDRL2
NEWADDRL3 = NEWADDRL3
NEWADDRL4 = NEWADDRL4
NEWADDRL5 = NEWADDRL5
NEWADDRL6 = NEWADDRL6
NEWADDRL7 = NEWADDRL7
US_GOUT_STREET_SUFFIX = US_GOUT_STREET_SUFFIX
US_CEN_RESLV_LONGITUDE = US_CEN_RESLV_LONGITUDE
US_CEN_RESLV_LATITUDE = US_CEN_RESLV_LATITUDE
US_CEN_RESLV_COORD_LEVEL = US_CEN_RESLV_COORD_LEVEL
US_GOUT_HOUSE_NUMBER = US_GOUT_HOUSE_NUMBER
US_GOUT_STREET_NAME = US_GOUT_STREET_NAME
US_GOUT_POSTAL_CITY_NAME = US_GOUT_POSTAL_CITY_NAME
US_GOUT_STATE_NAME = US_GOUT_STATE_NAME
US_GOUT_POSTAL_CODE = US_GOUT_POSTAL_CODE
US_GOUT_MATCH_LEVEL = US_GOUT_MATCH_LEVEL
US_GOUT_SECONDARY_NUMBER = US_GOUT_SECONDARY_NUMBER
US_GOUT_RECORD_TYPE = US_GOUT_RECORD_TYPE
LEV2_MATCHED_PATTERN = LEV2_MATCHED_PATTERN
PR_NAME_GENDER_01 = PR_NAME_GENDER_01
PR_GIVEN_NAME1_RECODED_01 = PR_GIVEN_NAME1_RECODED_01
PR_SURNAME1_RECODED_01 = PR_SURNAME1_RECODED_01
PR_NAME_SUFFIX_RECODED_01 = PR_NAME_SUFFIX_RECODED_01

For Matcher:

```
[XML<CustomerData/PrimaryCustomer>]          ## Map data for reference match

###Existing Variables

BusinessName = Name
StreetAddress = Address/StreetLine1
City          = Address/City
State         = Address/State
Country       = Address/Country
PostalCode    = Address/ZipCode
ObjectType    = XMLAddress
```

- 5 On the MDM side, in the *bcmenv* batch file located at
`<MDM_Install_Directory>\custom\<SampleApplicationName>\bin` need to set and add
the Trillium files

```
set TRILLIUM_CLASSPATH=Trillium Software <installed location >
\tsq11r0s\Software\bin;

set
CLASSPATH=%CLASSPATH%;%POI_JARS%;%JAVA_HOME%\lib\tools.jar;%TRILLIUM_CLASSPATH%
```
- 6 For Sample Application with Trillium base, execute the below steps at server machine.
Navigate to specific projects (*director_proj*)/settings.
 - a Open the `..\TrilliumSoftware\tsq11r0s\director_proj\settings\DIRCleanser.xml` and for
US country, modify the DATAREC tag as below.

```
<DATAREC NAME="usDatarec" ENABLED="y" TRACE="">
```

- b Open the `..\TrilliumSoftware\tsq11r0s\director_proj\ddl\ustsqfrmt.ddx` and locate the
field "LINE_01" and modify the REDEFINE tag as below.

```
<REDEFINE>N</REDEFINE>
```

- c Open the `..\TrilliumSoftware\tsq11r0s\director_proj\ settings\DIRCleanser.xml` and
locate winkey which is for non-country specific and modify the value as below:

```
<WINKEY NAME="winkey" ENABLED="n" TRACE="">
```

- d In the above file (*DIRCleanser.xml*) locate *uswinKey* which is for US country and
modify value as below:

```
<WINKEY NAME="usWinkey" ENABLED="y" TRACE="">
```

- e Open the `..\TrilliumSoftware\tsq11r0s\director_proj\ settings\uspmatch.stx` and locate
tag `</PROCESS_SETTINGS>` and next to that add the below lines. In highlighted
section, provide actual path information to enrichment.

```
<CENSUS_SETTINGS>
<ARGUMENTS>
<CENSUS_INTERPOLATED_LEVEL1_DATA_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USCINDEX1.tbl</CENSUS_IN
TERPOLATED_LEVEL1_DATA_FILE_NAME>
<CENSUS_INTERPOLATED_LEVEL2_DATA_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USCINDEX2.tbl</CENSUS_IN
TERPOLATED_LEVEL2_DATA_FILE_NAME>
<CENSUS_INTERPOLATED_BASE_DATA1_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USCBASE1.tbl</CENSUS_INT
ERPOLATED_BASE_DATA1_FILE_NAME>
<CENSUS_INTERPOLATED_BASE_DATA2_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USCBASE2.tbl</CENSUS_INT
ERPOLATED_BASE_DATA2_FILE_NAME>
<CENSUS_INTERPOLATED_BASE_DATA3_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USCBASE3.tbl</CENSUS_INT
ERPOLATED_BASE_DATA3_FILE_NAME>
<CENSUS_INTERPOLATED_BASE_DATA4_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USCBASE4.tbl</CENSUS_INT
ERPOLATED_BASE_DATA4_FILE_NAME>
<CENSUS_INTERPOLATED_MSA_DATA_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USMSA.tbl</CENSUS_INTERPOL
ATED_MSA_DATA_FILE_NAME>
<CENSUS_INTERPOLATED_NAME_DATA_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USNAME.tbl</CENSUS_INTER
POLATED_NAME_DATA_FILE_NAME>
<CENSUS_XREF_LEVEL1_DATA_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USXINDEX1.tbl</CENSUS_XREF_LEVEL
1_DATA_FILE_NAME>
<CENSUS_XREF_BASE_DATA_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USXBASE.tbl</CENSUS_XREF_BASE_DA
TA_FILE_NAME>
<CENSUS_GACI_DATA_FILE_NAME>..\TrilliumSoftware\HH\tsq11r0s\tables\census_tables\USGACI.tbl</CENSUS_GACI_DATA_FILE_NAME>
</ARGUMENTS>
</CENSUS_SETTINGS>
```

- f Open the ..\TrilliumSoftware\tsq11r0s\director_proj\settings\usdrrules.sto and locate rule name "copy_all" and add below lines for "PR_NAME_FORM_01 = '1'"

```
move PR_SURNAME1_ORIGINAL_01, NEWADDR17;
append PR_GIVEN_NAME1_ORIGINAL_01, NEWADDR17;
move PR_SURNAME1_ORIGINAL_01, NEWADDR18;
append PR_GIVEN_NAME1_ORIGINAL_01, NEWADDR18;
```

- g Open the ..\TrilliumSoftware\tsq11r0s\director_proj\settings\usdrrules.sto and locate rule name "matched_box_rr" and add below lines:

```
move PR_SURNAME1_ORIGINAL_01, NEWADDR17;
append PR_GIVEN_NAME1_ORIGINAL_01, NEWADDR17;
move PR_SURNAME1_ORIGINAL_01, NEWADDR18;
append PR_GIVEN_NAME1_ORIGINAL_01, NEWADDR18;
```

- h Open the ..\TrilliumSoftware\tsq11r0s\director_proj\settings\usdrrules.sto and locate rule name "matched_routine_address" and add below lines

```
move US_GOUT_DELIVERY_ADDRESS NEWADDR15;
append_pack ",", NEWADDR15;
append US_GOUT_POSTAL_CITY_NAME NEWADDR15;
append US_GOUT_STATE_NAME NEWADDR16;
append_pack ",", NEWADDR16;
append US_GOUT_POSTAL_CODE[1:5] NEWADDR16;
append_pack "-", NEWADDR16;
append_pack US_GOUT_POSTAL_CODE[6:4] NEWADDR16;
```

- 7 Re-start the Director, Cleanser and Matcher.

Note: (Tested only on Windows) and all the above setup will work for remote access.

Based on the configuration of TrilXML.cfg file and input format of MDM EXECUTE_TRILLIUM tag, EXECUTE_TRILLIUM tag will return the output variables for Cleanser and Matcher. TrilXML configuration file and input/output format will depend on the projects available in the Trillium server side. For more detailed information about configuration of TrilXML.cfg, refer the Trillium Director Guide and for more detailed information about EXECUTE_TRILLIUM tag, refer *Section Services Reference in MDM Platform Reference Guide*.

Note: The above configuration supports only Trillium version 11, but EXECUTE_TRILLIUM tag supports Trillium version 11 and 12.

APPENDIX C MDM Custom Web Services

What's In This Appendix

This appendix provides information on MDM custom Web services.

Topics include:

- [Introduction](#)

Introduction

Sample Application has two type of web service support: Incoming Web Services (Teradata MDM Web Service) and Outgoing Web Services (Teradata MDM Web Service and Third Party Web Service).

Incoming Teradata MDM Web Service

SampleApplication uses a set of operations to show the examples of Web services. Sample Application has a rule common entity support that is CRUD (Create, Read, Update Delete) method. The CRUD method can perform five operations:

- Addition - Creation of new record
- Modification
- Deletion
- Mass modification
- Get - Query the specific entity

SampleApplication has a workflow called Core Service to use CRUD and Mass Update operation. The Core Service workflow uses customWebService.xml rule that contains five methods which describe CRUD and Mass Update operation as listed below:

- addDocumentDetails
- getDocumentDetails
- modifyDocumentDetails
- massUpdateDetails
- delDocumentDetails

Custom web service WSDL uses authentication to provide security, so only MDM user can access the Custom web service. The Workflow file and Rule file are available at:

```
<MDM_Install_Directory>\custom\sampleApplication\cfg\xservice\sampleApplication\workflows\webServices\ customWebService_WF.xml
```

```
<MDM_Install_Directory>\custom\sampleApplication\cfg\xservice\sampleApplication\rules\WebService\ customWebService.xml
```

Installation and Setup Instructions for MDM Web Service

For enabling MDM Web services in custom application, perform the steps listed in *Installation and Setup Instructions given in Web Services Implementation chapter of MDM Platform Studio User Guide.pdf*.

To Enable Custom Web Service in SampleApplication

Perform the following steps for enabling custom web service in sampleapplication:

- 1 Install sample application and open MDM Studio.
- 2 Copy the BCM_MASTER.aar file from
<MDM_Install_Directory>\custom\sampleApplication\testdata\WSDL\customWebServices to <MDM_Install_Directory>\web\mdmclient\WEB-INF\services
- 3 On the MDM Studio, in the **Project Navigator** pane, add a new group under **Web Services**.
- 4 Right click the new group and select Insert WSDL.
- 5 Select Loading Mechanism as "File Based" and click OK.
- 6 Provide the path to the BCM_MASTER.wsdl from
<MDM_Install_Directory>\custom\sampleApplication\testdata\WSDL\customWebServices
- 7 Right click imported WSDL in Studio and Select Activate (MDM need to be restarted if it is already running).
- 8 Clear cache and temp folder for web logic server and Start BEA web logic server.

For detailed steps, refer to *Web Services Implementation chapter of MDM Platform Studio User Guide.pdf*.

Sample Web Service Function in MDM

Custom Web Service rule expects the below listed format and gives the response according to this definition.

Example:

```
<DEFINE_METHOD Name="addDocumentDetails" Access="public">
  <API_DOC>
    <INPUT>
      <REQUEST Name="addDocumentDetails"
DocumentName="LEGACY_CUSTOMER"
      ServiceName="BCM_MASTER" OrderIndicator="true" Any="true">
    <ATTRIBUTE OrderIndicator="true">
```

```

        <ATTRIBUTE Name="CUST_ID" Value="1" Repeatable="true"
        Optional="true"/>
    </ATTRIBUTE>
</REQUEST>
</INPUT>
<OUTPUT>
    <ON_SUCCESS>
        <RESPONSES Status="Success">
            <RESPONSE Status="Success" OrderIndicator="true">
                <ANY Repeatable="true" Optional="true" Any="true"/>
            </RESPONSE>
        </RESPONSES>
    </ON_SUCCESS>
</OUTPUT>
</API_DOC>
<RULE>
    <ACTION>
        .....
        .....
        .....
    </ACTION>
</RULE>
</DEFINE_METHOD>

```

Request Format

All the rules expect some kind of request format. So to achieve that format sampleApplication uses Core service workflow where you can user select the operation to perform on selected document and provide some inputs based on operations.

By using the input to that workflow, the desired request format for a selected operation is formed, call the WSDL node and pass the constructed request to WSDL node.

Adding a record to a document

METHOD NAME: addDocumentDetails

Example: - Request Format to invoke the rule

```

<bcm:addDocumentDetails xmlns:bcm="http://www.teradata.com/
BCM_MASTER"

    DocumentName="ADDRESS" ServiceName="BCM_MASTER">
    <ATTRIBUTE>
    <ATTRIBUTE Value="1" Name="Address_Id"/>

```



```
<ATTRIBUTE Value="1" Name="Address_Type_Cd"/>
<ATTRIBUTE Value="UI" Name="SOURCE"/>
<ATTRIBUTE Value="ACTIVE" Name="ENTITY_STATE"/>
</ATTRIBUTE>
</bcm:addDocumentDetails>
```

Send a request to WSDL node under **WSDL Transform Input** tab as in [Figure 89](#). WSDL node uses "WsdparametersIn" variable as an input parameter.

Figure 89: WSDL Transform Input

The screenshot shows the 'Node Add_WSDL' dialog box. The 'Name' field is 'Add_WSDL'. The 'Description' field is empty. The 'Node Input Var' and 'Node Output' fields are empty, each with a 'Type' dropdown. The 'TimeOut' field is '0'. The 'Exception Output' field is empty, with a 'Type' dropdown. The 'Binding Url' field is empty. The 'Service' dropdown is 'BCM_MASTER'. The 'Operations' dropdown is 'addDocumentDetails'. Below these fields is a text area containing '<html>null</html>'. The 'WSDL Transform Header' tab is selected, showing an XML schema definition for 'addDocumentDetails'. The 'WSDL Transform Input' tab is also visible, showing an action that assigns 'WsdparametersIn' to a variable. The 'Outline' and 'Source' tabs are visible at the bottom. The 'OK' and 'Cancel' buttons are at the bottom right.

Response

```
<ns0:addDocumentDetailsResponse
  xmlns:ns0="http://www.teradata.com/BCM_MASTER"
  Status="Success">
  <RESPONSE Status="Success">
    <RESPONSE>
      <CUST_ID Value="1"/>
      <ORG_ID Value="1"/>
      <MESSAGE Value="RECORD ADDED SUCCESSFULLY"/>
    </RESPONSE>
  </RESPONSE>
</ns0:addDocumentDetailsResponse>
```

The [Figure 90](#) displays the response under **WSDL transform Output** tab. WSDL node uses "WsdParametersOut" variable as an output parameter.

Figure 90: WSDL Transform Output

Node Add_WSDL

Name:

Description:

Node Input Var: Type:

Node Output: Type:

TimeOut:

Exception Output: Type:

Binding Url:

Service: Operations:

WSDL Transform Header | WSDL Transform Input | Transport Level Endorsement | **WSDL Transform Output**

parameters

```
<xsd:element xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="addDocumentDetailsResponse">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="RESPONSE">
        <xsd:complexType>
```

```
<action>
  <PRINTLN Var="WsdparametersOut"/>
  <IF_TEST Test="$WsdparametersOut/*(name() = 'faultcode')">
    <THEN>
      <TO_DOCVAR AssignToVar="paramOut">
        <TEMP>
          <RESPONSE>
            <RESPONSE>
```

Outline | Source

OK Cancel

Getting a record from a document

METHOD NAME: getDocumentDetails

Example: Request Format to invoke the rule

```
<bcm:getDocumentDetails xmlns:bcm="http://www.teradata.com/
BCM_MASTER"
```

```
DocumentName="ADDRESS" ServiceName="BCM_MASTER">
```

```
<ATTRIBUTE>
```

```
<ATTRIBUTE Value="1" Name="Address_Id"/>
```

```
</ATTRIBUTE>
</bcm:getDocumentDetails>
```

Response

```
<ns0:getDocumentDetailsResponse
  xmlns:ns0="http://www.teradata.com/BCM_MASTER"
  Status="Success">
  <RESPONSE Status="Success">
    <RESPONSE>
      <ADDRESS ExistingDocument="yes">
        <Address_Id Value="1"/>
        <Address_Type_Cd Value="MAILING"/>
        <ENTITY_STATE Value="ACTIVE"/>
        <SOURCE Value="BackEnd"/>
      </ADDRESS>
    </RESPONSE>
  </RESPONSE>
</ns0:getDocumentDetailsResponse>
```

Updating a record in a document

METHOD NAME: modifyDocumentDetails

Example: - Request Format to invoke the rule

This request has two types of tags:

DOCUMENT_CONTEXT - This tag has the columns for which you need to modify the record.

UPDATE_PROPERTIES - This tag has the columns which you need to modify.

```
<bcm:modifyDocumentDetails xmlns:bcm="http://www.teradata.com/
BCM_MASTER"
  DocumentName="ADDRESS" ServiceName="BCM_MASTER">
  <DOCUMENT_CONTEXT>
    <DOCUMENT_CONTEXT Value="1" Name="Address_Id"/>
  </DOCUMENT_CONTEXT>
  <UPDATE_PROPERTIES>
    <UPDATE_PROPERTIES Value="MAILING" Name="Address_Type_Cd"/>
    <UPDATE_PROPERTIES Value="ACTIVE" Name="ENTITY_STATE"/>
    <UPDATE_PROPERTIES Value="BACKEND" Name="SOURCE"/>
  </UPDATE_PROPERTIES>
</bcm:modifyDocumentDetails>
```

Response

```
<ns0: modifyDocumentDetailsResponse
  xmlns:ns0="http://www.teradata.com/BCM_MASTER"
  Status="Success">
  <RESPONSE Status="Success">
    <RESPONSE>
      <MESSAGE Value="RECORD MODIFIED SUCCESSFULLY"/>
    </RESPONSE>
  </RESPONSE>
</ns0: modifyDocumentDetailsResponse >
```

Mass update in a document

METHOD NAME: massUpdateDetails

Example: - Request Format to invoke the rule

This request has two types of tags:

DOCUMENT_CONTEXT - This tag has the columns for which we need to modify the record.

UPDATE_PROPERTIES - This tag has the columns which we need to modify.

```
<bcm:massUpdateDetails xmlns:bcm="http://www.teradata.com/
BCM_MASTER"
  DocumentName="ADDRESS" ServiceName="BCM_MASTER">
  <DOCUMENT_CONTEXT>
    <OR>
      <AND>
        <Address_Id Value="61"/>
        <Address_Type_Cd Value="MAILING"/>
      </AND>
      <AND>
        <Address_Id Value="101"/>
        <Address_Type_Cd Value="MAILING"/>
      </AND>
      <AND>
        <Address_Id Value="19"/>
        <Address_Type_Cd Value="MAILING"/>
      </AND>
    </OR>
  </DOCUMENT_CONTEXT>
  <UPDATE_PROPERTIES>
    <SOURCE Value="UI"/>
```

```
<CREATED_BY Value="USR_1"/>
</UPDATE_PROPERTIES>
</bcm:massUpdateDetails>
```

Response

```
<ns0: massUpdateDetailsResponse
  xmlns:ns0="http://www.teradata.com/BCM_MASTER"
  Status="Success">
  <RESPONSE Status="Success">
    <RESPONSE>
      <MESSAGE Value="MASS UPDATED SUCCESSFULL"/>
    </RESPONSE>
  </RESPONSE>
</ns0: massUpdateDetailsResponse>
```

Deleting a record from a document

METHOD NAME: delDocumentDetails

Example: - Request Format to invoke the rule

```
<bcm:delDocumentDetails xmlns:bcm="http://www.teradata.com/
BCM_MASTER"
  DocumentName="ADDRESS" ServiceName="BCM_MASTER">
  <ATTRIBUTE>
    <ATTRIBUTE Value="1" Name="Address_Id"/>
    <ATTRIBUTE Value="MAILING" Name="Address_Type_Cd"/>
  </ATTRIBUTE>
</bcm:delDocumentDetails>
```

Response

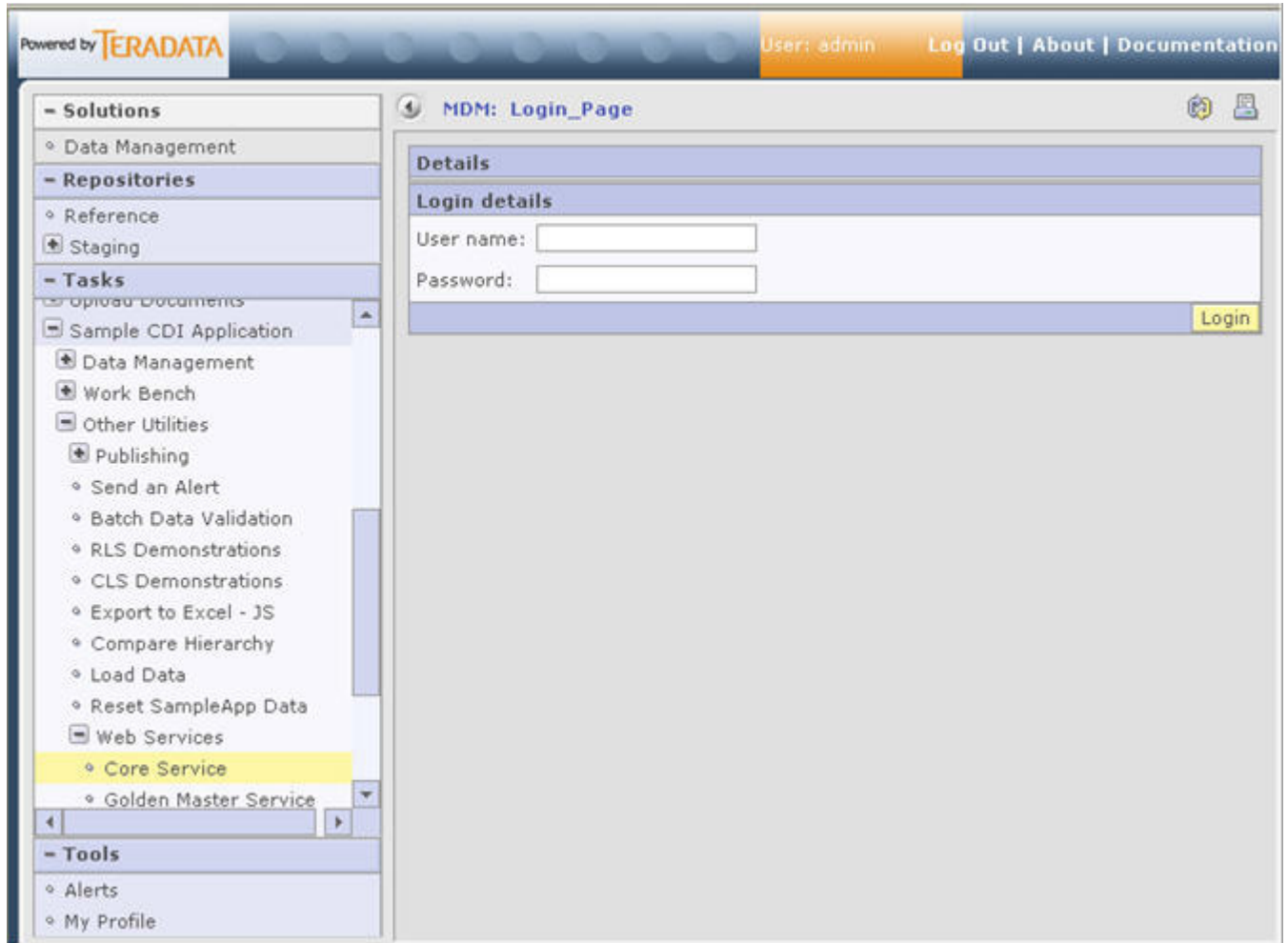
```
<ns0:delDocumentDetailsResponse
  xmlns:ns0="http://www.teradata.com/BCM_MASTER" Status="Success">
  <RESPONSE Status="Success">
    <RESPONSE>
      <Address_Id Value="1"/>
      <Address_Type_Cd Value="MAILING"/>
      <MESSAGE Value="RECORD DELETED SUCCESSFULLY. "/>
    </RESPONSE>
  </RESPONSE>
</ns0:delDocumentDetailsResponse>
```

Core Service Workflow

Perform the following steps to use Core service workflow:

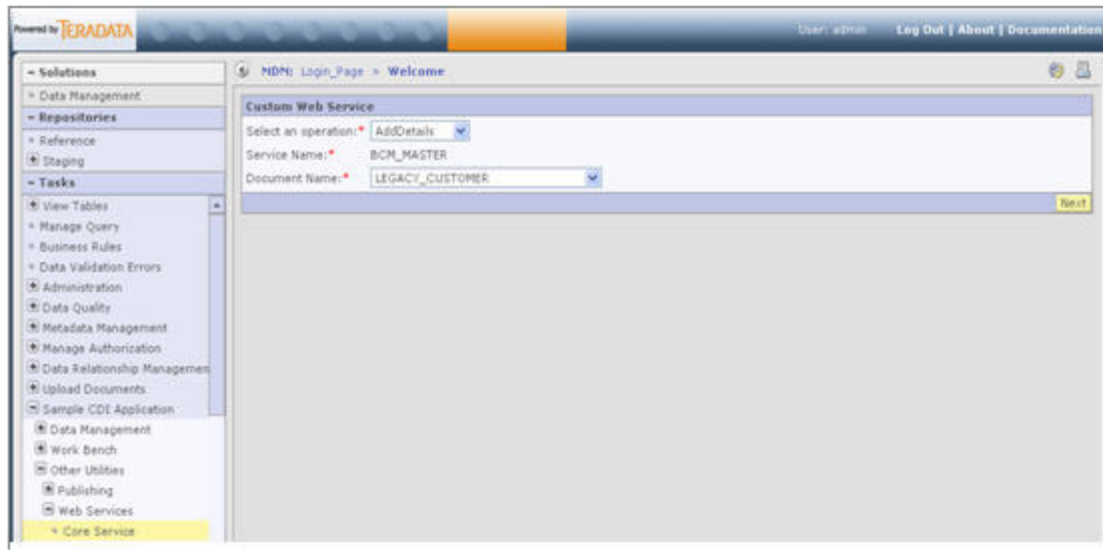
- 1 On the MDM UI, in the Tasks pane, expand **Sample CDI Application**, expand **Other utilities**, expand **Web Services** and click **Core Service** as in [Figure 91](#).

Figure 91: MDM_Login_Page



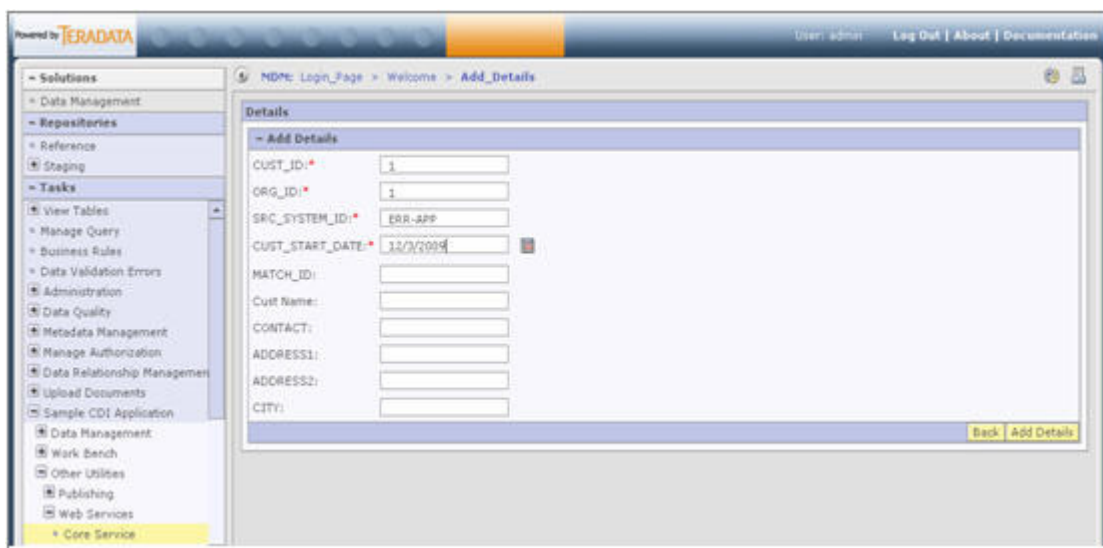
- 2 On the **Login** page ([Figure 91](#)), enter the MDM username and password and click **Login**.
Only the registered MDM users can access the custom web service, if user name or password will not be registered the user can't navigate further on the workflow.
The **Welcome** page ([Figure 92](#)) is displayed.

Figure 92: Welcome



- 3 On the **Welcome** page (Figure 92) select the operation and document name for the BCM_MASTER service from the respective drop-downs and click **Next**.
The **Add Details** page (Figure 93) is displayed.

Figure 93: Add Details



- 4 On the **Add Details** page (Figure 93), enter the required details and click **Add Details**.
Message: The record added successfully is displayed.

To Get Details, perform the following steps:

- 1 On the **Welcome** page (Figure 92) select the operation as Get Details and document name as Legacy_Customer from the respective drop-downs and click **Next**.
The **Get Details** page (Figure 94) is displayed.

Figure 94: Get Details

Powered by **TERADATA**

User: admin Log Out | About | Documentation

MDM: Login_Page > Welcome > Get_Details

Details

Get details

CUST_ID: 1

ORG_ID: 1

Back Search

Result

Search Result

CUST_ID	ORG_ID	CUST_START_DATE	SOURCE	SRC_SYSTEM_ID
1	1	12/03/2009 00:00:00.000	Backend	ERR-APP

To Modify Details, perform the following steps:

- 1 On the **Welcome** page (Figure 92) select the operation as Modify Details and document name as Address from the respective drop-downs and click **Next**.

The **Details** page (Figure 95) is displayed.

Figure 95: Details

Powered by **TERADATA**

User: admin Log Out | About | Documentation

MDM: Login_Page > Welcome > Modify_Details

Details

Search Result

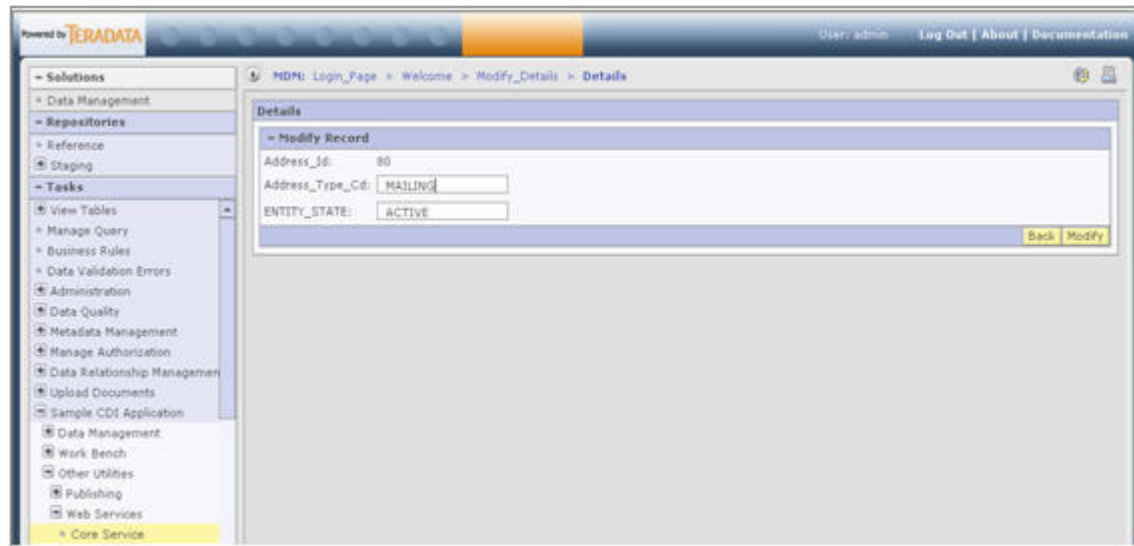
Address_Id	Address_Type_Cd	ENTITY_STATE
80	MAIL	ACTIVE
40	MAILING	ACTIVE
19	MAILING	ACTIVE
61	MAILING	ACTIVE
97	MAILING	ACTIVE
101	MAILING	ACTIVE
38	MAILING	ACTIVE
59	MAILING	ACTIVE
15	MAILING	ACTIVE
120	MAILING	ACTIVE
57	MAILING	ACTIVE
99	MAILING	ACTIVE
34	MAILING	ACTIVE
78	MAILING	ACTIVE
36	MAILING	ACTIVE
17	MAILING	ACTIVE
32	MAILING	ACTIVE
118	MAILING	ACTIVE
55	MAILING	ACTIVE

Back Modify

- 2 On the **Details** page (Figure 95), edit any column and click on **Modify**.

The **Details-Modify Record** page (Figure 96) is displayed.

Figure 96: Modify Details



- 3 On the **Details-Modify Record** page (Figure 96), click **Modify**.

The record will be modified and the success message will be displayed on the same screen.

To Delete Details, perform the following steps:

- 1 On the **Welcome** page (Figure 92) select the operation as Delete Details and document name as Address from the respective drop-downs and click **Next**.

The **Details** page (Figure 95) is displayed.

- 2 On the **Details** page (Figure 95), select the required record and click **Delete**.

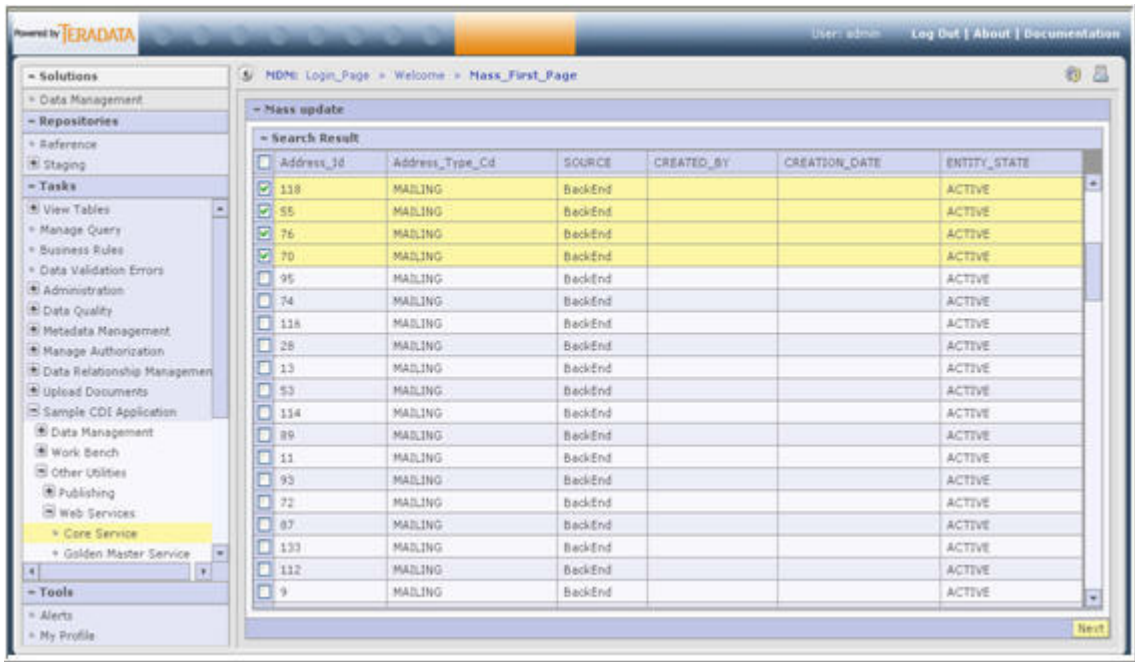
The record will be deleted and the success message will be displayed on the same screen.

To Mass Update Details, perform the following steps:

- 1 On the **Welcome** page (Figure 92) select the operation as MassUpdate and document name as Address from the respective drop-downs and click **Next**.

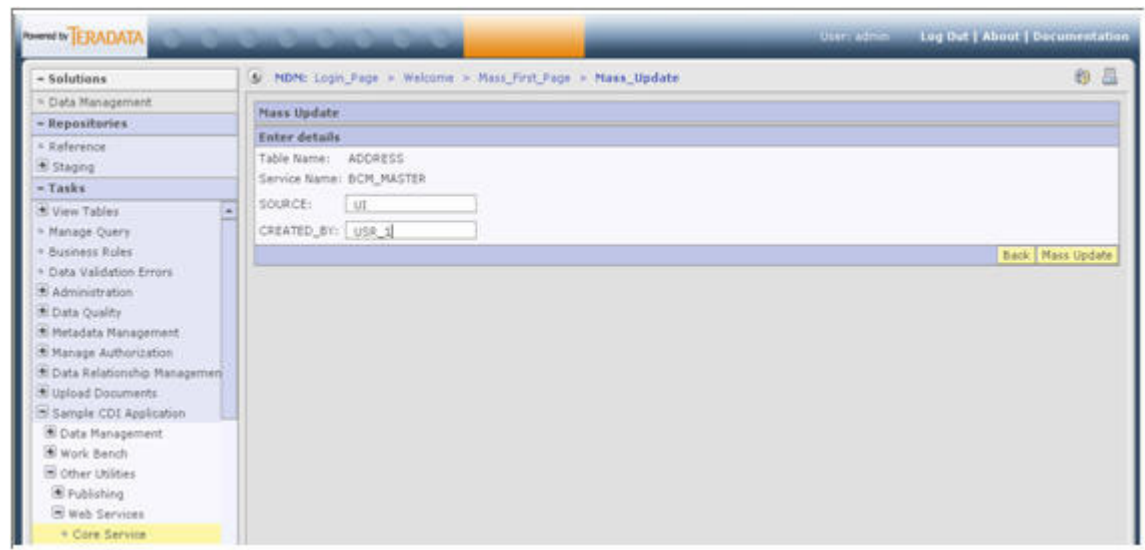
The **Mass Update** page (Figure 97) is displayed.

Figure 97: Mass Update



- 2 On **Mass Update** page (Figure 97), select the required records and click **Next**.
The **Mass Update-Enter Details** page (Figure 97) is displayed.

Figure 98: Mass Update



- 3 On the **Mass Update-Enter Details** page (Figure 97), edit the details and click **Mass Update**.
Message: Mass updated successfully is displayed.

Outgoing Third Party Web Service

To consume third party web service, create below listed request. To achieve that request format, sampleApplication uses yahoo workflow where user provides some inputs. By the input to that workflow, form the desired request format, call the WSDL node and pass the constructed request to WSDL node.

Third Party Web service - Request Format

Note: This request format will be different for other third party web service call.

Request to WSDL node passes under **WSDL Transform Input** tab of WSDL node.

Example:

```
<t:GetStockHeadlines xmlns:t="http://www.xignite.com/services/">
  <t:Symbols>TDC</t:Symbols>
  <t:HeadlineCount>2</t:HeadlineCount>
</t:GetStockHeadlines>
```

The [Figure 99](#) displays the WSDL transform input.

Figure 99: WSDL Transform Input

Node Get Stock Headlines

Name: Get Stock Headlines

Description:

Node Input Var: Type:

Node Output: Type:

TimeOut: 0

Exception Output: Type:

Binding Url:

Service: XigniteNews Operations: GetStockHeadlines

Provide stock and market news operations.

<html>Get headlines for a list of US Domestic equities.· Sour

WSDL Transform Header | **WSDL Transform Input** | Transport Level Endorsement | WSDL Transform Output

parameters

```
<?xml version="1.0" encoding="UTF-8"?>
<s:element xmlns:s="http://www.w3.org/2001/XMLSchema" name="GetStockHeadlines">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="Symbols" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="HeadlineCount" type="s:int" />
    </s:sequence>
  </s:complexType>
</s:element>
```

```
<action>
  <TO_DOCVAR AssignToVar="WsdParametersIn">
    <t:GetStockHeadlines xmlns:t="http://www.xignite.com/services/">
      <t:Symbols>{$outXml/COMPANY/@Value}</t:Symbols>
      <t:HeadlineCount>{$outXml/HEADLINES_COUNT/@Value}</t:HeadlineCount>
    </t:GetStockHeadlines>
  </TO_DOCVAR>
  <PRINTLN Var="WsdParametersIn"/>
</action>
```

Outline Source

OK Cancel

Third Party Web Service - Response

```
<GetStockHeadlinesResponse xmlns="http://www.xignite.com/services/">
  <GetStockHeadlinesResult>
    <StockNews>
      <Outcome>Success</Outcome>
      <Identity>IP</Identity>
      <Delay>0</Delay>
```

```

        <Headline>The final frontier of business advantage</Headline>
        <Ticker>TDC</Ticker>
        <Date>11/26/2009</Date>
        <Time>1:10 PM</Time>
        <Source>FT.com</Source>
        <Url><![CDATA[http://www.ft.com/cms/s/bd48ef86-d95b-11de-b2d5-
00144feabdc0.html?referrer_id=yahooofinance&ft_ref=yahool&segid=03058
]]></Url>
    </StockNews>
</StockNews>
    <Outcome>Success</Outcome>
    <Delay>0</Delay>
    <Headline>France Telecom, TDC to merge Swiss operations</
Headline>
    <Ticker>TDC</Ticker>
    <Date>11/25/2009</Date>
    <Time>6:31 AM</Time>
    <Source>AP</Source>
    <Url>http://biz.yahoo.com/ap/091125/
eu_france_telecom_tdc_swiss.html</Url>
</StockNews>
</GetStockHeadlinesResult>
</GetStockHeadlinesResponse>

```

The [Figure 100](#) displays the WSDL transform output.

Figure 100: WSDL Transform Output

The screenshot shows a dialog box titled "Node Get Stock Headlines". It contains several input fields for configuration: Name (Get Stock Headlines), Description, Node Input Var, Node Output, TimeOut (0), Exception Output, and Binding Url. Below these are Service (XigniteNews) and Operations (GetStockHeadlines). A description field contains the text "Provide stock and market news operations." and a source field contains the HTML snippet: "<html>Get headlines for a list of US Domestic equities.· Sour".

The "WSDL Transform Output" tab is selected, showing the following XML code:

```
<s:element xmlns:s="http://www.w3.org/2001/XMLSchema" name="GetStockHeadlinesResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetStockHeadlinesResult" type="tns:ArrayOfStockNews" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Below the XML code is an action block:

```
<action>
  <PRINTLN Var="WsdparametersOut"/>
  <SET Var="headlinesXml" FromSelect="$WsdparametersOut"/>
</action>
```

At the bottom of the dialog are "OK" and "Cancel" buttons.

Index

B

Business Rules 101
 Sql Filter Rule 101
 Sql Insert Rule 103
 Sql Update Rule 104
 Sql Validation Rule 102

C

Categorize Field Types 92, 93
 Custom Application 60
 Creation 62
 Existing 62, 74
 Folder Structure 87
 Location 62, 63
 Models and Dictionary 64, 65
 Schema Generation 69
 Service 65, 68
 Studio Display 63
 Web UI Component 70
 Customizations 20

D

Data Modeling 89
 Data Models 4
 Data Quality Monitors 49
 Deployment Manager 50
 Source of Deployment
 Database option 77
 File System option 79
 Document Type Definition 10
 Documents 18

E

Enterprise Data Warehouse 39
 ERwin 92
 Erwin data model 39

F

Facets 19

H

Hierarchy Management 46

I

Index 19

J

JSP 22

K

Keys 19

L

Links 19

M

Manage 117
 Master Data 17
 Master Data Management 2, 58
 Best Practices 40
 Business Architecture 5
 Diagram 6
 Business Solution 58
 Custom Application 60
 Data Architecture 17
 Database Topology 8, 14
 Logical Database Table 16
 Development Elements 44
 X-Documents 44
 X-Operations 44, 45
 X-Path 44, 46
 X-Rules 44
 Documentation viii
 Implementation Methodology 6
 Key Features 59
 Overall Technical Architecture 8, 9
 Platform Architecture 8, 11
 Development and Deployment Relationship 12
 Logical Deployment 12
 Reference Application 5
 Sample Application 60
 CDI 61
 CDI Trillium 61
 Studio Architecture 8, 9, 10
 Supported Data Type Mappings 16
 Technical Architecture 3, 8
 Master tables 42

- N
- Net Change tables 42
- O
- Open Model Ingestion 39
- P
- Primary Key 19
- Properties 18
- Publication Meta Data 52
- Publication Object 121
- Publication Services 120
- Publication Workflow Node 53
- Purpose v
- R
- Recommendations 40
- REQUEST_ID 124
- S
- Sample Application 60, 74
 - CDI 61
 - Publishing 112
 - CDI Dictionary and Models 89
 - CDI Process Flow 99, 100
 - Customer Dashborad 107, 108
 - Data Acquisition 100
 - Data Cleansing 105
 - Data Syndication 112
 - Data Validation and Enrichment 101
 - CDI Trillium 61
 - CDI Trillium Process Flow 113, 114
 - Data Acquisition 114
 - Data Cleansing 114
 - Data Syndication 118
 - Data Validation 114
 - Publishing 117
 - Configure Project in Eclipse 84
 - Customer Dashboard 107
 - Auto Merge 107
 - Interactive Data Steward 108, 117
 - Manage Customer 109, 117
 - Customer Model Relationship 99
 - Data Model 98
 - CRM Model 98
 - RLDM Model 99
 - Enable Web Services 84
 - Load Predefined Data 84
 - Model Creation 89
 - Model Import 89
 - Erwin 91
 - Relational Database 90
 - XDocs 93
 - Setup 74
 - SCHEMA generation code 15
 - T
 - Table Editors 41
 - Teradata Studio 3
 - Timer Service 52
 - U
 - Unique Key 19
 - User Defined 19
 - V
 - Validation Rules 41
 - W
 - Web Services 2, 47
 - Workflow Manager 11
 - Workflows 20
 - World Class Customer Support 54
 - X
 - XML script 21
 - XSL 22