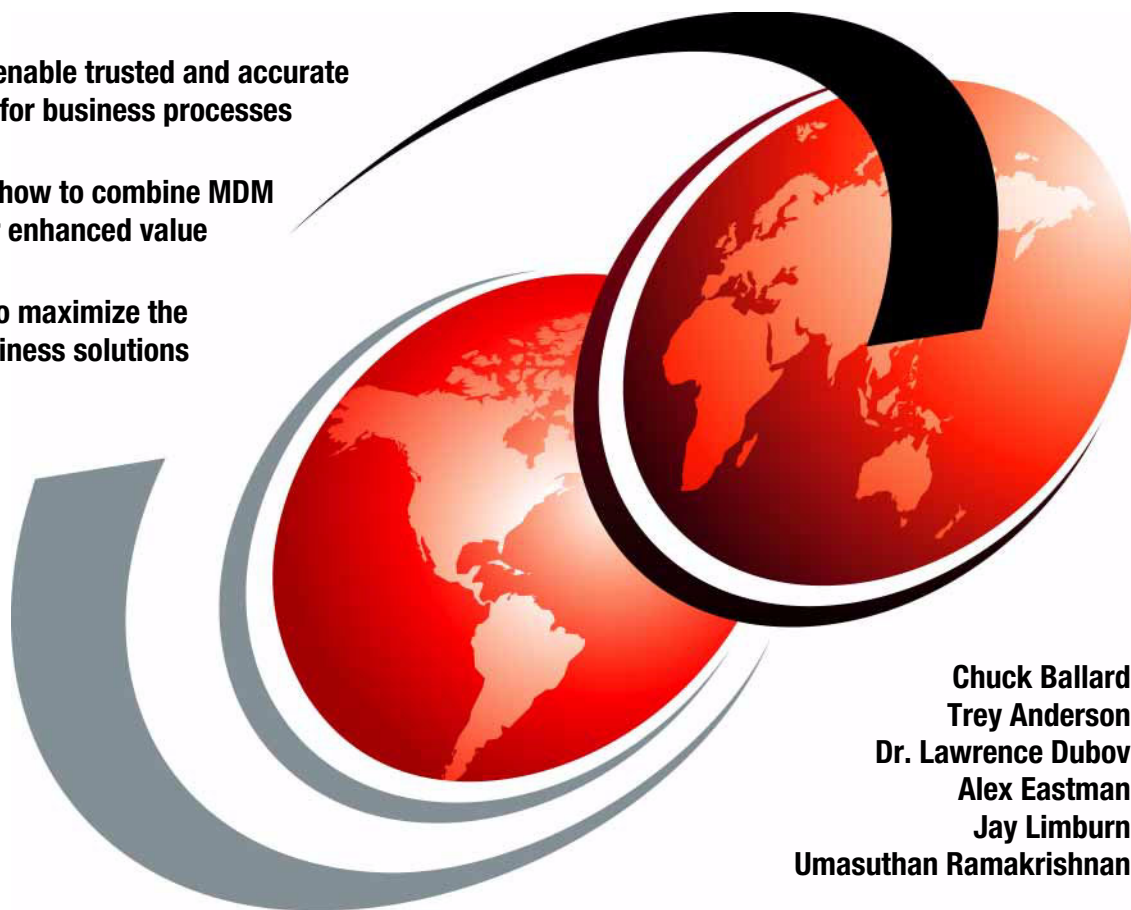


Aligning MDM and BPM for Master Data Governance, Stewardship, and Enterprise Processes

See how to enable trusted and accurate information for business processes

Understand how to combine MDM and BPM for enhanced value

Learn how to maximize the value of business solutions



Chuck Ballard
Trey Anderson
Dr. Lawrence Dubov
Alex Eastman
Jay Limburn
Umasuthan Ramakrishnan



International Technical Support Organization

**Aligning MDM and BPM for Master Data Governance,
Stewardship, and Enterprise Processes**

March 2013

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (March 2013)

This edition applies to Version 10 IBM InfoSphere Master Data Management Server, IBM Business Process Manager V7.5 Express, and IBM Business Process Manager V7.5 Standard or Advanced editions.

© Copyright International Business Machines Corporation 2013. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

- Notices** vii
- Trademarks** viii
- Preface** ix
 - The team who wrote this book x
 - Now you can become a published author, too! xii
 - Comments welcome xii
 - Stay connected to IBM Redbooks xiii
- Chapter 1. Introduction to master data management and business process management** 1
 - 1.1 A business case for master data management 2
 - 1.1.1 Benefits 3
 - 1.1.2 Value proposition 3
 - 1.2 A business case for business process management 4
 - 1.2.1 Benefits 4
 - 1.2.2 Value proposition 5
 - 1.3 Creating a synergistic value with MDM and BPM 5
 - 1.3.1 Implementation patterns 6
- Chapter 2. Aligning MDM and enterprise business process strategies** . . 11
 - 2.1 Master data creation 12
 - 2.2 Master data consumption 13
 - 2.2.1 Improved business decisions with the single version of truth from the trust source 13
 - 2.2.2 Improving conversion rates by present offers that are more relevant 14
 - 2.2.3 Reducing cost by removing redundant or irrelevant offers and mailings 14
- Chapter 3. Master data governance and stewardship** 15
 - 3.1 Definition and objectives of master data governance 16
 - 3.2 Data governance maturity 18
 - 3.3 Master data quality 19
 - 3.3.1 Policies 19
 - 3.3.2 Processes 20
 - 3.3.3 Metrics and KPIs 22
- Chapter 4. Integration approaches and proven practices** 27
 - 4.1 Business Process Manager tools 28

4.1.1 IBM BPM Process Designer	28
4.1.2 IBM Process Portal	28
4.2 Integration overview	28
4.2.1 Java Integrator	29
4.2.2 Web Services Integrator	30
4.3 MDM integration approaches	31
4.3.1 MDM web services	31
4.3.2 Virtual MDM Java interface	32
4.4 MDM Application Toolkit capabilities	32
4.5 Overview of Process Designer	33
4.5.1 Starting Process Designer	33
4.5.2 Process authoring environment	35
Chapter 5. Introduction to the InfoSphere MDM Application Toolkit	37
5.1 Components of the InfoSphere MDM Application Toolkit	38
5.1.1 Architectural overview	38
5.1.2 MDM data types	40
5.1.3 Hierarchy widget overview	41
5.1.4 Integration services	44
5.1.5 Overview of the MDM REST service	45
5.1.6 Mobile applications	46
5.2 Installing the InfoSphere MDM Toolkit	48
5.2.1 Installing the MDM Application Toolkit .twx file	48
5.2.2 Installing the MDM REST Service	50
5.3 Building applications with the InfoSphere MDM	
Application Toolkit for BPM	51
5.3.1 MDM data types for creating BPM variables	51
5.3.2 Customizing an InfoSphere MDM Application Toolkit data type	52
5.3.3 The MDM Tree UI control	53
5.3.4 Configuring the REST service for BPM	57
5.3.5 Extending and adding services	59
Chapter 6. Defining a master data creation process	61
6.1 Overview of the example business process	62
6.2 Creating the business process definition	63
6.2.1 Creating a process application	63
6.2.2 Creating a human service	69
6.2.3 Defining variables	74
6.2.4 Building coach UI elements	75
6.2.5 Building the integration service	81
6.2.6 Connecting the UI coaches and integration service	95
6.2.7 Running the process	98
6.2.8 Checkpoint	98

6.3	Integrating the business process with MDM by using the MDM Tree	99
6.3.1	Extending the existing human service.	99
6.3.2	Configuring the nested service	100
6.3.3	Configuring the MDM Tree	103
6.3.4	Connecting the nodes with the service	104
6.3.5	Running your process	106
6.3.6	Checkpoint	106
6.4	Extending the process to save updates to an entity	107
6.4.1	Extending the process.	107
6.4.2	Creating variables to hold the properties	108
6.4.3	Creating a connection variable	108
6.4.4	Binding a variable to the MDM Tree	111
6.4.5	Joining the process flow steps	112
6.4.6	Running the process	113
6.4.7	Checkpoint	114
6.5	Data stewardship processes within enterprise processes.	114
6.5.1	Enterprise processes	115
6.5.2	Integration of enterprise and MDM governance processes	115
6.6	Summary	116
Chapter 7. Master data policy enforcement with BPM Express		119
7.1	Overview of policy enforcement	120
7.1.1	Components of master data governance	121
7.1.2	Data stewardship process example	121
7.1.3	Sales territory distribution and payouts example	121
7.2	Business case	122
7.3	A technical perspective	123
7.3.1	Aspects of policy enforcement	123
7.4	Assurances that master data is a trusted asset	124
7.5	Business process scenarios	125
7.5.1	Critical data change review and approval	125
7.5.2	Policy remediation.	128
7.5.3	Entity resolution.	129
Chapter 8. Using and extending MD policy enforcement samples.		131
8.1	Configuring MDM Standard Edition for messaging	132
8.1.1	Configuring the event handler.	132
8.1.2	Event handler settings.	134
8.1.3	Deploying the Event Handler	135
8.1.4	Configuring the hub model	136
8.1.5	Configuring IBM BPM Express	137
8.2	Configuring InfoSphere MDM Advanced Edition for messaging	138
8.2.1	MDM Advanced Edition configurations	138

8.2.2 Configuring metadata	139
8.2.3 Configuring IBM BPM Express	140
8.3 Importing samples	140
8.4 Configuring process applications	142
8.5 Integrating BPM in the enterprise	144
8.5.1 Integration patterns used in the samples	144
8.6 Extending the samples based on a business case	145
8.6.1 Adding and removing an attribute	146
8.6.2 Changing the attribute label	153
8.6.3 Adding and changing a policy	167
8.6.4 Adding an activity	169
8.6.5 Adding activity UIs	170
8.7 Reporting for success	171
Appendix A. MDM Advanced Edition configuration script	173
Appendix B. BPM Express Edition for messaging integration	185
Related publications	191
IBM Redbooks	191
Online resources	192
Help from IBM	192

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®


Cognos®

IBM®

InfoSphere®

Rational®

Redbooks®

Redbooks (logo) ®

WebSphere®

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.

Preface

An enterprise can gain differentiating value by aligning its master data management (MDM) and business process management (BPM) projects. MDM provides consistency to improve the integrity of business processes, making those processes smarter, more effective, and productive. BPM is an agile process platform that can provide consistent visibility, collaboration, and governance. By aligning their MDM and BPM projects, organizations can optimize their business performance through agile processes that empower decision makers with the trusted, single version of information.

Every enterprise process uses data that must be trusted, accurate, and beneficial before an organization can use it as information to differentiate itself in the marketplace. Whether human or automated, decisions are only as good as the information that is available at the time they are made. Many companies deploy MDM strategies as assurances that enterprise master data can be trusted and used in the business processes.

From a technical perspective, MDM is a set of software solutions that manage the creation, governance, delivery, and use of master data across the organization. IBM® InfoSphere® Master Data Management creates trusted views of data assets and elevates the effectiveness of an organization's most important business processes and applications. It improves business results, lowers costs, reduces risk, and enables strategic agility to meet current and future business needs.

This IBM Redbooks® publication provides an overview of MDM and BPM. It examines how you can align them to enable trusted and accurate information to be used by business processes to optimize business performance and bring more agility to data stewardship. It also provides beginning guidance on these patterns and where cross-training efforts might focus.

This book is written for MDM or BPM architects and MDM and BPM architects. By reading this book, MDM or BPM architects can understand how to scope joint projects or to provide reasonable estimates of the effort. In addition, BPM developers (or MDM developers with BPM training) can learn how to design and build MDM creation and consumption use cases by using the MDM Toolkit for BPM. They can also learn how to import data governance samples that are ready for immediate use and extend them to enable collaborative stewardship of master data.

The team who wrote this book

This book was produced by a team of specialists from around the world working with the International Technical Support Organization (ITSO), in San Jose, California (CA).



Chuck Ballard is a Project Manager at the ITSO, in San Jose, CA. He has over 35 years of experience, holding positions in the areas of product engineering, sales, marketing, technical support, and management. His expertise is in the areas of database, data management, data warehousing, business intelligence, and process re-engineering. He has written extensively on these subjects, taught classes, and presented at conferences and seminars worldwide. Chuck has both a bachelor degree and a master degree in industrial engineering from Purdue University.



Trey Anderson is the MDM Product Manager for IBM Software Group. He has had various leadership positions in engineering, professional services, and product management. Trey is focused on strategic integrations and solutions between InfoSphere MDM and other IBM technologies. He is currently responsible for defining the Go-To-Market (GTM) strategy and solution for IBM BPM. Trey graduated from Texas A&M University with a degree in accounting and management information systems.



Dr. Lawrence Dubov (Larry) is Master Data Governance Product Director for IBM Software Group. He is a recognized IT practitioner, product director, scientist, business executive, and thought leader in complex business-driven technology solutions and products. Larry's MDM experience spans from product management and R&D to technical selling and consulting. He is an internationally recognized speaker and an author of over 200 publications and blogs including the *Master Data Management and Customer Data Integration for a Global Enterprise* (McGraw-Hill, 2007) and *Master Data Management and Data Governance* (McGraw-Hill, 2010).



Alex Eastman is a Senior Product Manager for MDM at IBM. His experience includes implementing, architecting, selling, and defining the roadmap and strategy of MDM products. Alex focuses on helping organizations use MDM to improve business processes through the IBM MDM Application Toolkit. He also helps organizations integrate MDM with big data capabilities to enable the use of data of large volume and variety within MDM. Alex has a computer science degree from Brigham Young University and an Master of Business Administration degree from the Darden Graduate School of Business Administration at the University of Virginia.



Jay Limburn, MBCS CITP, is an IBM Senior Technical Staff Member at the IBM Software Development Laboratory in Hursley, UK. Working within the InfoSphere division of Information Management, Jay is the lead architect for a part of the portfolio in which the primary focus is on master data consumption, ensuring master data can be delivered in an efficient manner to business users and increase their ROI. Jay is a recognized expert on master data governance and strategies that allow organizations to extract value from their master data engagements. He has presented at conferences worldwide on these topics, and as a UK Senior Inventor, he holds 9 patents in these areas.



Umasuthan Ramakrishnan is a Senior Software Engineer at the IBM Software Lab in Bangalore, India. He has worked in various positions in application and product development and delivery. He also has vast experience in the phases of software development life cycle. In his current role, Umasuthan works as a product architect for the IBM MDM portfolio of products with a focus on integrating IBM MDM products with other key IBM Technology for increased consumability of the IBM MDM portfolio. He holds two patents in the areas of collaboration and statistical modeling.

We thank the following contributors to this book who provided content review, subject expertise, and support:

Scott Nichols
Sean Pizel
Ron Poggio
Kinjal B. Shah
Wei Zheng

Client Technical Specialists, IBM Software Group, IBM US

Mary Comianos, Publications Management
Ann Lund, Residency Administration
Linda Robinson, Graphics Support
IBM ITSO

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an email to:


redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>



Introduction to master data management and business process management

This chapter introduces master data management (MDM) and business process management (BPM). It describes the synergistic value that is derived from aligning these two often separate initiatives.

Organizations should use the free BPM Express supporting program entitlements that are available with MDM to build data stewardship processes to enforcing data quality policies. Broader enterprise processes (such as customer on-boarding) should use MDM to assure that decision makers have the most accurate, comprehensive, and timely information available. The remainder of this book explores the technical considerations and implementation standard practices that organizations can use to realize the benefits and value of these initiatives.

This chapter includes the following sections:

- ▶ A business case for master data management
- ▶ A business case for business process management
- ▶ Creating a synergistic value with MDM and BPM

1.1 A business case for master data management

Master data is the one true source of data about customers, patients, suppliers, partners, products, materials, employees, accounts, and other critical entities. It is the basis for the high-value, core information that is used to support critical business processes across the enterprise. It is also at the heart of every business transaction, application, report, and decision.

Organizations hold and replicate master data across many different applications and customer touch points, such as order processing, customer service, and reporting systems. However, many of these source systems create, update, and maintain the data in their own unique way, typically resulting in a lack of consistency among them. Critical data elements might be missing, incomplete, duplicated, or otherwise inconsistent. With no single, unified, and accurate “version of truth” about the business, organizations miss opportunities to increase revenues, use a competitive advantage, and reduce costs. Critical business processes are hampered by incorrect data that can lead to poor decisions and result in poor business outcomes.

Regardless of how good the quality of the information is that a business maintains, the desired business goals cannot be realized if the main business processes have the following characteristics:

- ▶ Are inefficient
- ▶ Do not meet client needs
- ▶ Are difficult to adapt to meet new business challenges

Many companies have deployed MDM strategies to resolve the problem of inconsistent data. MDM is a discipline that provides a single, unified, and trusted view of master data entities for any user or application. From a technical perspective, MDM is a set of software solutions that manage the creation, governance, delivery, and use of master data across the organization.

MDM connects all the information that was gathered about a particular entity or event from all the enterprise systems to form a more complete view of that entity or event to enable an understanding of its true value. MDM also provides mechanisms and governance for consistent use of master data across the organization. As a result, MDM enables better business processes.

IBM InfoSphere Master Data Management creates trusted views of data assets and elevates the effectiveness of an organization’s most important business processes and applications. It improves business results, lowers costs, reduces risk, and enables strategic agility to meet current and future business needs.

The IBM InfoSphere Master Data Management platform provides the assurance that enterprise master data is accurate, valid, relevant, and timely, and it can be trusted. InfoSphere MDM provides trust through its own data governance and stewardship processes for continuous enforcement of data quality, ensuring that business process requirements are satisfied. MDM and BPM are a mix of technology and methodology that enable organizations to improve process performance through better management and governance.

1.1.1 Benefits

IBM InfoSphere Master Data Management offers organizations the following advantages:

- ▶ Reduced time to market and new product or service introduction
- ▶ Better multichannel integration
- ▶ Increased supply chain visibility and a simplified environment for increased multienterprise collaboration
- ▶ Reduced customer churn and improved customer loyalty
- ▶ Greater customer intimacy
- ▶ Improved regulatory compliance
- ▶ Better analysis and decision making

1.1.2 Value proposition

Business and technical leaders recognize the opportunity to create and use trusted information to establish a single view, gain deeper insight, make better decisions, and improve business outcomes. IBM enables these leaders to address the key drivers for MDM, which can lead to success in the following areas:

- ▶ Strategic initiatives to increase revenue, including customer intimacy programs, consistent user experience across multiple sales channels, cross-sell/up-sell programs
- ▶ Cost-cutting projects, including automating manual processes, eliminating duplicate mailings, consolidating or retiring applications, and integrating systems from acquired companies)
- ▶ Business agility programs, including faster time to market and the ability to create tailored or personalized product offerings
- ▶ Compliance and governance and initiatives

1.2 A business case for business process management

Organizations use business processes to define the execution and management of their day-to-day business operations. BPM applications provide organizations the ability to measure the efficiency and effectiveness of those business operations. From this information, business processes can be easily changed by using the agile development capabilities of BPM to improve operational throughput and quality and to reduce cost. This approach supports an ongoing transformation effort to achieve continuous improvements in vital business processes, which can lead to continuous improvements in the business results.

Using BPM to improve business processes is a critical component for the continued success of businesses. BPM can also satisfy the requirement that these processes be agile enough to adapt to organizational changes, provide a rich user experience, be open to existing data systems, and provide visibility into process and individual performance.

BPM is a comprehensive management approach. It aligns the business processes of an enterprise with its corporate goals and strategies and then continuously improves them. This approach is most often assisted by a BPM software suite, which helps with the governance, deployment, tracking, and automation of business processes. This suite also provides process visibility so that decision makers can quickly and effectively change the processes to achieve the strategic goals of the business. This focus on strategic goals and continuous improvement generally demands a robust implementation and governance methodology that insists on continuous and direct business stakeholder involvement.

1.2.1 Benefits

BPM methodologies, supported by a business process management system (BPMS) allow for the rapid creation of value-centric process solutions with a dramatically reduced time to return on investment (ROI). When implemented, these processes are more easily modified to adapt to change, whether as a result of market conditions, regulations, or strategic shifts in corporate goals. By abstracting the business process logic and rules from traditional applications and services layers into business processes, the reuse and efficiency of IT resources increases. Business key performance indicators (KPIs) and data governance metrics, intrinsically provided by process solutions that are implemented by a BPMS, prove the effectiveness of process improvements, allowing businesses to better prioritize their IT Initiatives.

1.2.2 Value proposition

Many of the benefits of BPM are realized during an enterprise's first process implementation, such as an IBM led Quick Win Pilot (QWP). QWP is a 12-week services offering that achieves only a limited production release. It often creates a chain reaction of process-oriented adjustments that allow businesses to achieve dramatic improvements. Often only the degree to which senior business and technical leaders can transform their IT and the business operation dictates how these process successes can be scaled into BPM programs that support and drive corporate initiatives. Such initiatives include improved product quality, reduced time-to-market, expanded markets, increased customer satisfaction, and improved profit margins.

1.3 Creating a synergistic value with MDM and BPM

Organizations are looking to transform their business. To fully realize that goal, they must bring their MDM and BPM projects closer together. Aligning the priorities, goals, requirements, milestones, and stakeholders of these often separate teams affords significant benefit to each team, and the organization and its employees benefit.

When organizations align their MDM and BPM projects, they maximize the value of each solution. Many analysts recommend that clients and vendors adopt a strategy that supports this aligned approach. As a result, IBM includes BPM Express in the IBM InfoSphere MDM product as a platform to support process-oriented data stewardship. In addition, IBM provides the InfoSphere MDM Application Toolkit for BPM that enables organizations to use InfoSphere MDM as a service from BPM. It simplifies the development of MDM-powered enterprise processes such as customer onboarding and order to cash.

The number of joint MDM and BPM projects is anticipated to grow at a considerable rate in the upcoming years. Despite IBM and IBM Business Partners having deep competencies in these technologies, the ability to successfully implement these joint patterns requires some level of cross-training and a new perspective.

A combined MDM and BPM solution brings more value to business processes. As a result, enterprises are looking for ways to connect all of these processes in a managed technology solution. The IBM InfoSphere MDM product portfolio provides a way to apply governance to enterprise master data by using IBM InfoSphere MDM solutions. With InfoSphere MDM, you can combine process agility with trusted data and support processes and policies that can help enforce data quality throughout the enterprise for enhanced business efficiency and effectiveness.

1.3.1 Implementation patterns

By aligning MDM and BPM, high-performance, agile business processes can use trusted and accurate information to improve performance, bringing trusted data to processes and more agility to data stewardship. For the key business drivers and value propositions to align MDM and BPM projects, see the white paper, *Transforming business processes by aligning BPM and MDM*, at:

<http://www.ibm.com/common/ssi/cgi-bin/ssialias?infotype=SA&subtype=WH&htmlfid=WSW14176USEN>

MDM and BPM intersect or align at three points as illustrated in Figure 1-1.

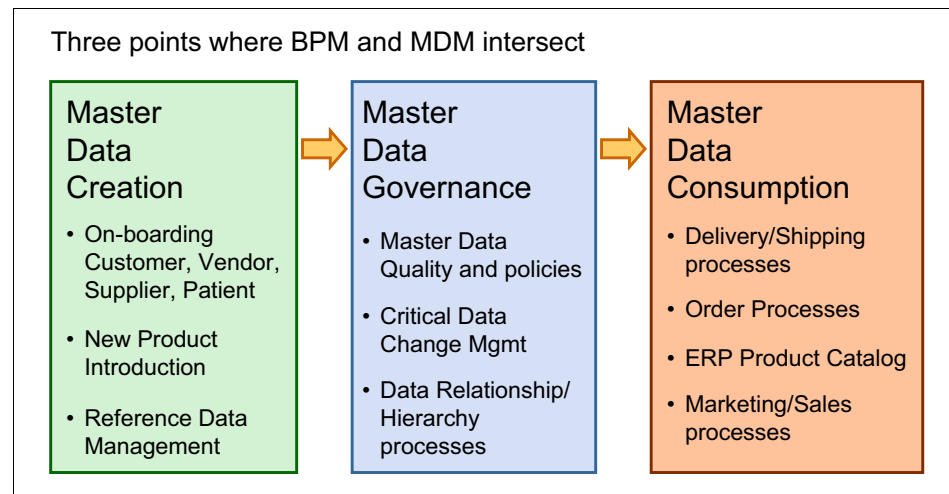


Figure 1-1 Points where BPM and MDM intersect

The points at which MDM and BPM intersect are called *patterns*. MDM and BPM have the following patterns:

► Master data creation

After an organization identifies its, the organization must aggregate it to link and resolve duplicates and to ensure that processes are not creating duplicates. The Master Data Creation process calls for master data to be added to the MDM hub. In this pattern, customer onboarding, account creation, vendor onboarding, and other typical business processes push the master data that they author into MDM and search MDM to update existing master data. Probabilistic, deterministic, and other automated entity consolidation techniques are applied to resolve the master data into the single accurate version of the truth.

► Master data consumption

Master data is created for use within business process. Business process decisions that use accurate data are more productive and efficient. Consider the following example where a business process does not use accurate data.

John's business recently moved. When he places an order with the call center representative (Billy), John is not found by his address nor his home phone number. The CRM system creates an unwanted duplicate account. John expects this purchase to promote him to a higher class of service that includes free shipping.

John is confused when he sees that his invoice has a shipping charge. He calls back into customer service where, after some investigate, Sue finds two accounts for John with the same cell phone number. Sue attempts to resolve the issue: updating the contact information for the original account and deactivating the duplicate account. Sue must manually upgrade John's class of service, and his recent purchase history is orphaned from this account.

Every process uses data. Whether data comes from various enterprise systems or one application, it must be accurate and trusted before it can be used. Whether human or automated, decisions are only as good as the information available at the time they are made. MDM ensures that accurate, trusted master data is available to the process decisions that need it.

► Master data governance

Creating master data and using it within business processes are the two primary points where MDM and BPM intersect within a business. In some scenarios, newly created master data must have additional assurances applied before business users can have confidence in the master data.

This confidence can be required for consumption within their business processes, as illustrated in Figure 1-2.

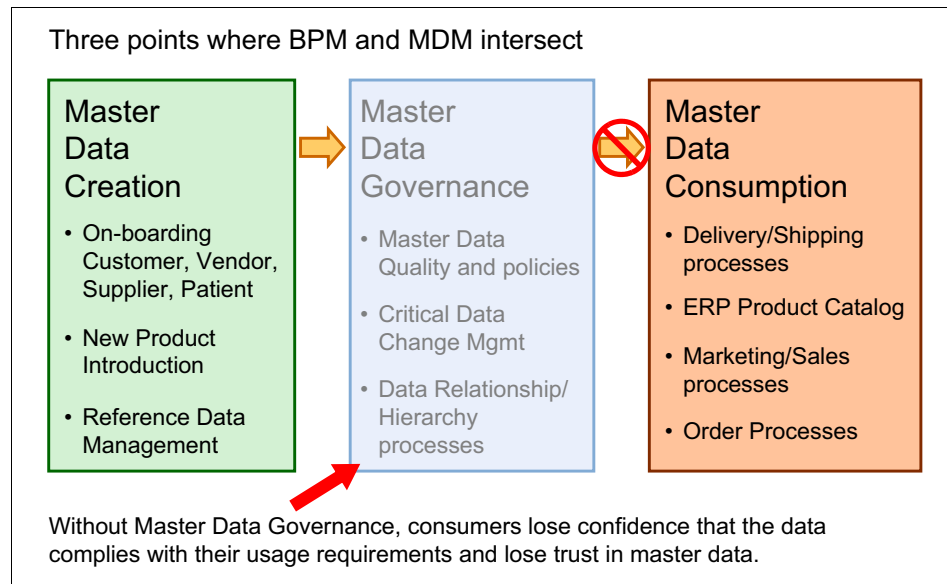


Figure 1-2 Consumer confidence

Master data governance provides business owners assurances that master data is a trusted asset that is ready for use within their business processes. Consuming processes have requirements that master data must comply with. These consumption-centric master data requirements can include attribute validation requirements that are associated with the completeness of a record, specific attribute values, and code table validation. Master data governance provides the capabilities that are necessary to administer and monitor these requirements as policies. Using a master data governance process to enforce these policies provides the assurances to business owners that master data is not only accurate but also supports their usage requirements. BPM Express, which is the MDM supporting program, is the platform for implementing this process-oriented data stewardship.

Figure 1-3 shows the master data governance process to enforce these policies. This process include policy administration, policy monitoring, and policy enforcement.

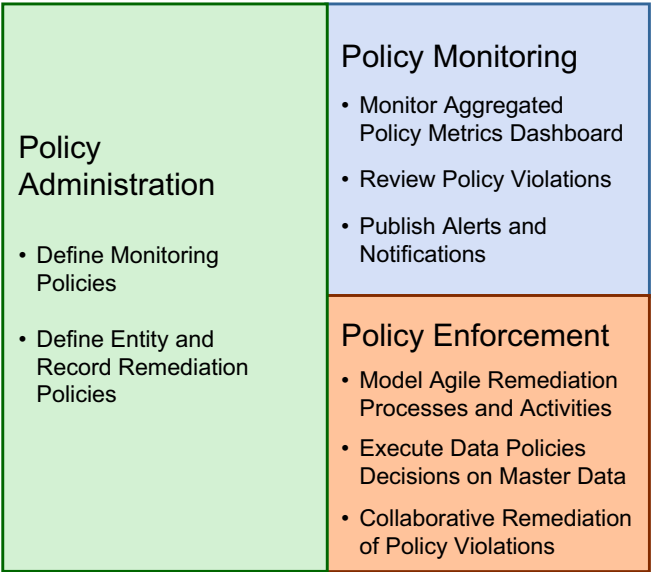


Figure 1-3 Master data governance process to enforce these policies



Aligning MDM and enterprise business process strategies

As explained in Chapter 1, “Introduction to master data management and business process management” on page 1, the integration of MDM and BPM delivers real value to organizations. MDM ensures that master data is complete, accurate, and trustworthy, and BPM ensures that this data is available to the business users who need it within the business processes that require it. Without MDM, business processes act on inaccurate data and result in ineffective outcomes. Without BPM, master data is not used to improve business processes and outcomes.

This chapter highlights ways that MDM and BPM work together to improve specific business processes. It includes the following sections:

- ▶ Master data creation
- ▶ Master data consumption

2.1 Master data creation

Organizations that deploy MDM solutions often start with a back-office focus. They start by integrating the sources of master data to the MDM hub. They reconcile multiple records for each master data entity into consolidated references or records. They also implement data stewardship to improve the quality of these sources master data. You might call this *passive integration*. Passive integration is an approach that uses MDM capabilities to make master data that is created elsewhere more accurate and trustworthy. This approach is generally where most organizations start with on their MDM journey, resulting in quick time-to-value.

Having conquered passive integration, organizations can then expand their use of MDM to use MDM capabilities to improve business processes that create master data. Rather than just using MDM to passively improve master data after-the-fact, these organizations use MDM to ensure that master data is accurate and trustworthy at the point of creation. You might call this *active integration* because it is a proactive, rather than reactive, approach to improving the quality of master data.

Consider this question: Where do duplicate customer records come from? Often, the root cause of this duplication comes from existing systems and processes that make it difficult to find existing records for customers. When a user does not know that a record for the customer exists (because they cannot find it), the user creates a customer record.

The following example shows how MDM and BPM can improve this process of master data creation. A salesperson just found a sales lead for a prospect named *Redboks Corp.* Before the salesperson enters this prospect into the CRM system, the salesperson searches for *Redboks*. After not finding a match, the salesperson creates a customer record. Fortunately, the organization implemented MDM, at least in a passive approach. The MDM solution recognizes that Redboks Corp. is the same as an existing customer *Redbooks Corporation*, and the data stewards correct the information later. When used appropriately, MDM can help to actively find the existing customer record. By integrating the process for entering sales prospects with MDM for the initial search, the salesperson can find the existing customer record from the start and avoid cleaning up the data later.

The combination of BPM (to model and enforce the business process) and MDM (to accurately identify the customer) provides a powerful solution to ensure that master data is accurate and trustworthy from the point of its creation.

2.2 Master data consumption

As explained in the last section, organizations can realize more value from their MDM investments by using MDM in their processes that create master data. The benefits of MDM do not end with improving master data creation. The ultimate value of MDM is achieved when complete and accurate master data is made available to business users and improves operations and decision making. This concept is called *master data consumption*.

How often do customers receive offers for credit cards that they already own? How many hotel associates have to ask whether a customer has stayed there before because their point-of-service system cannot tell them? Shortcomings in business processes and applications like these types can lead to customer dissatisfaction and lost revenue opportunities. Alternatively, receiving relevant and targeted offers and being recognized as a valuable customer engenders loyalty and increases customer spending.

2.2.1 Improved business decisions with the single version of truth from the trust source

When decision makers have access to complete and accurate master data, they make better decisions and feel more confident about the data on which they base their decisions. Consider the following questions that benefit from accurate master data:

- ▶ How valuable is this customer?
- ▶ How much effort should we invest into retaining this customer?
- ▶ What level of service does this customer deserve?

When you answer these questions and make decisions based on them, you need all of the relevant data that is necessary to inform your decision. You need to know all of the business that this customer does with your company, across all lines of business, regardless of any silos. You need to know how profitable the customer is, taking all of this business into account. You should know the relationships that this customer has with other key customers and stakeholders so that you can consider how this decision might affect them.

MDM ensures that all of this data has been consolidated and is complete and accurate. Business processes that involve these kinds of decisions should make this data available to the decision makers when and where they need it.

2.2.2 Improving conversion rates by present offers that are more relevant

Organizations try to increase wallet share of existing customers by presenting them cross-sell and up-sell offers. To present meaningful offers to customers for additional products and services, organizations need accurate and trustworthy data about these customers. They also need to understand which offers are likely to be most interesting to each customer. Otherwise, customers start to ignore the offers, or worse, they become dissatisfied with them.

The business processes that identify campaigns and offers for customers can be improved by using master data. These processes should incorporate master data into campaign and offer management.

2.2.3 Reducing cost by removing redundant or irrelevant offers and mailings

How much money do organizations waste sending multiple mailings to the same customers, sending mailings to the wrong address, or attempting to send mailings to invalid addresses? MDM helps eliminate this waste by ensuring that the customer master data is complete and accurate. Duplicate customer records are consolidated into one customer master. Master data governance ensures that mailing addresses are valid, complete, and accurate.

Business processes that support mailings need to use master data to reduce the costs that are associated with mailing mistakes. Furthermore, when customers notify organizations of changes, such as a change of address, the business processes that support these changes must ensure that the new information is propagated to the applications and processes that need it.



Master data governance and stewardship

This chapter introduces the concept of master data governance, its objectives, and scope. It defines master data governance for multidomain master data management (MDM) with several data governance aspects and policy control types. It also describes the roles of the data governance processes and key performance indicators (KPIs) and their critical importance for achieving mature levels of data governance maturity.

This chapter includes the following sections:

- ▶ Definition and objectives of master data governance
- ▶ Data governance maturity
- ▶ Master data quality
- ▶ Processes
- ▶ Metrics and KPIs

3.1 Definition and objectives of master data governance

It is commonly recognized that business information is one of the most important assets of any modern enterprise. Enterprises have developed policies that declare information a most important corporate asset. However, these policies are often cannot be acted upon. No commonly accepted standard exists for measuring and reporting on the corporate information assets. A lack of information asset KPIs makes it difficult to establish data governance organizations. In addition, it is also difficult to assign individuals who are accountable for the quantity and quality of enterprise information assets overall. These assets might greatly contribute to the equity of the corporations and market capitalization. However, enterprises lack an ability to set meaningful quantitative targets for corporate information assets, their growth, and improvement.

Master data is the most critical and valuable subset of enterprise information assets. *Master data* refers to data that is foundational to business processes and is widely distributed. The distributed data, when well managed, directly contributes to the success of an organization, and when not well managed, poses the most risk.

Master data typically includes a few major master information domains, such as party/customer, product, location, service, account, employee, and supplier. It also includes the relationships between those domains. In addition, master data includes reference data that serves as qualifiers for the master information domains, such as account types, customer categories, industry codes, and geographical areas. The reference data justifies a priority treatment of master data, which requires both technology and business strategy.

In many enterprises, master data has been a major strategy and implementation focus area for the last few years. The MDM strategy and implementation effort require support from business. Insufficient business directions and support can adversely impact the outcomes of the MDM initiative. By the nature of master data, its broad distribution, and multifunctional use, no single business function can take exclusive responsibility for master data requirements, rules, and processes. In some cases, requirements from different functional areas can even conflict with each other in terms of their content and priorities. This situation substantiates the need for a cross-functional master data governance council.

The MDM market recognizes that data governance is critical for enterprise MDM implementations. Master data governance is a focus area of data governance that is dedicated to MDM implementations. Master data governance, as a discipline, concentrates on controls, methodologies, capabilities, and tools that are developed by modern MDM over the last decade.

Master data governance has the following objectives:

- ▶ Establish a master data governance council or board.
- ▶ Formulate master data governance policies that establish accountability and enforcement.
- ▶ Monitor, oversee, and enforce proactive, collaborative, and effective data stewardship that is driven by data governance.
- ▶ Acquire and use tools that enable master data governance and data governance-driven stewardship, including policy administration, enforcement, remediation, and monitoring.

IBM offers a comprehensive set of products and components for master data management and governance.

Figure 3-1 illustrates the IBM multidimensional approach to master data governance.

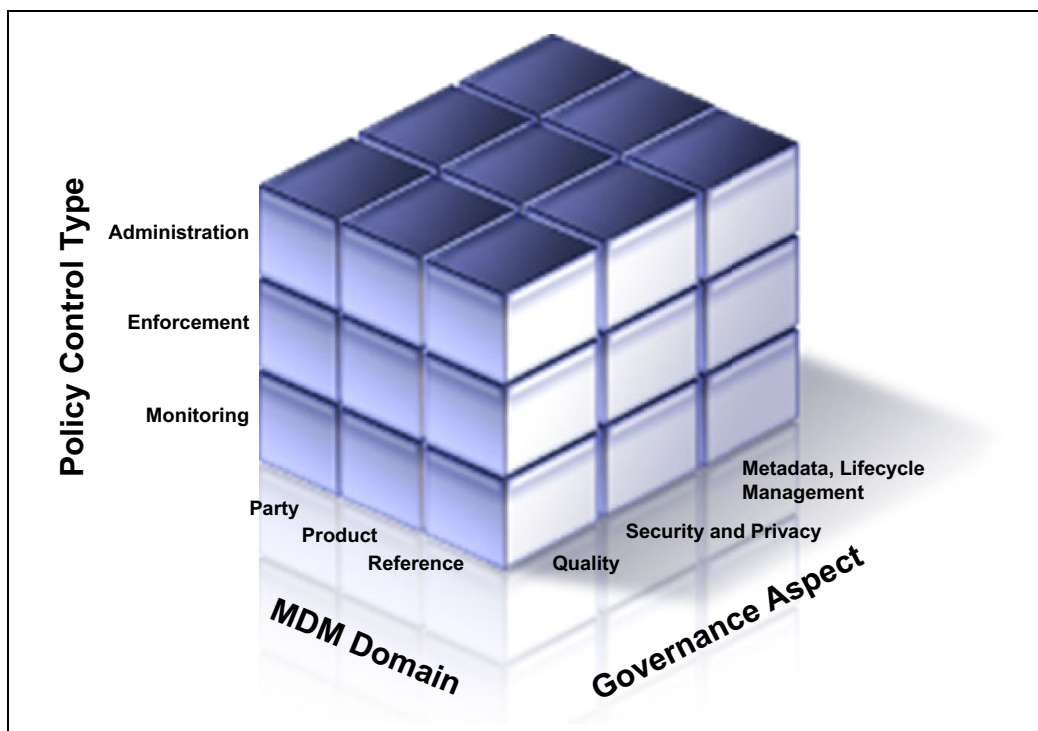


Figure 3-1 Multidimensional approach to master data governance

As illustrated in Figure 3-1, policy management is addressed by the policy administration, enforcement, and monitoring components. All aspects of master

data governance are addressed. The governance aspects include data quality and stewardship for master data, data visibility, privacy, and security (especially critical for customer data), and business and technical metadata.

The master data governance council is a cross-functional leadership team. This team works to approve and establish policies that dictate how master data is captured, managed, propagated, and used across the enterprise to achieve short-term and long-term goals. The pursuit of master data governance is an ongoing effort, rather than a one-time project.

Data stewards are responsible for everyday operations that enforce the guidance and policies of the master data governance council. Policy enforcement in mature data governance can require complex data stewardship remediation processes to adequately support business and master data governance requirements with agile and flexible data flows. Traditionally built applications might be unable to provide the required level of agility and flexibility in administration of master data governance policies and their remediation. This requirement puts forward the need to use business process management (BPM) software and templates that are integrated with MDM.

Master data governance is part of an information governance or data governance discipline. The primary focus is on master data implementations where modern MDM patterns are used with MDM data hub technologies. If an enterprise establishes a data governance council, a section of this council that specializes on MDM can play a role on the master data governance council. Conversely, if a data governance council is not established in the enterprise, master data governance can become a seed and starting point for an enterprise data governance council.

Most information governance and data governance methodologies lack a master data focus and do not reflect the specifics of modern MDM hub implementations. Few organizations achieve high master data governance levels of maturity. The percentage is even lower than for data governance in general.

3.2 Data governance maturity

The IBM Data Governance Council defines five levels of data governance maturity:

- Level 1: Initial

Ad hoc operations rely on individuals' knowledge and decision making.

- Level 2: Managed

Projects are managed but lack cross-project and cross-organizational consistency and repeatability.

- ▶ Level 3: Defined
Consistency in standards across projects and organizational units is achieved.
- ▶ Level 4: Quantitatively Managed
The organization sets quantitative quality goals that use statistical and quantitative techniques.
- ▶ Level 5: Optimizing
Quantitative process improvement objectives are firmly established and are continuously revised to manage process improvement.

To achieve advanced levels of data governance maturity (levels 4 and 5), organizations should be able to define and set quantitative policies, to effectively administer them, and to enforce the policies through agile data quality and stewardship processes.

Innovative IBM software products and components can help enterprises establish the advanced levels of master data governance maturity. They can also enable and sustain proactive, agile, and efficient policy-driven data stewardship on the MDM initiatives and MDM powered projects and programs.

3.3 Master data quality

Master data quality relies on the following components to effectively manage master data:

- ▶ Policies
- ▶ Processes
- ▶ Metrics and KPIs

3.3.1 Policies

One goal of master data governance is to enable a master data governance council to define and issue master data quality policies. Policies or sets of business rules quantify the policy compliance criteria and define how certain levels of data quality must be achieved. *Hard policies* prevent a record from being saved if a policy is not met and prevent more flexible *soft policies*. By using soft policies, a user can save records even if a record is not compliant with data governance policies. The noncompliant records are routed to a data stewardship inbox for resolution.

Figure 3-2 illustrates a typical set of master data quality policies.

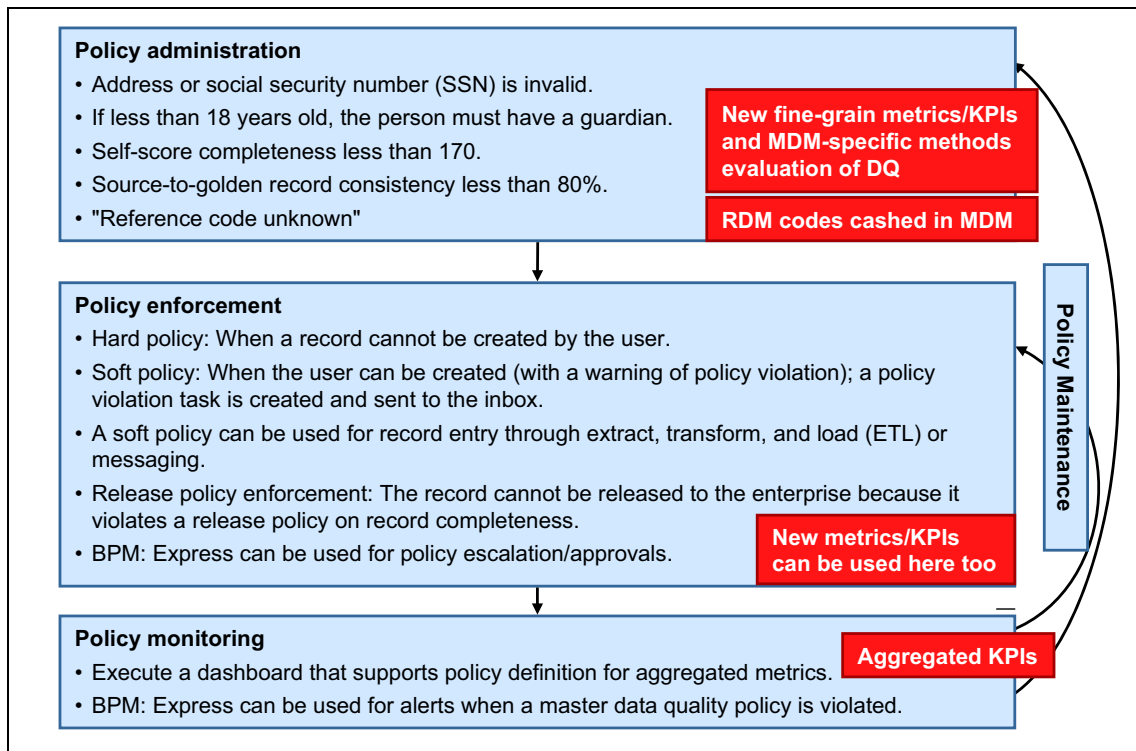


Figure 3-2 Master Data Policy management

In IBM InfoSphere Master Data Management V10.1, policy administration and policy enforcement are implemented within IBM Business Process Manager Express. Policy monitoring is implemented in the Master Data Policy Dashboard. IBM Cognos® is used to report on the data quality metrics and their compliance with established policy targets.

3.3.2 Processes

Also known as *procedures*, *workflow*, or *practices*, master data quality processes define how policies are to be implemented. Usually processes refer to the role or job function that is responsible for taking an action. They might also specify certain systems, screens, or forms that users in those roles must read, follow, or complete. The processes also address how exceptions are managed, which is typically by starting a separate process.

Exceptions: Some of the exceptions might be common across many different policies and processes. Therefore, you must ensure that, if changes are made to the parent processes and policies, you keep them sync.

Processes might specify certain time constraints for a process. For example, they might state that certain actions must be accomplished within a number of hours, minutes, or seconds. Processes might also specify escalation paths for higher-level approvals by people in other roles. The processes might start other processes. Therefore, use care to ensure that the processes and procedures are clearly articulated, leave little, if any, room for misinterpretation, and are as complete as possible.

Processes are also meant to be living specifications. That is, as experience is gained by practitioners, as systems and regulatory guidelines change, and as other factors come into play, policies and procedures might change.

At a high level, two categories of processes are used with MDM. The first category includes business processes that use master data, such as the following examples:

- ▶ Global account opening and client onboarding
- ▶ Patient registration
- ▶ Customer relationship management (CRM)
- ▶ Building the content of complex data warehousing dimensions
- ▶ Product definition
- ▶ Development and catalog management
- ▶ Industry-specific and function-specific applications

The second category includes data quality and stewardship processes. These processes, despite differences in customer requirements, can be defined in a general way. This section focuses on the following data quality processes:

- ▶ The *Benchmark Maintenance process* is responsible for the creation and maintenance of the golden record. The notion of the enterprise-wide trusted record, which is referred to as the *golden record*, is one of the key concepts in MDM. Data governance experts agree that an enterprise has reached an important milestone in its master data governance journey when it can agree on the golden record for customer, product, location, service, and other master entities.
- ▶ The *Benchmark Proliferation process* ensures that all enterprise systems upstream and downstream of the MDM data hub use the golden record that is created and maintained in the Benchmark Maintenance process. The importance of this process is fairly clear. Why does the enterprise need the golden record if it is not used consistently across the organization? Unfortunately enterprises that implemented MDM often lack a properly

implemented Benchmark Proliferation process, which limits the usefulness of the MDM solution.

Each process, which is shown in Figure 3-3, requires quantitatively defined policies and agile configurable workflows.

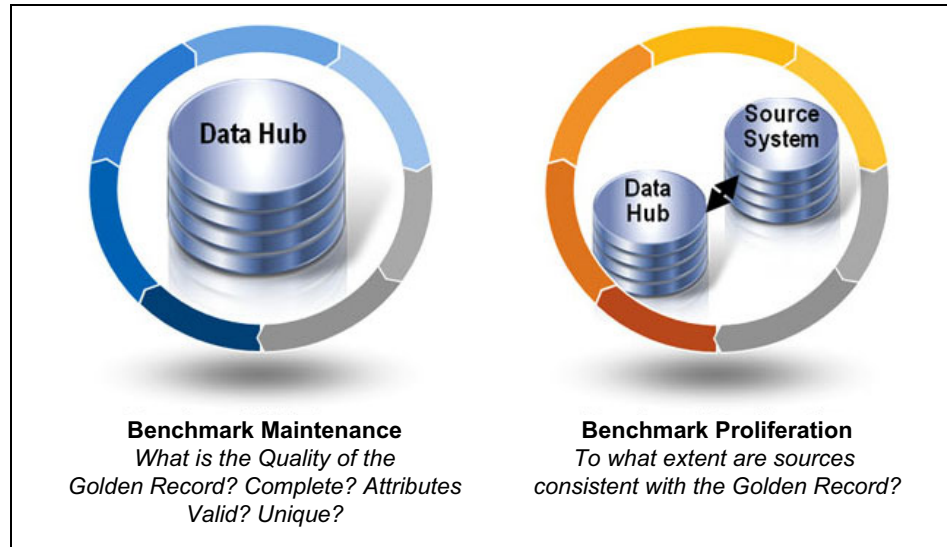


Figure 3-3 Master data quality processes

3.3.3 Metrics and KPIs

Data governance is a control discipline. A master data governance council initiates its control through master data quality policies. Metrics are necessary to the degree that the master data governance council wants to monitor and measure the levels of quality and consistency that are being achieved, in accordance with policies. For example, a policy states that a goal of completeness or uniqueness must be achieved, or the number of overlays must be reduced. Then, the metrics are the measurements that are taken so that managers can monitor trends in achieving the goals of the policies.

The metrics in Table 3-1 can be used as KPIs for the data quality processes.

Table 3-1 Primary metrics that drive data quality for master data

Metric	Quality process	Description
Estimated number of false positives in the GOLDEN RECORD SET	Benchmark Maintenance	Computed with the use of Receiver Operating Characteristic (ROC) curve that was originally developed in Signal Detection Theory. Provides an estimate for the false positive rate in the GOLDEN RECORD (Composite View). For mission-critical applications (such as healthcare and financial services), the false negative rate is expected to be in the range 10^{-7} - 10^{-8} .
Estimated number of false negatives in the GOLDEN RECORD SET	Benchmark Maintenance	Computed with the use of ROC curve that was originally developed in Signal Detection Theory. Provides an estimate for the false negative rate in the GOLDEN RECORD (Composite View). For practical purposes, the false negative rate is often in the range of 10^{-4} - 10^{-5} .
Self-score Completeness of the GOLDEN RECORD SET	Benchmark Maintenance	The metric is computed by using scoring application programming interfaces (APIs) that were developed earlier to match pairs of records. Within this metric computation, each GOLDEN RECORD (Composite View) is matched to itself to quantify the amount of identifying information on the record. The concept is similar to Entropy. A self-score is expressed in the same units as the thresholds (autolink and clerical review).
Source to GOLDEN RECORD Consistency	Benchmark Proliferation	The metric is computed by using scoring APIs to match source system records (members) to the corresponding entity record (Composite View). Often organizations require at least 80% of consistency between each source and the golden record.
Uniqueness within sources	Benchmark Proliferation	The metric is used to identify duplicates within each source. Shows 100% in the absence of duplicates.

The policies that are defined by the master data governance council must be clearly defined. Mature data governance processes require quantitatively managed goals, advanced statistical metrics, and techniques. Quantitative process improvement objectives must be firmly established and continuously revised to manage process improvement.

The first two metrics, estimated number of false positives and estimated number of false negatives, determine the accuracy with which the golden record uniqueness is defined. A *false positive* indicates that an entity record should be separated into two customer records. A *false negative* indicates that two entity records should be linked because they represent one customer.

Advanced scoring algorithms are developed and used to quantify similarity of records for matching.

Scoring above a certain threshold, as illustrated in Figure 3-4, indicates that the two records belong to the same customer. This approach has been successfully used in probabilistic matching. The same scoring APIs can be used to quantify the completeness of the entity records. The metric that is obtained by scoring a record on itself is referred to as *self-score completeness*.

Figure 3-4 illustrates the difference between incomplete records in the lower part of the chart and a complete record set in which most of the records score high and the score distribution is shifted to the right side of the chart.

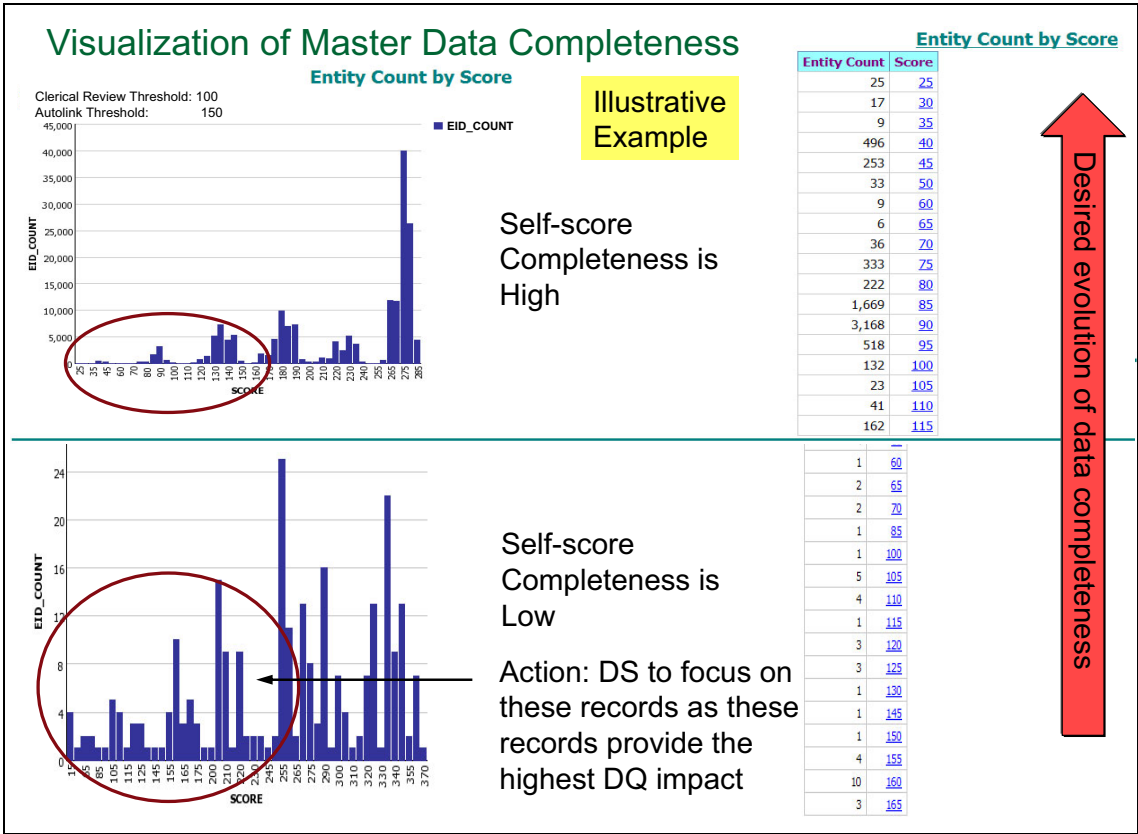


Figure 3-4 Master data completeness

The master data governance council directs data stewards to focus on the low scoring records, on the left side of the distribution in the lower graph in Figure 3-4. As a result of the data stewardship activity, the graph is expected to be transformed to look more like the distribution in the upper graph in Figure 3-4.

Figure 3-5 illustrates how the consistency between the data sources and the golden record is measured with the use of the scoring APIs.

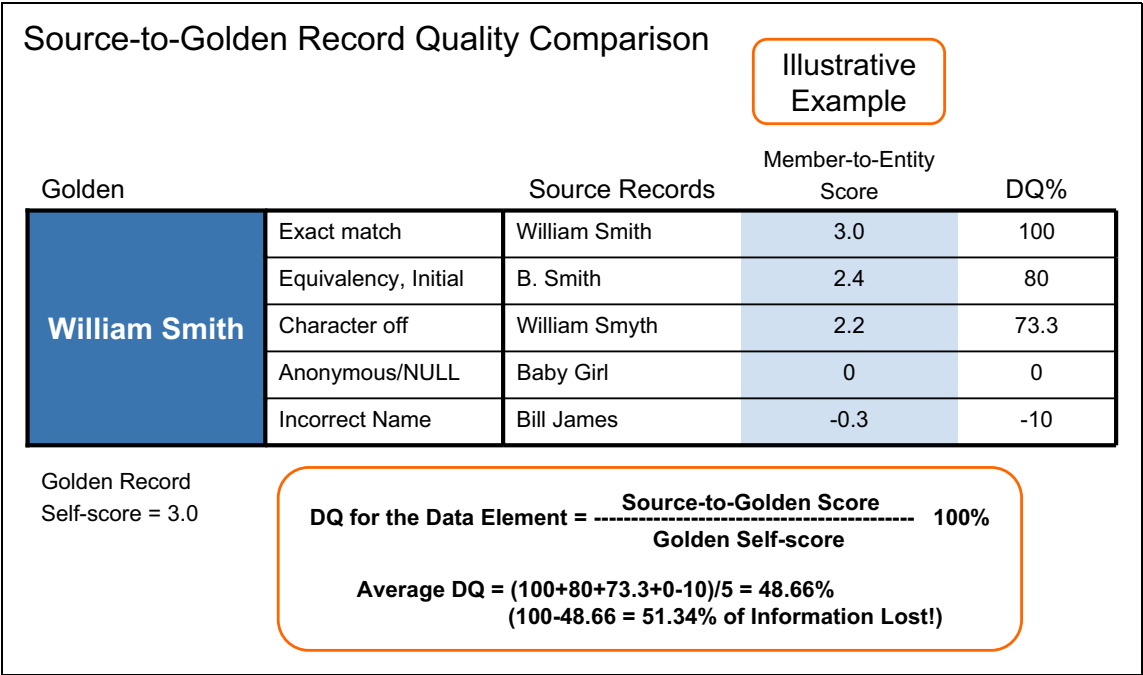


Figure 3-5 Master data policy metrics

When the attribute value is identical to the value in the golden record, the data source of the golden record shows 100% of the data quality. Data quality can be 0% when the attribute value in the source is blank or has an anonymous value, such as “Baby girl.” The data quality in the source can be negative if the attribute value is incorrect. The scoring APIs can recognize that “Bill” is a nick name for “William” and can identify and penalize small deviations such as missing or transposed characters.



Integration approaches and proven practices

With the capabilities provided by IBM InfoSphere Master Data Management (MDM), organizations can easily add master data to their enterprise processes (such as customer on boarding). These processes ensure that decision makers have the accurate, single version of truth for making informed decisions. These same capabilities bring process-orientated data stewardship to the organization, providing assurances that master data is compliant with enterprise process requirements.

This chapter includes the following sections:

- ▶ Business Process Manager tools
- ▶ Integration overview
- ▶ MDM integration approaches
- ▶ MDM Application Toolkit capabilities
- ▶ Overview of Process Designer

4.1 Business Process Manager tools

IBM Business Process Manager provides the following tools that facilitate rapid construction of business process across an organization:

- ▶ IBM BPM Process Designer
- ▶ IBM Process Portal

4.1.1 IBM BPM Process Designer

The IBM BPM Process Designer for IBM Business Process Manager is a rich development platform for creating business processes to be deployed to an IBM Business Process Manager instance. It offers the following features:

- ▶ A graphical interface to model business processes
- ▶ What you see is what you get (WYSIWYG) user interface building tools
- ▶ A rich set of prebuilt integration services that can be integrated to quickly create a robust business process

IBM Process Designer provides a rich environment so that process designers can develop, test, deploy, and version business processes across their organization. Process Designer also provides an advanced extension framework so that custom components can be easily plugged in to further extend the base capabilities.

4.1.2 IBM Process Portal

IBM Process Portal for IBM Business Process Manager is a runtime platform for deployed business processes. By using Process Portal, a user can authenticate with their credentials and use a runtime environment to run process steps that are assigned to the user. Process Portal provides web-based capabilities so that users can see all outstanding tasks that are assigned to them in one place. The web-based capabilities also give users the ability to start any new processes that they are authorized to launch. Graphical-based reporting capabilities are provided so that a user can monitor individual and team performance for a particular time.

4.2 Integration overview

Business processes inevitably need to interact with others systems across the enterprise. Whether the system is a customer relationship manager (CRM),

enterprise resource planning (ERP), or MDM system, data must be retrieved from these systems and decisions must be made on the data as part of a business process. A business process definition defines the various steps of this process. It defines when and how a user is required to interact with the process. It also defines the points in the process that are required to make calls into other systems to get data. How the business process integrates with the other systems largely depends on the requirements of the business process and the integration options that are provided by the other systems.

Figure 4-1 illustrates the role of a business process definition in defining the interactions between a business user and the remote systems that store the data for action as part of the business process.

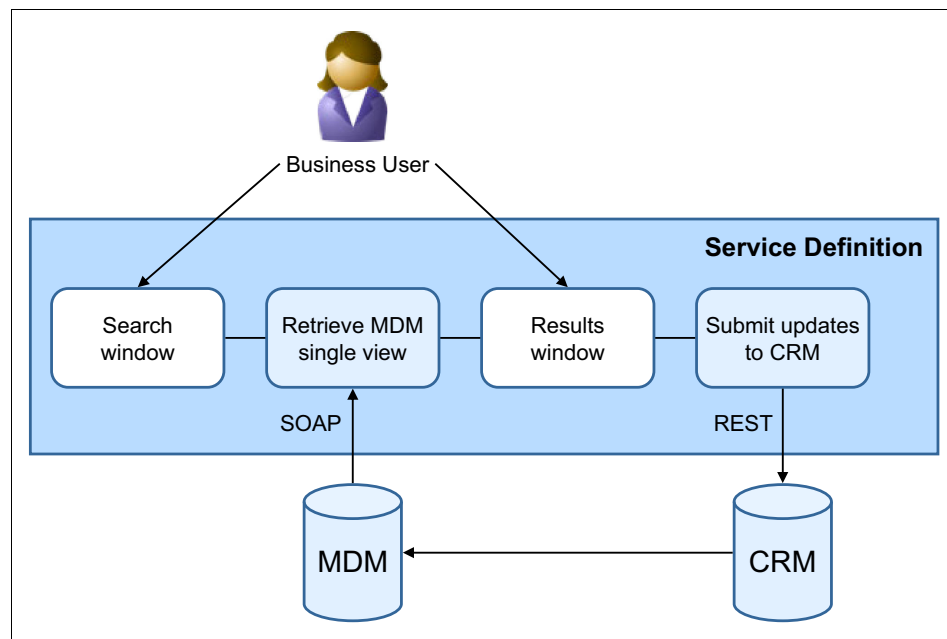


Figure 4-1 Defining interactions

IBM Business Process Manager provides a rich set of prebuilt integration components that facilitate interacting with other systems across an enterprise.

4.2.1 Java Integrator

The Java integration node is provided by IBM Business Process Manager. This node is a Java class to be provided by an integration developer who defines the operations that must be started on the consuming systems. The Java integration node can then be configured to run a method within the class to run a transaction

against the consuming system. Parameters can be defined and passed into the method that allows different values to be passed in, depending on the decisions that are made within the process. Return values from the running method can then be mapped back into the running process for action later in the process. Because the integration developer defines the logic in the provided methods, the Java Integrator provides a flexible approach to integration.

For more information about using the Java Integrator, see “Using the Java Integration step in an integration service” in the IBM Business Process Manager Information Center at:

http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/topic/com.ibm.wbpm.wle.editor.doc/modeling/topic/building_java_integration_service.html

4.2.2 Web Services Integrator

The Web Services Integrator node is also provided by IBM Business Process Manager. This node provides a simple way to integrate a business process with services that are provided by other systems that use a service-orientated architecture (SOA).

The Web Services Integrator allows a Web Service Definition Language (WSDL) from a consuming system to be uploaded to BPM Process Designer. Then, you use Process Designer to interrogate the WSDL and identify the business objects and transactions that were defined within the WSDL. Next, you can generate BPM business objects to represent the business objects that are defined in the WSDL. Finally, you can select which transaction from the WSDL to start. The business objects that were generated within BPM can then be mapped to the web service call to define the flow of data between the BPM engine and the consuming system.

For more information about using Web Services Integrator, see “Using a Web Service Integration step in an integration service” in the IBM Business Process Manager Information Center at:

http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/topic/com.ibm.wbpm.wle.editor.doc/modeling/topic/using_web_service_integration.html

Typically a business process might use a combination of the integration components throughout the business process. The decision to use one type of integration service over the other can be influenced by the following factors:

- ▶ Preferred integration method of connecting systems. One CRM system can provide only a Web Services API for connection, where another CRM system might provide only a Java interface.

- ▶ Skill set of the implementation team. A development organization might be more familiar with implementation of web services than with a proprietary Java API.
- ▶ Complexities in the web services that are provided by the consuming systems. The Web Services Integrator imposes certain limitations on the WSDL that are provided by the consuming system.

4.3 MDM integration approaches

When you consider a business process that involves operations against a master data hub, the choice about which integration approach to choose depends on Web Services Integrator and Java Integrator. InfoSphere Master Data Management provides rich interfaces that allow integration by using Web Services Integrator or Java Integrator components. Therefore, the choice can depend on the unique requirements of your business process and the environment in which it is operating.

4.3.1 MDM web services

The InfoSphere Master Data Management platform provides several web services interfaces so that external systems, such as the following examples, can take action on MDM data:

- ▶ MDM virtual eSOA interface
- ▶ MDM physical web services

The virtual eSOA interface can generate a set of web services to operate against the MDM data that is in a virtual MDM deployment. These web services are generated from the definition of the MDM domain and, therefore, are considered strongly typed. The WSDL files that define this generated interface can be imported into Process Designer. Also, business objects that relate to the virtual MDM hub can be generated for use within the business process. Transactions within the generated web services interface can be easily configured to be started from the web service integrator.

The MDM physical web services provide a definition of the entire physical MDM model. They include the entire definitions of all of the transactions that are available to be used within the physical MDM engine. The richness of the InfoSphere MDM platform means that the associated WSDLs are large and complex. The WSDLs that are ready for immediate use that describe the physical web services are too complex to be used by the BPM Web Services Integrator component. If the Web Services Integrator component is used with the MDM physical web services, use the Adaptive Services Interface (ASI) to customize

the default WSDLs. This approach provides a tailored set of business objects and transactions that are relevant to your specific requirements. WSDLs that are tailored with the ASI can then be imported into the Process Designer. Business objects that relate to the physical MDM hub can then be generated to be used within the business process. Transactions within the tailored web services interface can be easily configured to be started from the Web Service Integrator.

For more information about the ASI, search for *Tailoring the InfoSphere MDM Server Web service interfaces* in the InfoSphere MDM Information Center at:

<http://pic.dhe.ibm.com/infocenter/mdm/v10r0m0/index.jsp>

4.3.2 Virtual MDM Java interface

The IBM InfoSphere Master Data Management platform includes a Java API for interacting with the virtual MDM data model. This API works against the physical representation of the data within the virtual data model. Therefore, it can be used to operate against MDM member records, regardless of the type of business object that is defined in the virtual hub.

Custom Java code can be written by an integration developer to call this API and return data in the format they require. This custom Java code can be packaged within a Java archive (JAR) file and imported into Process Designer. After the Java code is imported, the Java Integrator component can be configured to call a particular method within this JAR file. It can also be configured to run a transaction against the virtual data that is stored in the MDM engine.

For more information about the virtual MDM Java API, see “Using the IBM Initiate SDK for Java and Web Services” in the InfoSphere MDM Information Center at:

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/nav/3_7_1

4.4 MDM Application Toolkit capabilities

The IBM InfoSphere Master Data Management platform includes the MDM Application Toolkit. This toolkit provides a suite of value-add capabilities to facilitate construction of process-orientated MDM powered applications. The MDM Application Toolkit takes advantage of the extensible nature of Process Designer to provide more capabilities to organizations that build MDM applications within IBM Business Process Manager.

The MDM Application Toolkit includes the following features:

- ▶ Prebuilt business objects that represent commonly described MDM business objects
- ▶ Business objects that represent the generic virtual MDM data types for use against the virtual MDM Java interface
- ▶ Additional UI controls
- ▶ Several prebuilt integration services to perform common operations against MDM data

Any combination of these capabilities can be used with the base capabilities that are provided by Process Designer to accelerate construction of business processes.

4.5 Overview of Process Designer

All process development takes place in IBM BPM Process Designer. The Process Designer is installed in your environment when Process Center is installed and is associated with it. Process Designer is the rich client (based on Eclipse) that enables process authoring. Before you start Process Designer, you must run the IBM WebSphere® Application Server instance on which the Process Center is installed.

4.5.1 Starting Process Designer

You can start Process Designer in one of the following ways:

- ▶ Double-click the **BPM Process Designer** Windows desktop shortcut.
- ▶ From the Windows desktop, click **Start** → **BPM** → **BPM Process Designer**.

After Process Designer is started, you are prompted to enter a valid user ID and password. If you do not already have a user account, contact your IBM BPM administrator. By default, BPM creates an administrative user ID (user ID is `admin` and password is `admin`), which you can use to open the authoring environment.

When you log in, you are connected to the Process Center that is designated during installation of Process Designer, as shown in Figure 4-2.

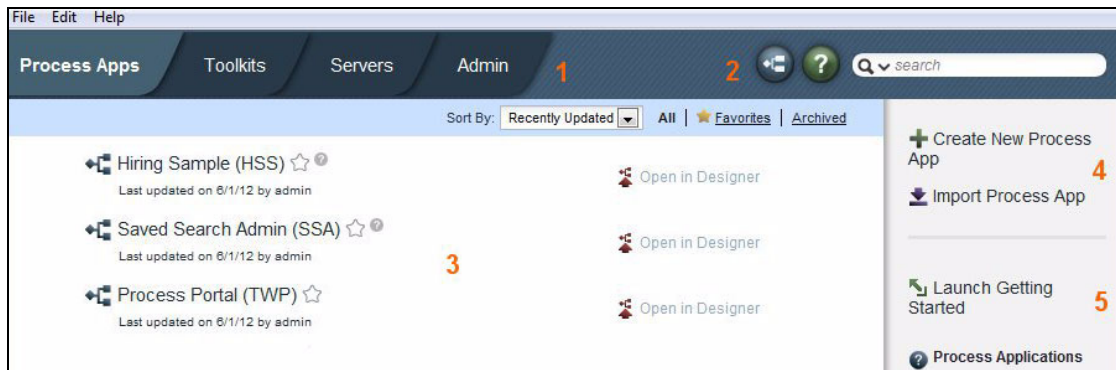


Figure 4-2 The Process Center that is designated during the installation of Process Designer

Table 4-1 describes the numbered parts of Process Designer that are shown in Figure 4-2.

Table 4-1 Parts of Process Designer

Part	Description
1	Click the available tabs to choose the type of item you want to create or manage: <ul style="list-style-type: none"> Click Process Apps to create and manage process applications. Click Toolkits to manage toolkits. Click Servers to manage the servers that are configured in your environment. Click Admin to manage access to the Process Center repository.
2	Click Designer to open the Designer interface in Process Designer. Click Help to open the online help for Process Designer.
3	The main area of the Process Center Console shows the items that you currently manage, such as process applications, snapshots, or servers. Click the All , Favorites , or Archived options to filter the items that are displayed. Click an item to view and manage its settings. For example, you can click one of the listed process applications to view and manage its snapshots, history, and general settings. To open a process application in the Designer, click Open in Designer for the process application that you want to access.
4	Use these options to create a process application or to import an existing one.
5	Help information is offered throughout the Process Center Console. Click the link that is shown to learn more about the subject, which in this case is the process application.

4.5.2 Process authoring environment

You can use the Process Designer interface to create process applications and their underlying implementations. For example, you can create services, run or test process applications, export process applications, and create versions of process applications.

When the process application is opened in Process Designer, you see the necessary tools and views to help you create service implementations and user interfaces (Figure 4-3).

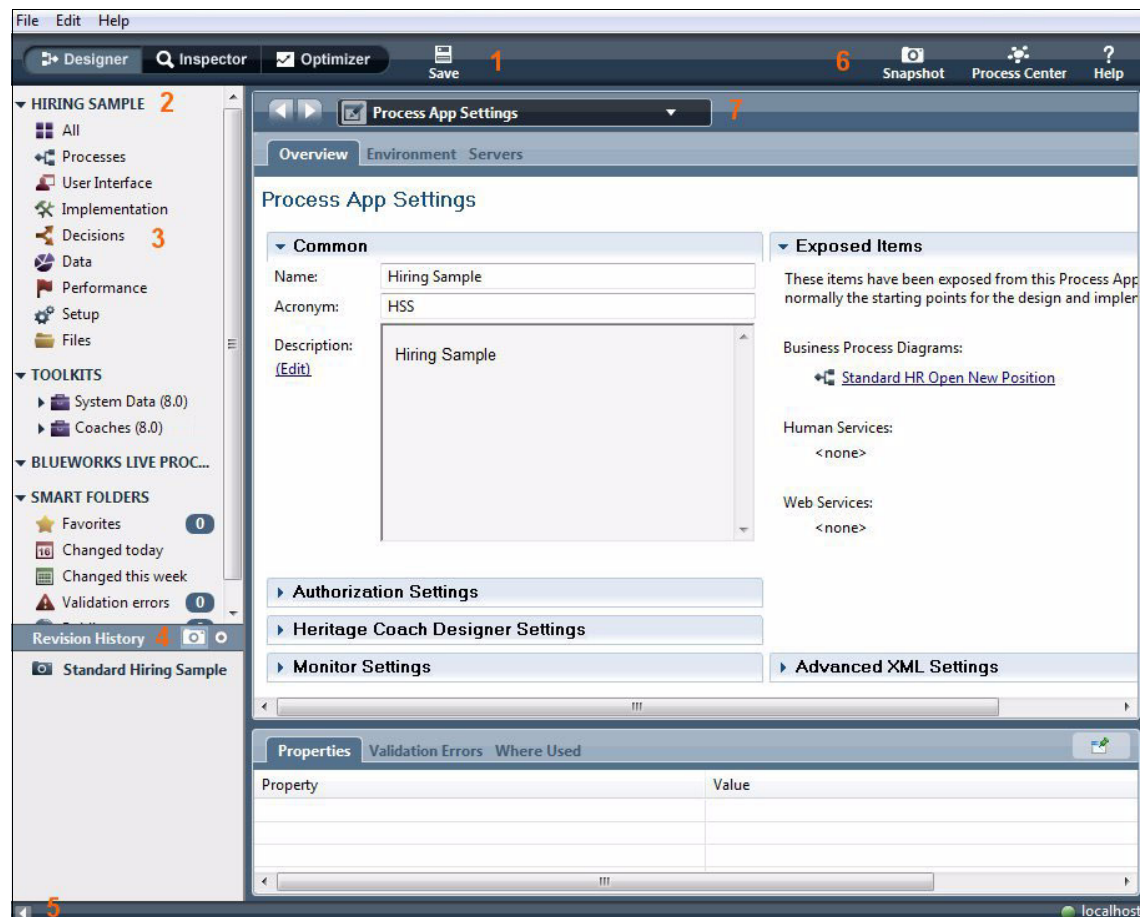


Figure 4-3 The process application in the Process Designer

Table 4-2 describes the numbered parts of the process authoring environment that are shown in Figure 4-3 on page 35.

Table 4-2 Parts of the process authoring environment

Part	Description
1	Click the appropriate button to open the interface that you want in Process Designer, including the Optimizer and Inspector views.
2	This item in the navigation tree shows the process application that is open. In this sample, the Hiring process application is open.
3	This list shows the types of library items that are included in the currently open process application. Click a category, such as Processes, to see the processes that you can open and alter.
4	The Revision history shows the history for the currently open process application. In this sample, a user recently added items to the open process.
5	Click the left arrow button in the lower-left corner of the window to hide the library and revision history. You can click this button to toggle back and forth if you want to alternately hide and view the library and revision history. This toggle control is available in all Process Designer interfaces including Designer, Optimizer, and Inspector.
6	Use the icons in the upper-left side of the window to create snapshots, to access the Process Center Console, or to access online assistance.
7	Shows the library item currently open for editing in the Designer. In this sample, the user has a process open and is working in the diagram, palette, and properties to create the steps of the process.

For more information about IBM Process Designer, see the IBM Business Process Manager V8.0 Information Center at:

<http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/index.jsp>



Introduction to the InfoSphere MDM Application Toolkit

The IBM InfoSphere Master Data Management (MDM) Application Toolkit for business partner management (BPM) offers a rich set of capabilities to further enhance the simple integration experience. These capabilities help organizations build MDM-powered applications and business processes. They help organizations put master data into the hands of business users and improve business processes by integrating them with complete and accurate master data.

Historically MDM was a back-office integration platform. Organizations had to do their own development work to make MDM capabilities available in user applications. For example, assume that an organization wanted to enhance their customer relationship management (CRM) system so that whenever a user added a customer, CRM first checked with MDM to determine whether the customer existed. This process required rewriting portions of the CRM application to call MDM services to do this check. The InfoSphere MDM Application Toolkit makes building these MDM-powered applications and business processes easier.

The chapter describes the components of the InfoSphere MDM Application Toolkit and how to use them to build MDM powered applications and business processes. It includes the following sections:

- ▶ Components of the InfoSphere MDM Application Toolkit
- ▶ Installing the InfoSphere MDM Toolkit
- ▶ Building applications with the InfoSphere MDM Application Toolkit for BPM

5.1 Components of the InfoSphere MDM Application Toolkit

The InfoSphere MDM Application Toolkit contains a set of BPM building blocks that you can incorporate into your business process to facilitate rapid construction of process-oriented MDM powered applications.

5.1.1 Architectural overview

The InfoSphere MDM Application Toolkit provides a design-time set of MDM-specific tools to the BPM Process Designer. Experienced users of Process Designer can use these tools with the more familiar standard BPM tools to build their applications that operate against their master data.

The InfoSphere MDM Application Toolkit can be installed into new or existing processes. It includes a configurable Representational State Transfer (REST) service to deploy to a WebSphere Application Server instance that is running the BPM server. By using this approach, some of the UI controls in the InfoSphere MDM Application Toolkit can retrieve data from an InfoSphere MDM Server in the format that is required by some of the InfoSphere MDM Application Toolkit UI controls. When the processes are running, the InfoSphere MDM Application Toolkit Widgets controls and the MDM REST service work together to operate against the MDM engine in use as illustrated in Figure 5-1 on page 39.

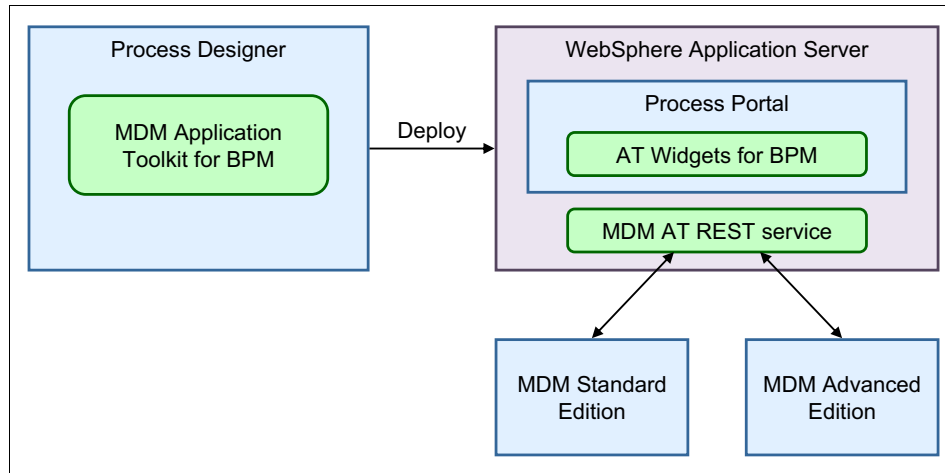


Figure 5-1 InfoSphere MDM Application Toolkit architecture

The InfoSphere MDM Application Toolkit has several capabilities (Figure 5-2).

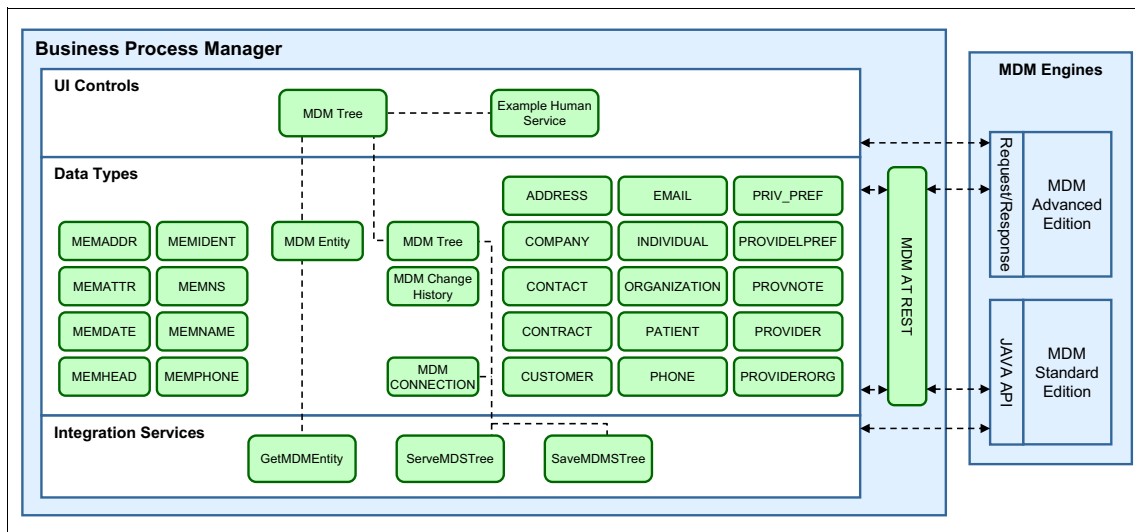


Figure 5-2 Capabilities in the InfoSphere MDM Application Toolkit

Figure 5-2 highlights the following key capabilities in the InfoSphere MDM Application Toolkit:

MDM Data Types For use when you define BPM variables that describe your MDM business objects.

- MDM UI Controls** Helps you to display your MDM data in a manner that is suited to MDM applications.
- MDM Integration Services**
 - Help facilitate easy integration between the MDM and BPM engines.
- MDM REST service** Helps to provide extra integration services to the UI controls.

5.1.2 MDM data types

A first step in building an MDM powered solution with BPM is to define the BPM business objects that are needed to manage the master data throughout the process flow. The InfoSphere MDM Application Toolkit makes this step easy by providing prebuilt business objects that mirror commonly used MDM data types and data model components. The following MDM data types are commonly used:

- ▶ Business objects for virtual MDM

The InfoSphere MDM Application Toolkit includes the following prebuilt business objects. These objects mirror the domain templates that are provided by virtual MDM and the attribute types that are used by these domain templates.

- Patient
- Provider
- Individual organization
- MEMADDR
- MEMATTR
- MEMDATE
- MEMNAME

For a complete list of these business objects for MDM Application Toolkit 10.1, see the “Business objects” topic the IBM InfoSphere MDM Version 10.1 Information Center at:

http://pic.dhe.ibm.com/infocenter/mdm/v10r1/topic/com.ibm.mdshs.mdmtoolkit.doc/topics/bpm_integration_biz_objects.html

- ▶ Business objects for physical MDM

The InfoSphere MDM Application Toolkit includes the following prebuilt business objects that reflect commonly used entities from the physical MDM data model:

- Customer
- Company
- Address

- Name
- Contact preferences

For a complete list of these business objects for MDM Application Toolkit 10.1, see the “Business objects” topic the IBM InfoSphere MDM Version 10.1 Information Center at:

http://pic.dhe.ibm.com/infocenter/mdm/v10r1/topic/com.ibm.mdshs.mdmtoolkit.doc/topics/bpm_integration_biz_objects.html

► Business objects for MDM-specific UI controls

The InfoSphere MDM Application Toolkit includes some MDM-specific UI controls (see 5.1.3, “Hierarchy widget overview” on page 41). These UI controls use the following prebuilt business objects:

- MDMTree
- MDM Entity

5.1.3 Hierarchy widget overview

The InfoSphere MDM Application Toolkit includes a UI control to manage relationship data as a tree or hierarchy. After you import the InfoSphere MDM Application Toolkit into Process Designer, the MDM Tree control is available within the pallet of controls in BPM *coaches* (or user interfaces).

Coach: A *coach* is a window for the user to indicate when in the process to perform an operation on data so that the process can proceed to the next step.

The MDM Tree control is a full-featured tree widget and supports a wide range of use cases. The MDM Tree control includes the following features:

► Lazy loading

The tree retrieves data from the MDM back end through an MDM REST service as needed. As you expand nodes in the tree, the control makes extra requests for more data from the REST service. Because the tree does not need to retrieve all of its data initially, the initial rendering of the tree is fast. Each subsequent retrieval of data is also fast because it retrieves only the information that is necessary to expand the tree by an extra level.

► Multiple relationships and node types

You can use a single tree to see multiple types of relationships and nodes. For example, each person node might have child nodes for “boss of,” “reports to,” and other relationships.

- ▶ **Filtering relationship types**
From the tree, users can filter the relationship types that are displayed in the window by using a simple menu option.
- ▶ **Moving nodes**
By using the tree, users can change relationships by dragging nodes to other places in the tree.
- ▶ **Deleting nodes**
Users can use the tree to remove relationships by deleting nodes in the tree.
- ▶ **Validation**
The tree control prevents users from making changes that might violate constraints by preventing the dropping of nodes into invalid locations in the tree. For example, a company node cannot be dropped under a “boss of” relationship that was meant to show a manager’s employees.
- ▶ **Undo and redo**
With the tree, users can undo changes and redo them as needed.
- ▶ **Export To CSV**
A user can export the data that is displayed within the hierarchy into a comma-separated value (CSV) file.

Figure 5-3 shows an example of the MDM tree with customer data.

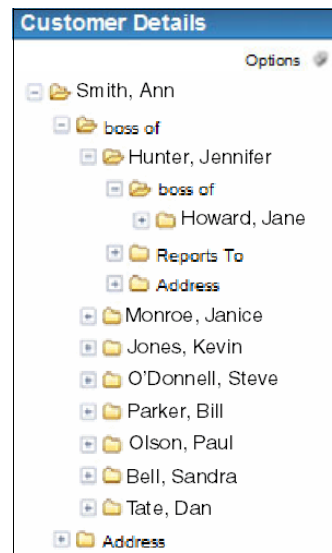


Figure 5-3 MDM tree with customer data

Tree styles

The MDM Tree control shows data that the MDM REST service provides. This data can be relationship data with multiple relationships as categories, or it can be relationship data with a single relationship such as a corporate hierarchy. If the relationship data has multiple relationships, each relationship type has a category node. Child nodes under the category node indicate entities that have that relationship. If the relationship data has a single relationship, each subsidiary of a company is shown directly under the node that represents the parent company. There is no need for category nodes to indicate the relationship type.

Component architecture

The MDM Tree works with the other capabilities that are provided by the InfoSphere MDM Application Toolkit to make it as simple as possible to integrate into a BPM-based application. The MDM Tree accepts MDM data in the form of an MDM_Entity. The Get MDM Entity integration service calls into the BPM REST Service of the InfoSphere MDM Application Toolkit to retrieve the MDM data as defined by an XML configuration file. The MDM data is then converted into an MDM_Entity data type. The data type is then passed into the MDM Tree UI control for display. After the data is displayed in the MDM Tree, more data is lazy loaded into the MDM Tree through subsequent calls into the MDM REST service. These calls are initiated each time that the user expands a tree node.

Changes to the data that are displayed in the MDM Tree are not persisted to the MDM engine until a user clicks the **Add** button to signal Process Designer to add the changes to the **Coaches** tab. To facilitate this behavior, the MDM Tree must be bound to a BPM variable of type MDM_Tree. This button must be configured to call the Save MDS Tree or Save MDMS Tree, depending on which edition of the MDM engine is in use. The MDM_Tree variable that was bound to the MDM Tree UI Control should then be passed with the connection details into the Save MDS Tree or Save MDMS Tree services as a parameter. Figure 5-4 on page 44 illustrates this process.

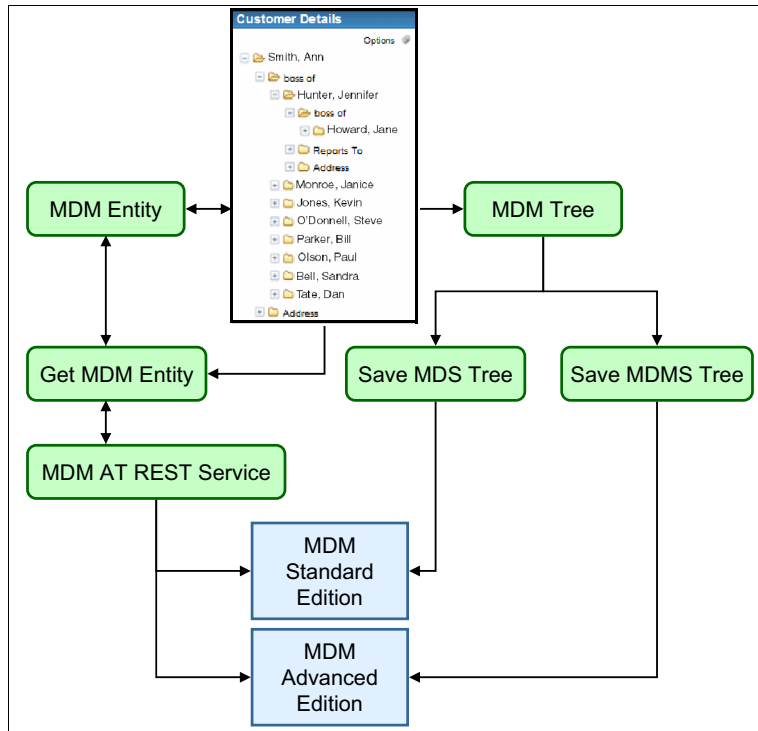


Figure 5-4 The MDM Tree designed to work with other capabilities

For more information about how to use the MDM Tree with the other capabilities provided by the MDM Application Toolkit, see 5.3.3, “The MDM Tree UI control” on page 53.

5.1.4 Integration services

The InfoSphere MDM Application Toolkit includes prebuilt BPM integration components that you can use to interact with an IBM InfoSphere MDM engine. These integration services facilitate rapid construction of process-orientated MDM powered applications by abstracting integration coding away from application developers. The integration services that are currently included within the MDM Application Toolkit focus specifically on simplifying the manipulation of data for display and updating within the MDM tree.

The InfoSphere MDM Application Toolkit includes the following integration components:

Get MDM Entity	An integration component to retrieve MDM tree data from the MDM engine. This integration component returns its data by using the MDM Entity business object. This data is then typically bound to an MDM Tree UI component so that users can view and manipulate the tree data in the format that is expected by the MDM Tree UI component.
Save MDS Tree	An integration component to save changes to relationships that are made by using the MDM Tree control. This service handles the complexity of determining which services to call on the MDM engine, based on the changes that are made by the user to the data displayed within the MDM tree. Use this service when working against MDM Standard Edition Engines.

5.1.5 Overview of the MDM REST service

The InfoSphere MDM Application Toolkit includes an MDM REST service for retrieving MDM data from an IBM InfoSphere MDM solution. The MDM REST service is used by the MDM Tree control of the InfoSphere MDM Application Toolkit to lazy load data as a user navigates a tree. The behavior of the REST service can be configured by using an XML file. It can also be extended by Java developers who want to have complete control over how and what data is retrieved.

The MDM REST service provides a pluggable framework to allow different back-end services to be configured and started by the MDM Tree (see 5.1.3, “Hierarchy widget overview” on page 41). This capability is provided through adapters that can be configured through an XML configuration file to call the required services. The REST service comes configured with four preconfigured adapters.

Table 5-1 shows the key features of the MDM REST service.

Table 5-1 Key features of the MDM REST service

Feature	Description
MDMSHierarchyWithCategories	A service to retrieve multiple relationships from the Physical MDM engine. The data that is returned by the service includes a category node for each relationship type. The service also returns a category node for showing suspected duplicates.
MDSHierarchyWithCategories	A service to retrieve multiple relationships from the virtual MDM engine. The data that is returned by the service includes a category node for each relationship type. The service also returns a category node for showing suspected duplicates.
MDMSHierarchy	A service to retrieve a single-relationship hierarchy from the physical MDM engine. The data does not include category nodes.
MDSHierarchy	A service to retrieve a single-relationship hierarchy from the virtual MDM engine. The data does not include category nodes.

You can also use the XML configuration to define other characteristics of the MDM tree. The following characteristics can be configured:

- ▶ Connection information to connect to the appropriate MDM engine
- ▶ The relationship types to include
- ▶ The label to use for each relationship type
- ▶ The fields to use as the label for each entity node
- ▶ Whether to include suspected duplicates as a category

5.1.6 Mobile applications

IBM Business Process Manager provides an app for iOS devices. By using this app, which is available for download free of charge from the Apple App Store, a user with a supported mobile device can point their device to their organization's BPM server that hosts their MDM-powered application. The user can then access their BPM task list and take action on tasks that are routed to them from their handheld device. The InfoSphere MDM Application Toolkit, which is coupled with this iOS app, opens the MDM powered application to users of handheld devices, giving them greater access to key governance processes and reducing the wait time between process steps.

Figure 5-5 shows a typical human task that is running from a handheld device.

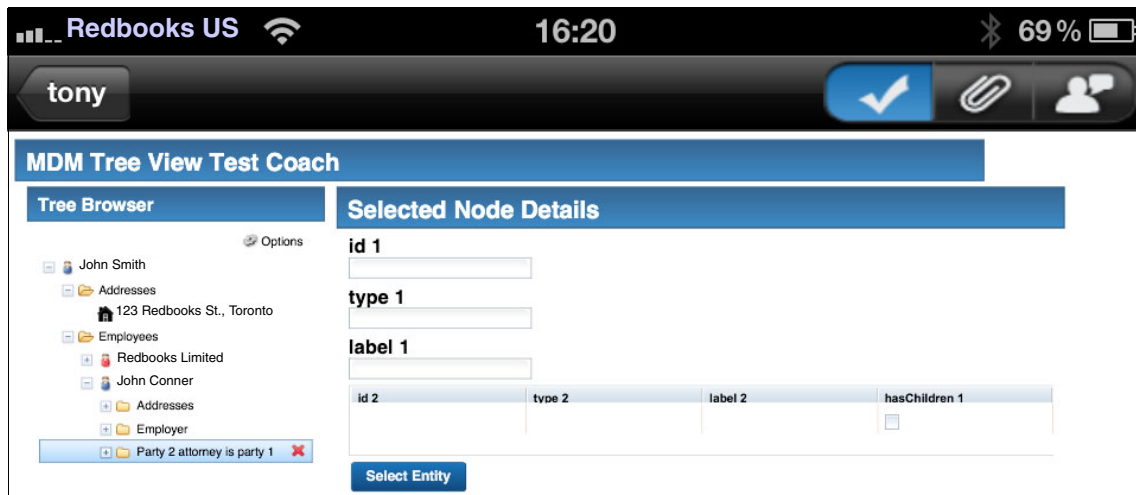


Figure 5-5 Accessing the BPM Server from the handheld device

The handheld device has the following features:

- ▶ Support for Apple iPhone and iPad
- ▶ Notifications of new tasks that require action
- ▶ Ability to initiate new workflows

The handheld device has the following prerequisites:

- ▶ Access to the BPM server from the handheld device, either on the same physical network or through a virtual private network (VPN)
- ▶ BPM V8 or later

5.2 Installing the InfoSphere MDM Toolkit

When you use the MDM installer to install IBM InfoSphere MDM, the installer places the InfoSphere MDM Application Toolkit components in your installation directory. After the installation, two files are placed on a disk (Table 5-2).

Table 5-2 Files installed

File name	Description
MDM_Application_Toolkit.twx	The main BPM controls to install into the BPM Process Designer for use in building your application.
MDMAT_BPM_REST_EAR.ear	The Enterprise Application File that contains the MDM REST service. This file must be deployed to the WebSphere Application Server instance that is used by your BPM engine.

5.2.1 Installing the MDM Application Toolkit .twx file

The InfoSphere MDM Application Toolkit components include the MDM_Application_Toolkit.twx file that represents the InfoSphere MDM Application Toolkit plug-in for BPM. To install this file:

1. Start Process Designer, and sign on.
2. Click the **Toolkits** tab to navigate the Toolkits view.
3. Click **Import Toolkit**.

4. Browse to the MDM_Application_Toolkit.twx file, and click **OK** (Figure 5-6).
5. Assign a name for your toolkit such as MDM Application Toolkit.
6. Specify a version for your toolkit, preferably one that matches the version of InfoSphere MDM that you installed.

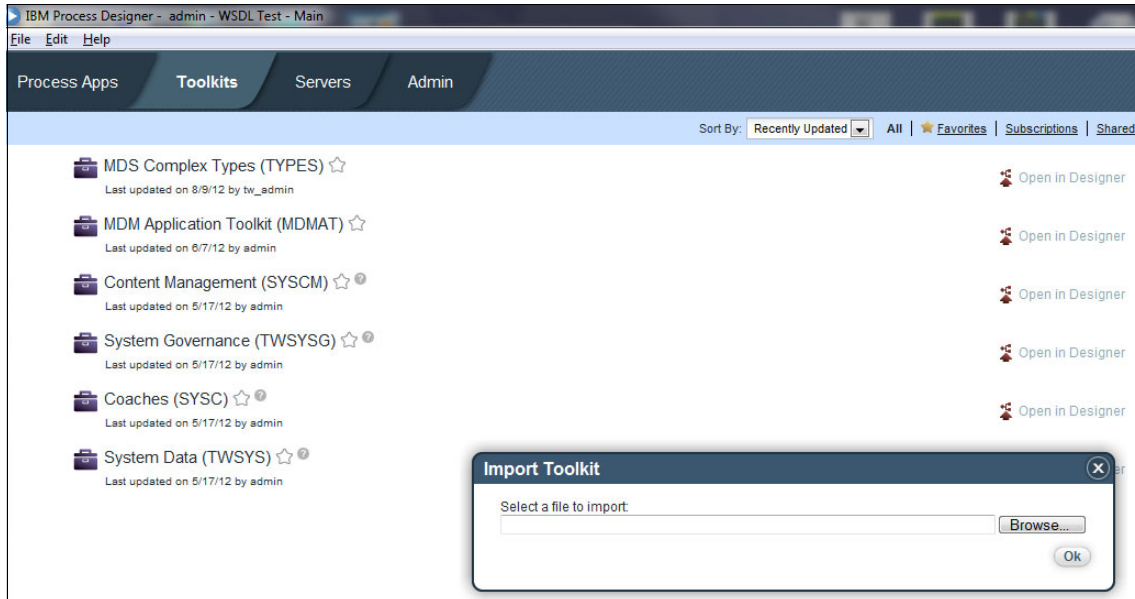


Figure 5-6 Browsing to the MDM_Application_Toolkit.twx file

In any business process where you want to use the InfoSphere MDM Application Toolkit to add a dependency (Figure 5-7):

1. Open your process in Process Designer.
2. In the left panel, hover over the **Toolkits** menu, and click the **+** button to add a dependency.
3. Click the **MDM Application Toolkit**.

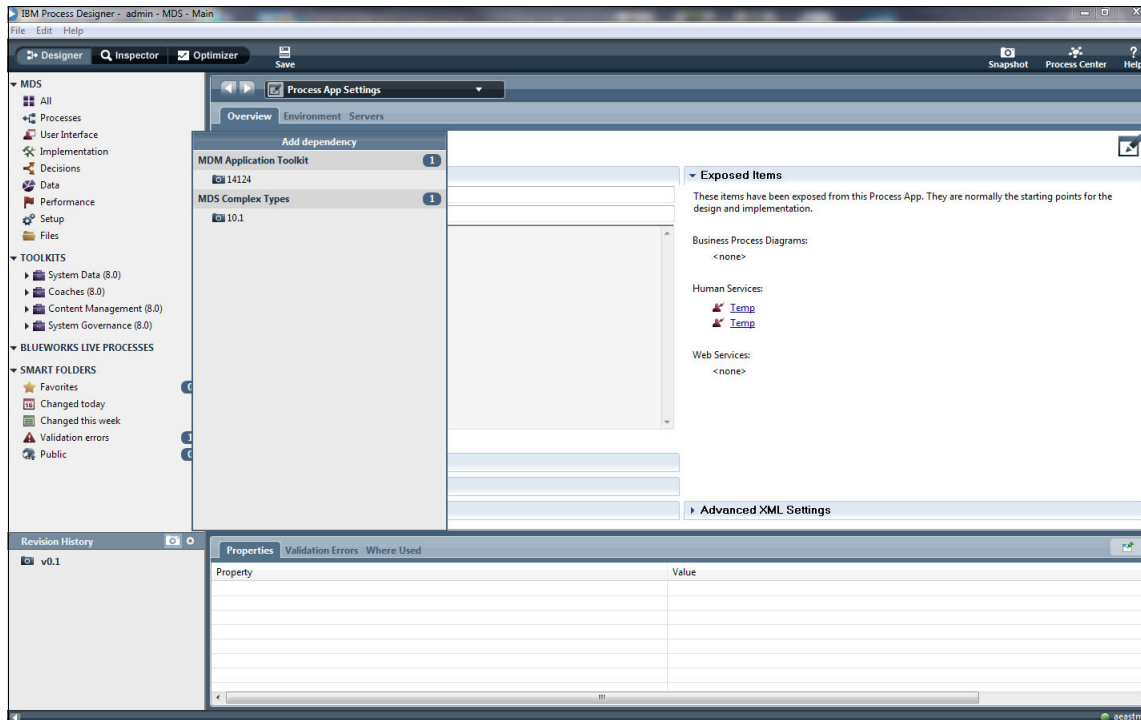


Figure 5-7 Adding a dependency

The MDM Application Toolkit components are now available for use within your process.

5.2.2 Installing the MDM REST Service

The MDM REST Service is a Java Platform Enterprise Edition (Java EE) enterprise archive (EAR) file. This EAR file must be deployed to the WebSphere Application Server instance to which the Business Process Manager Engine is deployed. Before you install the MDM REST service, make any changes to the `MDMConfig.xml` file that are required by your application.

To install the MDM REST service, you must have administrator access to the administrative console or to run the `wsadmin` scripts on the WebSphere Application Server. During the installation steps, you can use the default settings, unless you want to change them. By default, the `MDMAT_BPM_REST` is the context root for the deployed REST service.

For more information about how to configure the MDM REST service, see “Configuring the REST service for BPM” on page 57.

5.3 Building applications with the InfoSphere MDM Application Toolkit for BPM

As explained in 5.1, “Components of the InfoSphere MDM Application Toolkit” on page 38, BPM offers multiple components to help you build MDM-powered applications and business processes. These components fall into the following categories:

- ▶ MDM data types
- ▶ MDM UI controls
- ▶ MDM services

This section explains how to use each of these components.

5.3.1 MDM data types for creating BPM variables

When a business process uses data that comes from MDM or creates data that is targeted for BPM, you need to define BPM variables for that data. The InfoSphere MDM Application Toolkit includes predefined MDM data types that you can use to create BPM variables (Figure 5-8 on page 52).

For example, you are building a business process that interacts with an MDM deployment that uses the individual MDM domain template. In this case, you can use the predefined Individual business object of the InfoSphere MDM Application Toolkit that matches the data types of the individual MDM domain template. You can also add variables to these ready-to-use data types to customize your BPM variables to match any changes that you make to your MDM data model.

In addition to providing data types that match common MDM domains, the InfoSphere MDM Application Toolkit includes the building blocks, or individual attributes and fields, that make up each domain. You can use these building blocks to assemble your own BPM variables that reflect the MDM data that you want to use in your process.

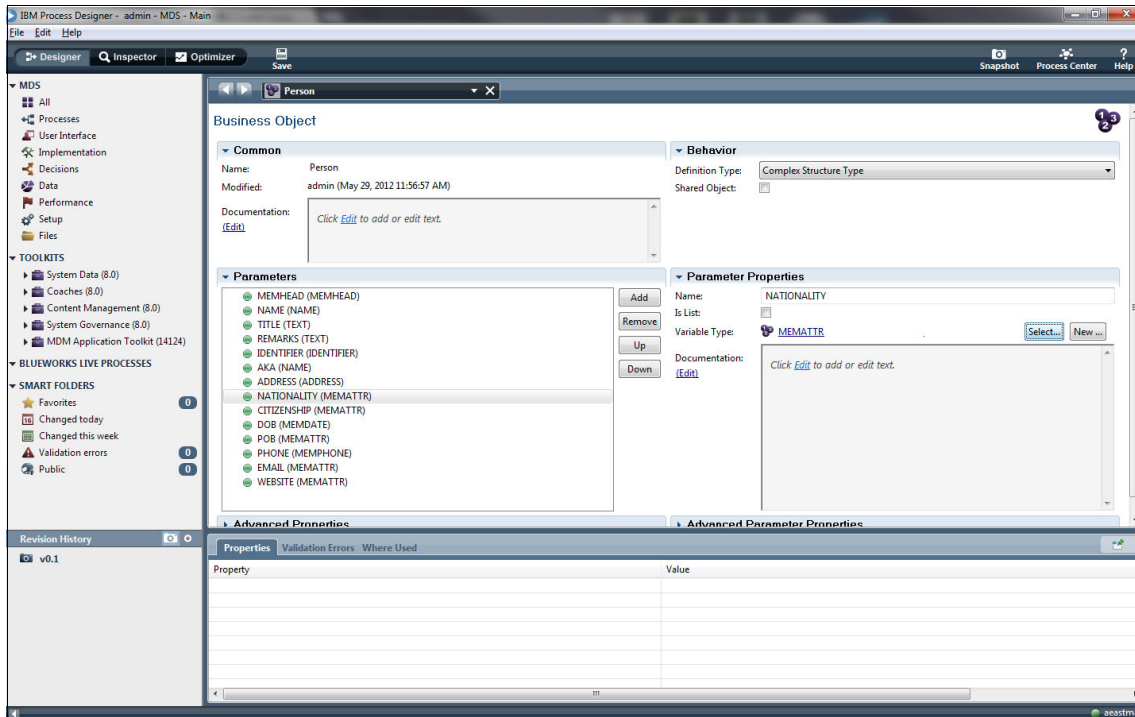


Figure 5-8 MDM data types

5.3.2 Customizing an InfoSphere MDM Application Toolkit data type

You can use the MDM data types from the InfoSphere MDM Application Toolkit as is by selecting them as the variable type for any variable you define. These data types were created to act as accelerators for building your applications. By using BPM, you can customize these data types to suit the specific requirements of your MDM powered applications.

To modify the default InfoSphere MDM Application Toolkit data types:

1. In the left navigation pane of Process Designer, expand **Toolkits**, and select **MDM Application Toolkit**.
2. Click **Data** to see the predefined data types.
3. Right-click any of the data types, select **Copy Item To**, and then click **Other Process App**.
4. Choose the process app to which you want to copy the data type.
5. Use Process Designer to edit the properties of the data type to suit your application.

5.3.3 The MDM Tree UI control

An application that includes an MDM Tree generally uses the following components:

- ▶ A variable of type MDMEntity
- ▶ The Get MDM Entity integration service
- ▶ A coach that uses the MDMTTree control
- ▶ An integration service to save changes to the tree

This section explains how to use each of these components.

MDMEntity data type

The MDMTree control uses the MDMEntity data type to store the data that it renders. To use the tree, you must first define a BPM variable of type MDMEntity. MDMEntity is a data type provided by the MDM Application Toolkit (Figure 5-9).

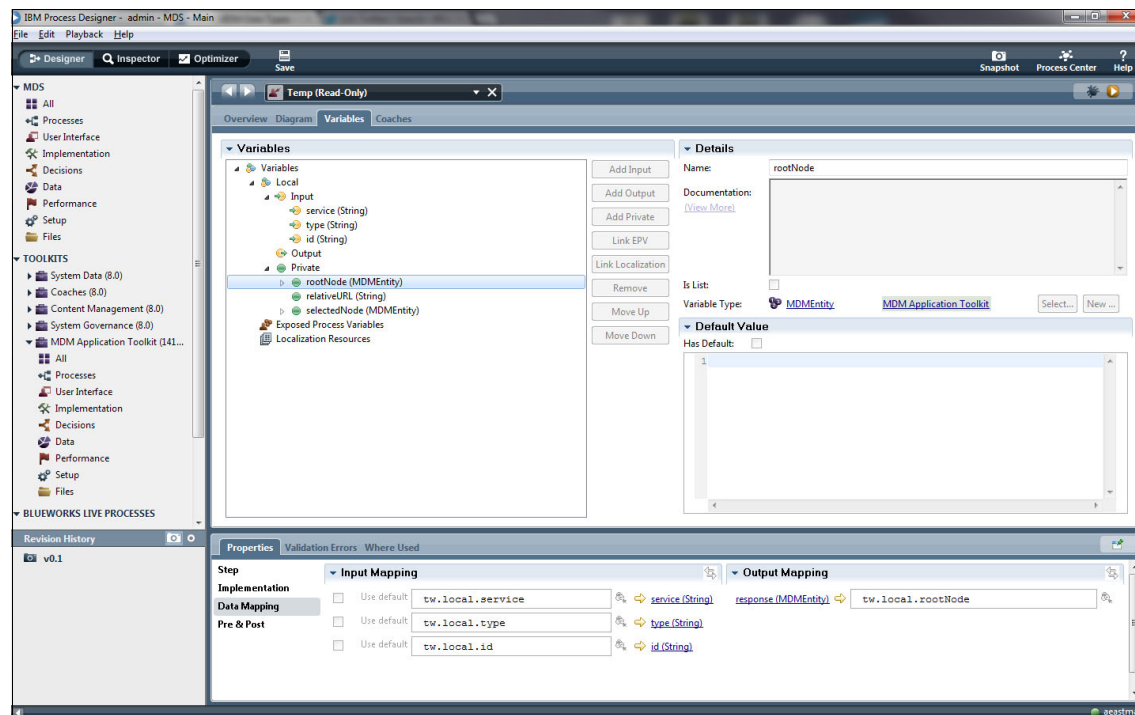


Figure 5-9 MDMEntity data type

Setting the data mappings by using the Get MDM Entity integration service

The Get MDM Entity integration service reads the initial tree data from the MDM system. To set the data mappings:

1. In the left navigation pane of Process Designer, expand **Toolkits**, and select **MDM Application Toolkit**.
2. Click **Implementations**.
3. Drag the Get MDM Entity integration service to your business process.

You connect this integration service into your business process as appropriate. For example, you might first define a search process that allows a user to search for an entity. When the user selects a search result, you might then connect that to this Get MDM Entity node to retrieve tree data for that selected entity and then visualize that data in an MDMTree control.

4. In the lower right panel of Process Designer (Figure 5-10 on page 55), click **Data Mappings**. Under **Input Mapping**, complete the following inputs for the Get MDM Entity integration service:
 - a. For Service, enter a name that matches the name of an adapter that is defined in your REST service configuration file, as explained in 5.3.4, “Configuring the REST service for BPM” on page 57. When integrated with the MDM Standard Edition engine, you might call this service MDS. When you integrate the service with the MDM Advanced Edition engine, you might call this service MDMS.
 - b. For Type, enter the type of entity that you want to retrieve. The type must match an entity type that is defined in your MDM configuration. For example, an entity type might be “person,” “customer,” or “organization.”
 - c. For ID, enter the unique ID of the entity that you want to retrieve.

Under **Output Mapping** for the Get MDM Entity integration service, set response (MDMEntity).

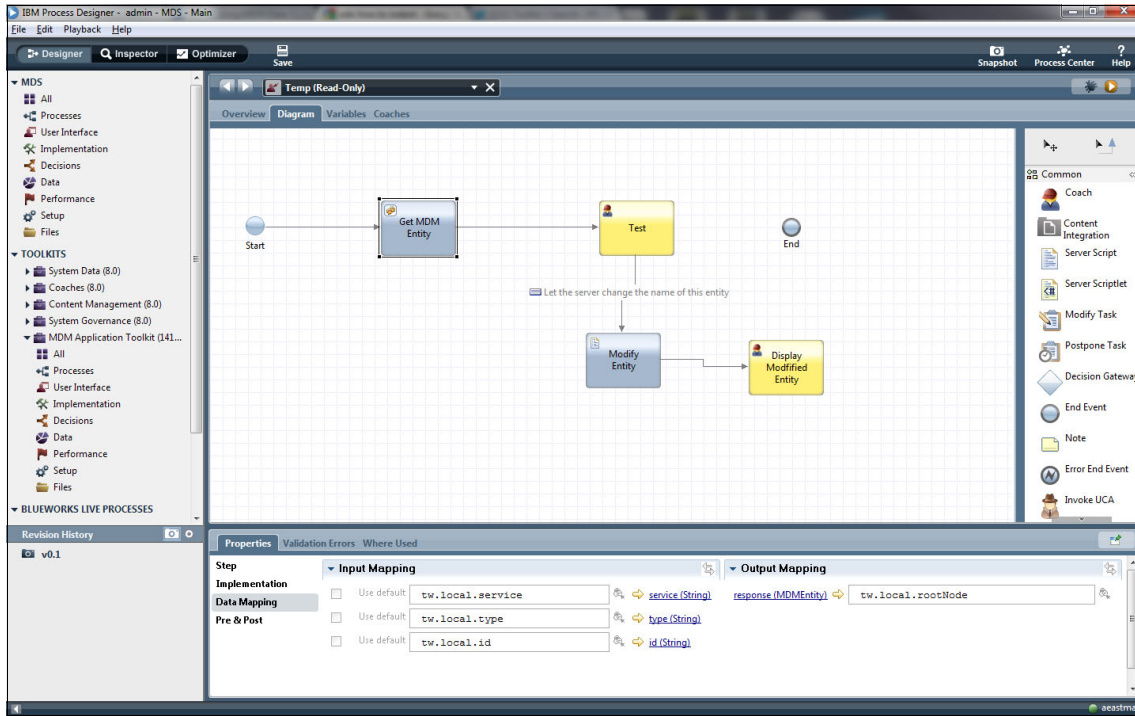


Figure 5-10 Data mappings for the Get MDM Entity integration service

Configuring the parameters for the MDMTree UI control

To configure the MDMTree UI control, you must first create a coach in your BPM process. To configure the parameters for the MDMTree UI control:

1. In the right pane of Process Designer, on the **Coaches** tab, open the MDM palette.
2. Drag the MDMTree UI control onto the coach.
3. In the Properties panel, click **General**, and bind the control to the variable of type `MDMEntity` that you created.

The MDMTree requires the configuration parameters that are listed in Table 5-3.

Table 5-3 Required MDMTree configuration parameters

Parameter	Description	Use
rootNode	A parameter of type <code>MDMEntity</code> that contains the <code>entityid</code> or <code>partyid</code> of the root entry to be displayed within the tree. Typically this object is the same object that is returned by the previous node in the process that called the Get MDM Entity integration service.	Mandatory

Parameter	Description	Use
iconClasses	A list of custom CSS classes that can be specified to display different icons in the MDM Tree.	Optional
restContext	A string parameter that contains the context root of the MDM Rest interface that is deployed to the server. The MDM Tree uses this parameter to initiate a call to the REST interface to support lazy loading.	Mandatory
service	A string parameter that you set to the name of the required adapter in the REST configuration file. For a list of default adapters that are available, see “Overview of the MDM REST service” on page 45.	Mandatory

- To define the configuration parameters, select the tree on the **Coaches** tab of the Designer page. Then, in the Properties pane, click **Configuration** (Figure 5-11).

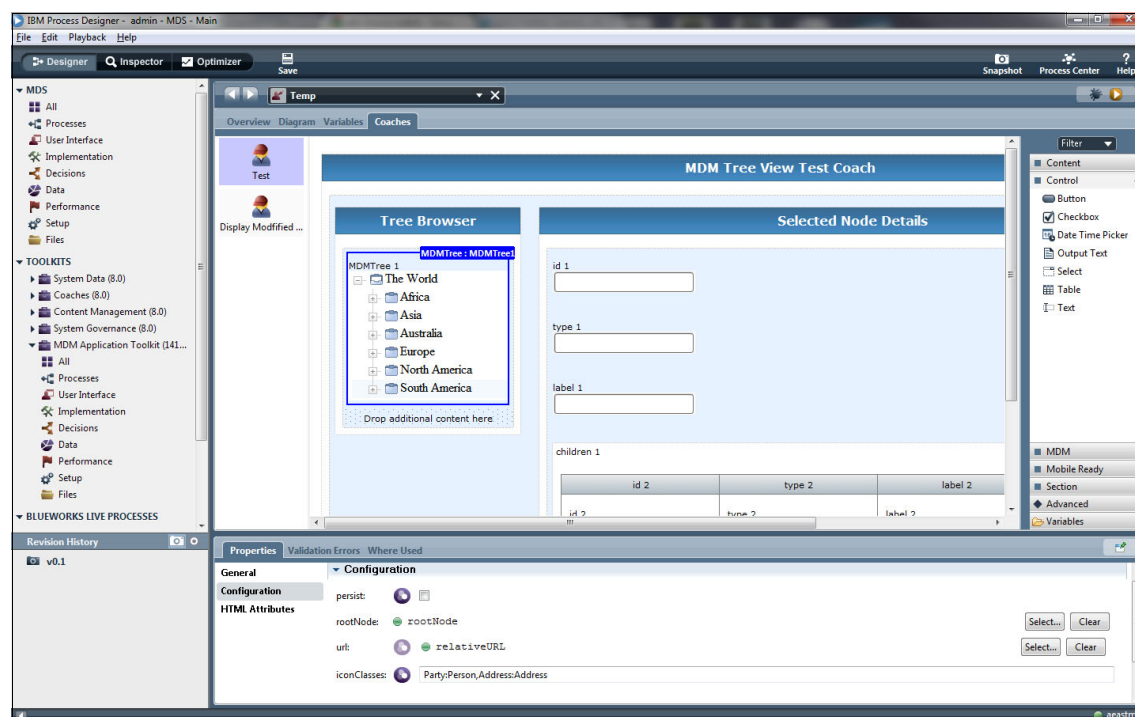


Figure 5-11 Configuration tab of the Properties pane

Saving changes to tree data

As described in 5.1.4, “Integration services” on page 44, the InfoSphere MDM Application Toolkit includes two prebuilt integration services. You can use them to persist, to the underlying MDM engine, any changes that are made to the data

that is displayed in the MDM Tree. These integration services handle the complexities that can be involved in determining which services to call to ensure the changes that are made by the users are saved to the MDM engine.

The MDM Application Toolkit includes two integration services for this purpose:

- ▶ *Save MDS Tree* to use with an MDM Standard Edition Engine Save
- ▶ *MDSM Tree* to use with an MDM Advanced Edition Engine

Both integration nodes use the same parameters, which are listed in Table 5-4.

Table 5-4 Integration nodes parameters

Service	Description	Use
tree	The MDMTree object that was bound to the MDM Tree UI control. This data type contains the change history that was built up as a result of user changes to the hierarchy data.	Required
connection	An object of type MDM_Connection. Its properties must be set to the connection credentials of the MDM Engine that you want to connect to.	Required

5.3.4 Configuring the REST service for BPM

You can configure the REST service of the InfoSphere MDM Application Toolkit to return only the data that is required by your application. For example, you can specify which relationships to return to the tree.

The REST service components include the BPMConfig.xml file in the WEB_INF/classes directory of the MDMAT_BPM_REST.war file. This configuration file defines the configuration for the REST service. The <Adapter> tag identifies the different connections for each MDM system, as shown in Example 5-1.

Example 5-1 The <Adapter> tag

```
<Adapter servicePath="MDSHierarchy"
adapterClass="com.ibm.atlantis.bpm.rest.toolkitdataadapters.MDSToolkitDataAdapter"> <server>
<host>localhost</host>
<port>16013</port>
and so forth...
```

Table 5-5 describes the parameters of the <Adapter> tag.

Table 5-5 <Adapter> tag parameters

Parameter	Description
servicePath	The name of the adapter. The name must match the name that is specified in the Get MDM Entity integration service and that you used within the URL for the REST service that you provide to the MDMTree control.
adapterClass	Specifies the Java class that implements this service.
<server>	Includes connection and authentication information for the MDM engine, which you must configure to suit your specific MDM engine connection credentials.

The <tree> tag in the configuration file determines the specific structure of the MDM objects to display in the MDM Tree UI control and the MDM objects to display in the MDM Tree UI control. See Example 5-2.

Example 5-2 The <tree> tag

```
<entities>
<entity type="person" cvw="Person EMCA">
<nodeLabel>Person: {NAME:name1}, {NAME:name2}, {NAME:name3} </nodeLabel>
<attributeCategory attribute="POB" categoryLabel="Place of
Birth">
<nodeLabel>{attrval}</nodeLabel> </attributeCategory>
<attributeCategory attribute="ADDRESS" categoryLabel="Address">
<nodeLabel>{stline1} {postalcode}</nodeLabel> </attributeCategory>
<attributeCategory attribute="EMAIL"
categoryLabel="Email Address"> <nodeLabel>{attrval}</nodeLabel> </attributeCategory>
<relationship type="employer" categoryLabel="Employees" /> <suspectedDuplicate
categoryLabel="Suspect Duplicates"/> </entity>
and so forth...
```

Table 5-6 describes the Tree tag parameters.

Table 5-6 Tree tag parameters

Parameter	Description
<nodeLabel>	Represents the attributes that are displayed when an entity is displayed in the MDM Tree.
<attributeCategory>	Represents the categories that are displayed in a tree node and to which MDM attributes they refer.
<relationship>	Represents any relationships that are required for display in the hierarchy.

Parameter	Description
<suspectedDuplicate>	Represents any suspect duplicate records that are required for display in the hierarchy.

Each entity type that is defined in your MDM configuration must have an entry. If you do not provide a configuration for a specific entity type, the REST service uses the default behavior for that entity type as defined by the MDM data model. For example, it returns any relationship types you defined that involve the entity type and use the label that is defined in the MDM configuration for each of those relationship types.

Node label: You can specify a node label for each entity type. You can specify attributes that you want to appear as nodes in the tree. You can also specify which relationship types to include and the label to use for each.

For more information about how to configure the REST service, see the Business Space Information Center at:

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/topic/com.ibm.websphere.wbpm.bspace.config.doc/doc/tcfg_bsp_rest.html

5.3.5 Extending and adding services

The MDM REST service that is used to retrieve MDM data for the MDM Tree is extensible. Java developers can extend the REST service to customize its behavior. For example, you might be required to retrieve more data from a third-party data source (other than MDM) to include in the tree. Your development team can write its own Java class and extend the classes of the InfoSphere MDM Application Toolkit that are used in this REST service. The XML configuration can be extended to define a new adapter that refers to the new Java integration service.

For more information about how to extend the REST service of the InfoSphere MDM Application Toolkit, search for *extending a rest service* in the IBM InfoSphere MDM Information Center at:

<http://pic.dhe.ibm.com/infocenter/mdm/v10r0m0/index.jsp>



Defining a master data creation process

This chapter explains how to construct a simple process-orientated master data management (MDM)-powered application by using IBM Business Process Manager Express. The business process is based on a simple customer onboarding scenario. This scenario demonstrates how to use BPM Express with Process Designer to build the user interface and integration points of an application. During construction of this business process, some of the capabilities in the InfoSphere MDM Application Toolkit are used with the capabilities in BPM Express to facilitate rapid construction of the application.

This chapter includes the following sections:

- ▶ Overview of the example business process
- ▶ Creating the business process definition
- ▶ Integrating the business process with MDM by using the MDM Tree
- ▶ Extending the process to save updates to an entity
- ▶ Data stewardship processes within enterprise processes
- ▶ Summary

6.1 Overview of the example business process

By using the application that is built in this scenario, business users can search for customer records in their MDM system, which provides a list of search results. A user can then drill down into the search results and view the customer record in a hierarchical view by using the MDM Tree from the InfoSphere MDM Application Toolkit. The user can then manipulate the relationships for the customer record. The user can save any changes that are made to the customer record back to the MDM system by using the integration services from the InfoSphere MDM Application Toolkit.

During construction of this application, you use the MDM data types that are in the InfoSphere MDM Application Toolkit to display elements of your MDM data model within your application. You discover how to configure the MDM Tree widget to display hierarchical data within an MDM-powered application. You also use several of the MDM service integration nodes to integrate your business process with the rich services provided by the MDM engine.

The application was created by using InfoSphere Master Data Management Standard Edition V10.1 and the bundled version of IBM Business Process Manager Express V8.0. In this scenario, the environment is already available with this configuration in place.

To complete the scenario, you need the following extra configuration:

- ▶ InfoSphere MDM Standard Edition v10.x instance with a sample individual domain deployed

For more information, see “Domain Templates” in the InfoSphere MDM Information Center at:

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.mdshs.wbuser.doc/topics/c_wbuser_aboutdomaintemplates.html

- ▶ MDM Application Toolkit deployed to BPM Process Designer V8

For more information, see 5.2, “Installing the InfoSphere MDM Toolkit” on page 48.

- ▶ REST interface of the InfoSphere MDM Application Toolkit deployed to the same IBM WebSphere Application Server instance as the BPM engine

See 5.2, “Installing the InfoSphere MDM Toolkit” on page 48.

For this scenario, you have full access to the configured environment, and all required MDM and BPM services are started. Also, the steps assume that you are familiar with the capabilities of the InfoSphere MDM Application Toolkit. For an overview of the capabilities of the toolkit, and for information about how to

install it, see Chapter 5, “Introduction to the InfoSphere MDM Application Toolkit” on page 37.

6.2 Creating the business process definition

In the following exercise, a simple application is built that presents a search option so that a user can enter search criteria. The criteria is then submitted to the MDM engine to run a search. The user receives a list of search results from the MDM engine. This scenario is extended later with more exercises.

6.2.1 Creating a process application

In Process Designer, you see a list of existing process applications. To create a process application:

1. From the list of processes, on the right side, click **Create New Process App** (Figure 6-1).

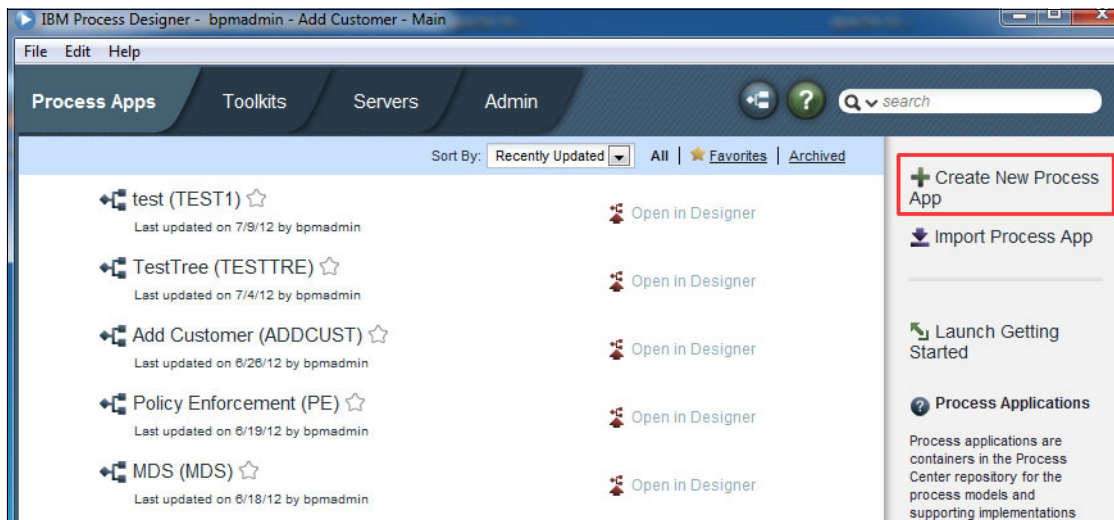


Figure 6-1 Creating a process app

2. In the Create New Process App window (Figure 6-2), enter a process app name and an acronym. The click **Create**. The new process is now added to the list of available processes.

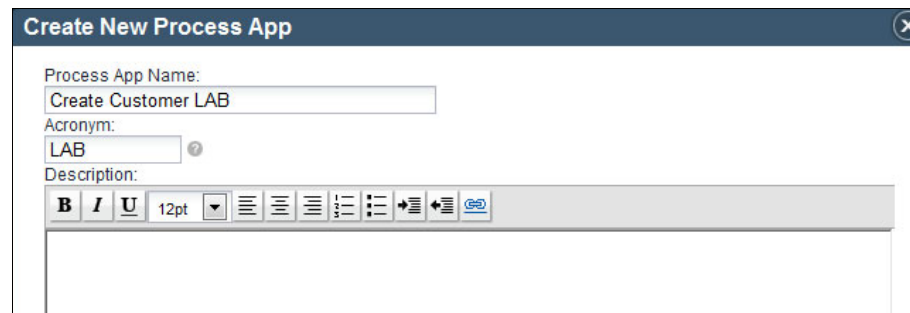
A screenshot of a web application window titled "Create New Process App". The window has a dark blue header bar with the title and a close button. Below the header, there are three input fields: "Process App Name:" with the text "Create Customer LAB", "Acronym:" with the text "LAB", and "Description:". Below the description field is a rich text editor toolbar with buttons for bold, italic, underline, font size (12pt), bulleted list, numbered list, link, and unlink. The description field itself is empty.

Figure 6-2 Enter a Process App Name and an Acronym

3. Click the **Open in Designer** link for your new process (Figure 6-3).

A screenshot of a web application showing a list of process apps. The list has four entries, each with a process icon, name, and a star icon. The first entry is "Create Customer LAB (LAB)" with a last updated date of 7/9/12 by bpmadmin. The second entry is "test (TEST1)" with a last updated date of 7/9/12 by bpmadmin. The third entry is "TestTree (TESTTRE)" with a last updated date of 7/4/12 by bpmadmin. The fourth entry is "Add Customer (ADDCUST)" with a last updated date of 6/28/12 by bpmadmin. To the right of each entry is a link labeled "Open in Designer" with a small icon. The "Open in Designer" link for the first entry is highlighted with a red rectangle.

Figure 6-3 Open in Designer

The new process opens, and the Process App Settings page is displayed.

4. Import the InfoSphere MDM Application Toolkit:
 - a. From the menu on the left side, click the + icon next to the TOOLKITS category, as shown in Figure 6-4.

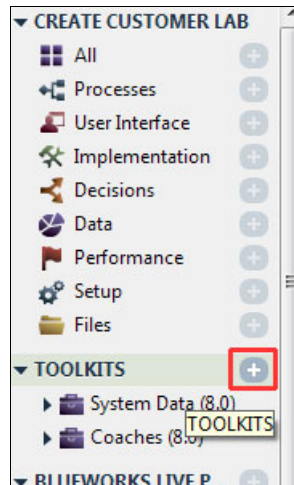


Figure 6-4 Select the plus (+) icon

- b. In the pop-up window, under the MDM Application Toolkit category, select the **10.1** toolkit (Figure 6-5). This version number might be different if you have a different version of the InfoSphere MDM Application Toolkit installed. For this exercise, assume that V10.1 is installed.

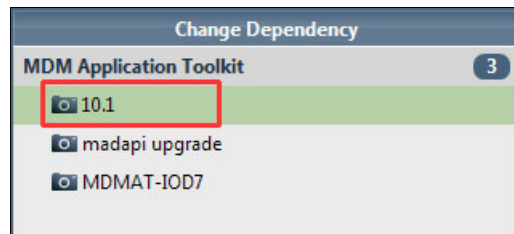


Figure 6-5 Selecting the 10.1 MDM Application Toolkit

The MDM Application Toolkit capabilities are now added to your process.

5. Under TOOLKITS on the left, expand **MDM Application Toolkit**, and click **All** (Figure 6-6). All of the capabilities are displayed that are provided by the InfoSphere MDM Application Toolkit for usage within a business process. Several of these capabilities are used throughout the exercises in this book.

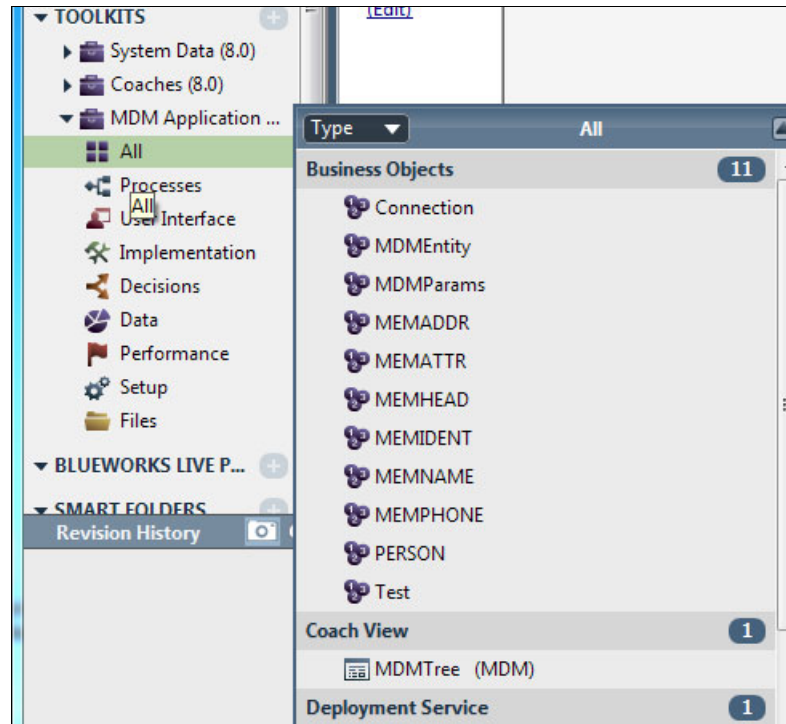


Figure 6-6 MDM Application Toolkit capabilities

6. To construct the Create Customer process:
- Under ADD CUSTOMER, click the + button next to the **Processes** label (Figure 6-7). The Processes label is for the Create Customer category and not for the Processes label in the MDM Application Toolkit category.

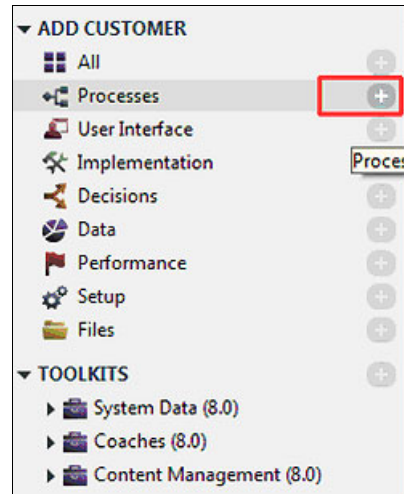


Figure 6-7 Add Customer

- In the Create New window (Figure 6-8), select **Business Process Definition**.

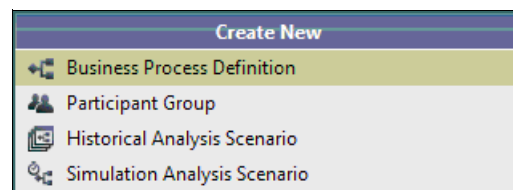


Figure 6-8 Selecting Business Process Definition

- c. In the New Business Process Definition window (Figure 6-9), enter Create Customer. Then, click **Finish**.

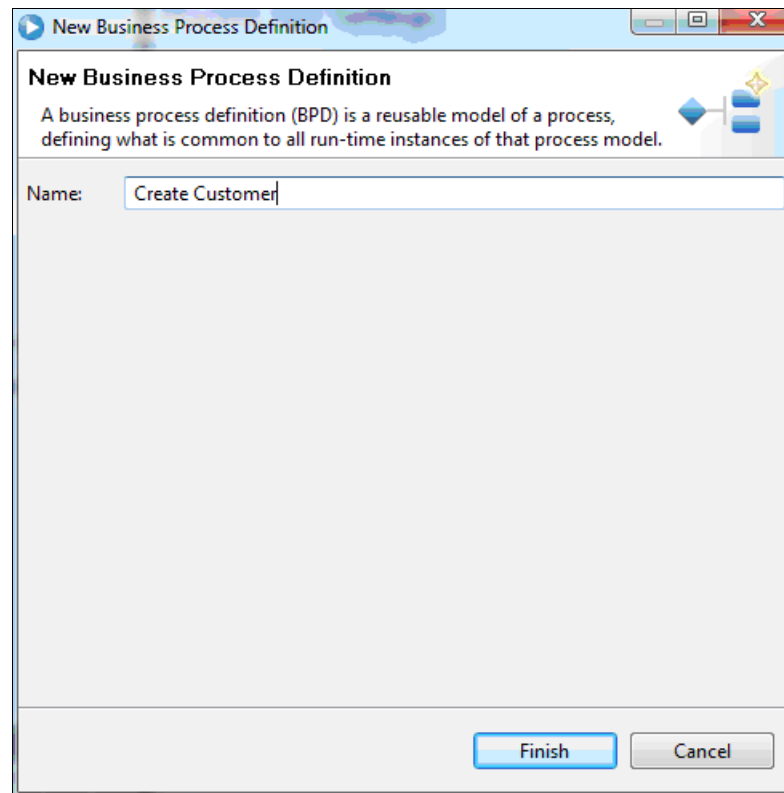


Figure 6-9 Entering Create Customer in the new business process definition

An empty process definition is displayed. Swimlanes across the panel determine who is the actor for each part of the process. This scenario uses only the Participant swimlane, because one individual is using the process.

7. In the palette on the right side, select the **Activity tool**, and drag it to the Participant swimlane between the Start and End nodes. Give it a name of Add Customer.
8. In the palette on the right side, select the **Sequence Flow Tool**. Drag relationships from the **Start** node to the **Add Customer** node and from the **Add Customer** node to the **End** node.

Figure 6-10 shows how the business process looks.

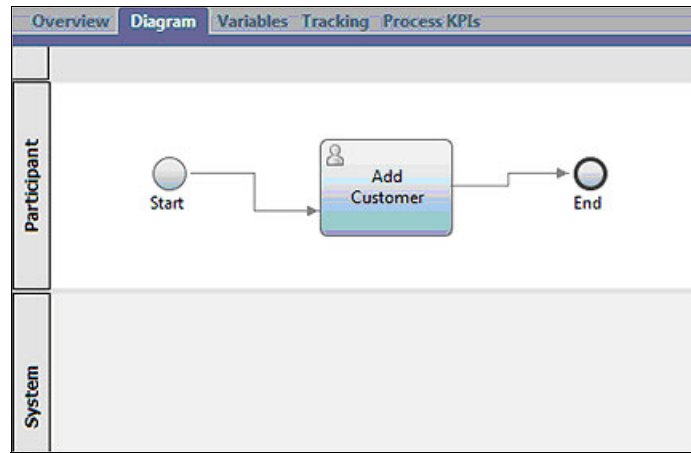


Figure 6-10 New business process

9. Press Ctrl+S to save your business process.

6.2.2 Creating a human service

Next, you drill into the Add Customer node and add the steps to the business process. The process that is being created predominantly requires human interaction. Therefore, you create a human service to define flow within the Add Customer business process.

To create a human service:

1. Under ADD CUSTOMER in the left panel, click the + button next to **User Interface** (Figure 6-11).

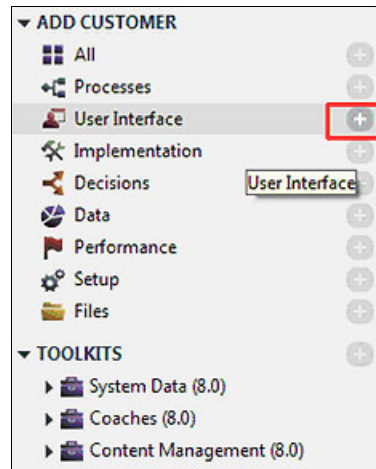


Figure 6-11 Creating a human service

2. In the Create New window (Figure 6-12), select **Human Service**.

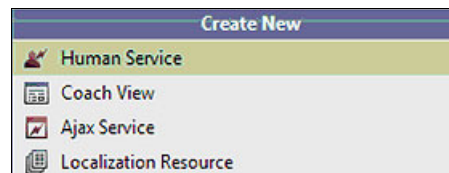


Figure 6-12 Selecting Human Service

3. In the New Human Service window (Figure 6-13), enter a name for the human service. In this example, we enter Add Customer UI Flow. Then, click **Finish**.

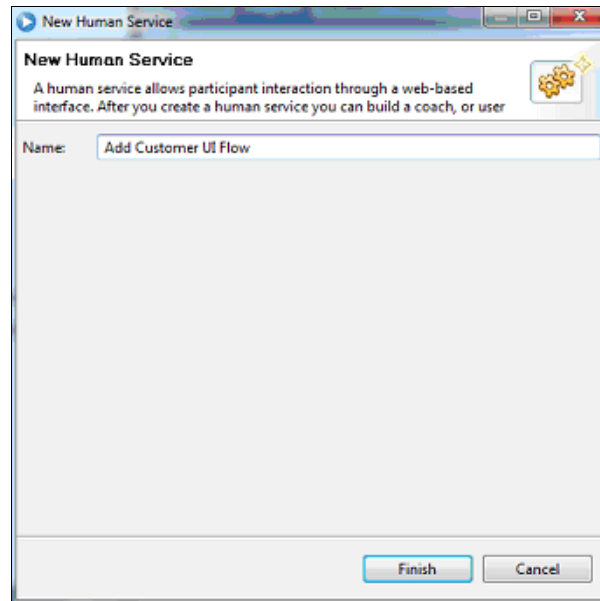


Figure 6-13 Enter a name for the Human Service of Add Customer UI Flow

A new blank canvas is created in Process Designer. This canvas is used to define the flow of the application window and the points at which the process will connect to the MDM engine to get or set data.

To build this flow, you use a coach.

4. Select the **Coach** control from the palette on the right side and drag it to the canvas. Enter a name of Search Customer, as shown in Figure 6-14.

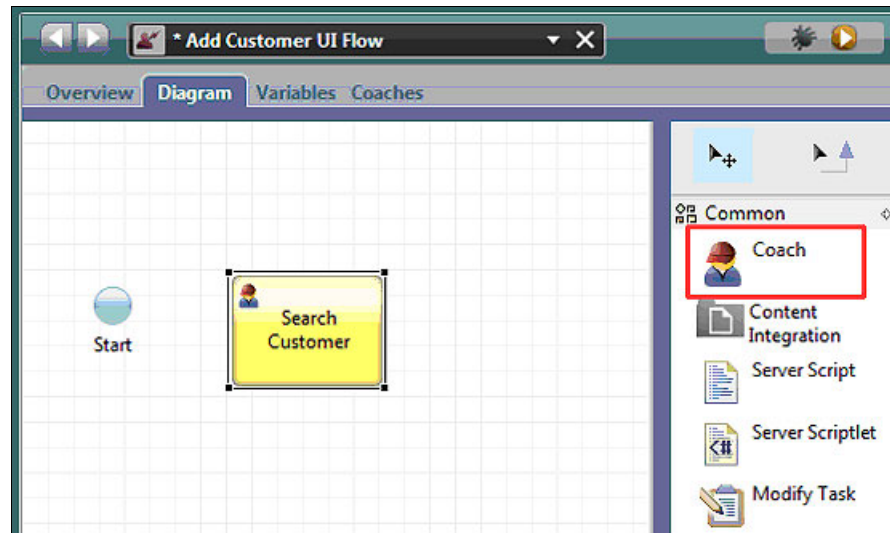


Figure 6-14 Selecting the Coach control from the palette

This node represents the UI panel that is created when searching for a customer.

5. Select the **Nested Service** tool from the palette on the right side (Figure 6-15). Drag it to the canvas, and enter a name of `Execute Search on MDM`. You might need to use the down arrow key to scroll down to the bottom of the palette window to see the **Nested Service** tool.)

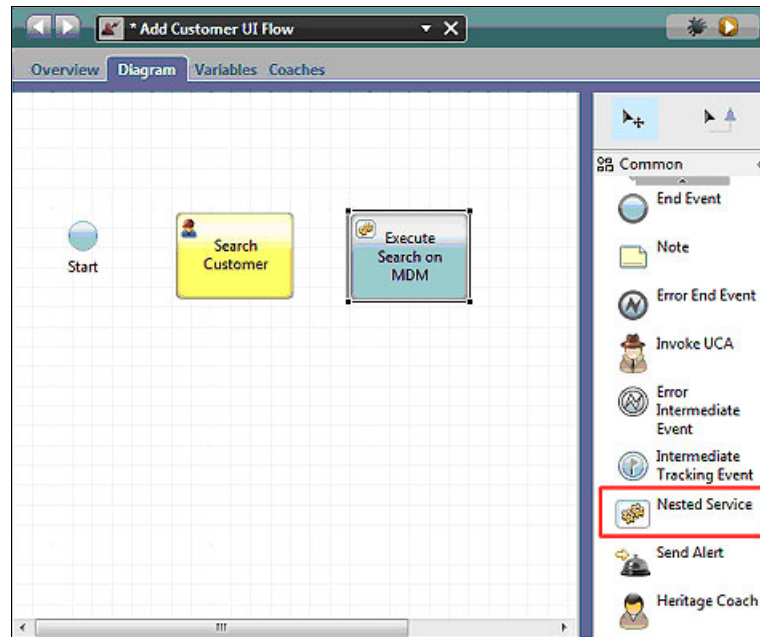


Figure 6-15 Selecting the Nested Service tool from the palette

This node represents a call to the MDM engine to run the search based on the criteria that are entered in the Search Customer UI window. After the call to search the MDM engine is completed, it returns a list of search results that need to show on another Coach UI node.

6. Select the **Coach** tool from the palette on the right side. Then, drag it to the canvas, and enter a name of `Search Results`. Figure 6-16 shows how the business process looks now.

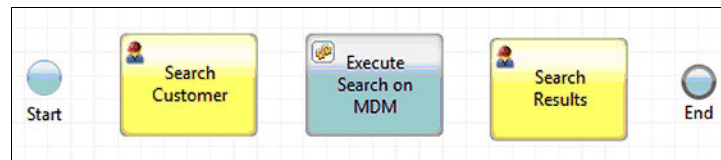


Figure 6-16 The new search results business process

6.2.3 Defining variables

Variables represent the data objects that are used to pass data between each node in the process. Each time that you want to show data on a UI window or pass data between nodes, you must ensure that you have variables that are defined in the process to accommodate this objective.

To define the variables:

1. Create a variable to represent the search criteria that will be sent to the MDM engine. On the **Variables** tab at the top of Process Designer (Figure 6-17), click **Add Private**.

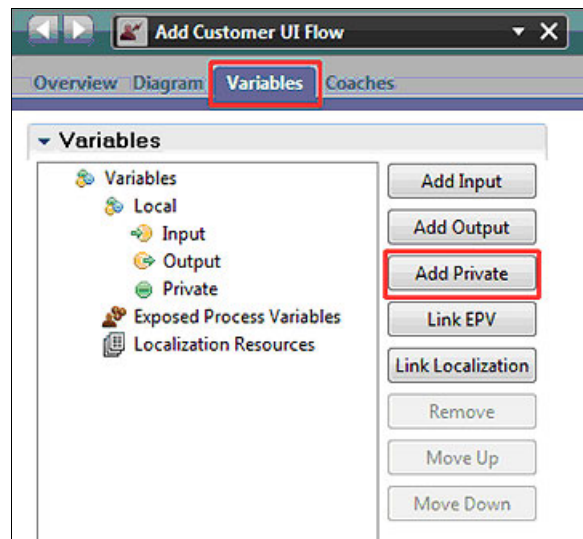


Figure 6-17 Creating a variable

2. Enter a name of CustomerSearchCriteria.

- Under Variable Type, click **Select**, and click **Individual** type (Figure 6-18). This step ensures that the new variable created is of type *Individual*, which maps to the type that is represented in the MDM engine.

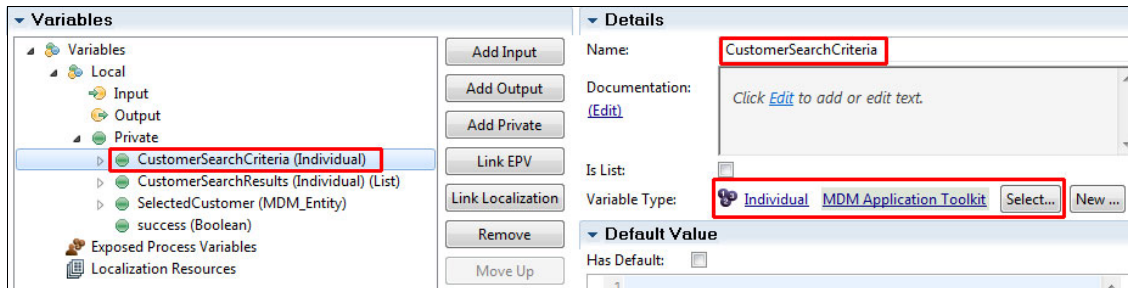


Figure 6-18 Variable type

- Create a second variable to hold the search results that are returned from the MDM engine. Click **Add Private** to create a second private variable.
- Enter a name of CustomerSearchResults.
- For Variable Type, click Select, and select the **Individual** type (Figure 6-19). Because this variable will be used to store the list of results that are returned from the MDM engine, select the **Is List** check box to ensure that BPM anticipates a list of records.

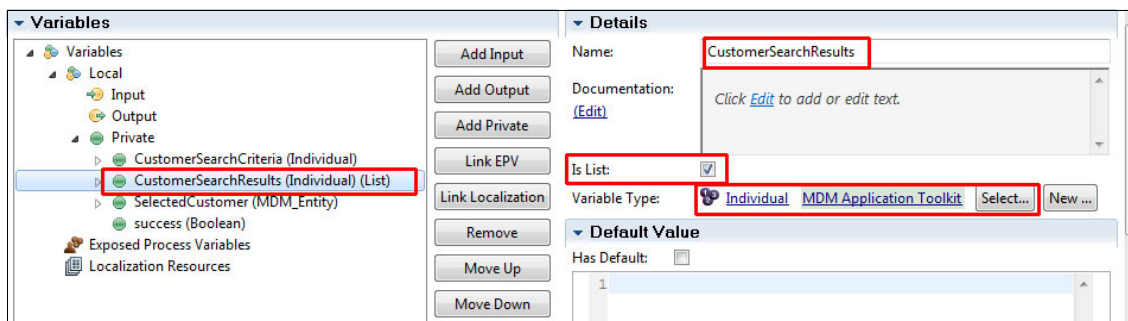


Figure 6-19 Creating a second variable

- Press Ctrl+S to save your work.

6.2.4 Building coach UI elements

Two coaches are now defined in the process. The first one allows a search to be performed, and the second one shows the list of search results that are returned

from the MDM engine. Now you populate these coaches with the required UI elements:

1. Switch back to the **Diagram** tab at the top of Process Designer.
2. Double-click the **Search Customer** coach node. The coach designer opens, and a palette of tools is displayed on the right side of the canvas.
3. Expand the **Variables** folder at the bottom of the palette. You see the two variables that you created (in the previous section).
4. Expand the **CustomerSearchCriteria** variable. You now see all of the properties that are associated with it (because it is defined as type *Individual*).
5. Expand the **LGLNAME** attribute.
6. Click the **onmfirst** variable, and then drag it to the canvas. Click the **onmlast** variable, and then drag it to the canvas. See Figure 6-20.

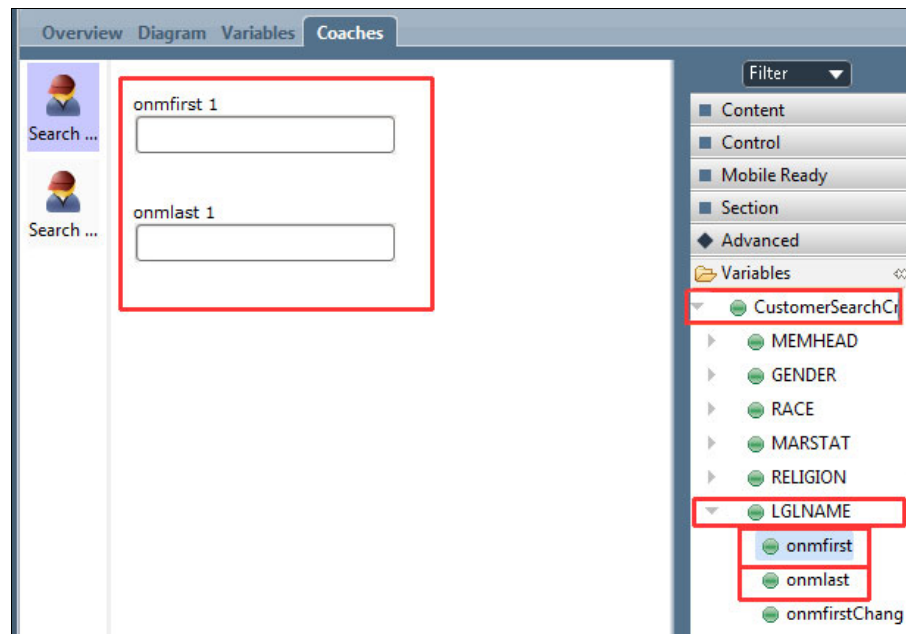


Figure 6-20 Building coach UI elements

Because the **onmfirst** and **onmlast** variables are set as type of String, a field is added by default. These fields are bound automatically to the **onmfirst** and **onmlast** variables. If data in those variables changes, the values are displayed in these fields.

7. Click the **onmfirst** field on the coach, and then click the **Properties** tab (Figure 6-21).

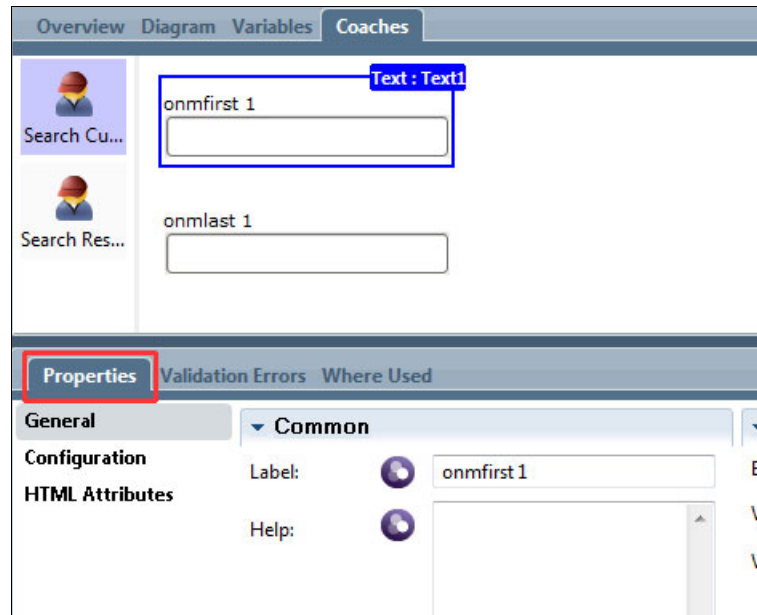


Figure 6-21 Selecting the *onmfirst* field and *Properties* tab

8. Familiarize yourself with the properties on this tab. Change the Label to a name that is more user-friendly, such as First Name.
Notice the binding property is bound to `CustomerSearchCriteria.LGLNAME.onmfirst`.
9. Click the **onmlast** field on the coach, and then in the **Properties** view, change the Label property to a value that is easier to remember, such as Last Name.
10. Expand the **Control** drawer on the palette.

11. Click the **Button** tool, and drag it to the canvas (Figure 6-22).

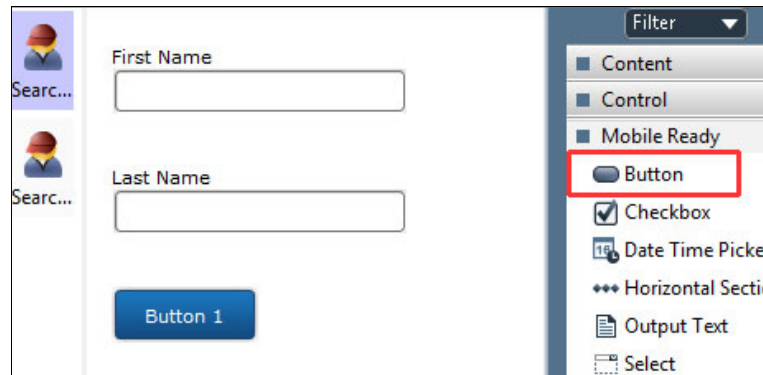


Figure 6-22 Dragging the Button tool

Button 1 becomes the button that moves the process to the next step and runs the search.

12. Click the **Button** in the coach, and in the **Properties** view, change the Label property to Search.

13. Press Ctrl+S to save your work.

The first search page is now complete, as shown in Figure 6-23.

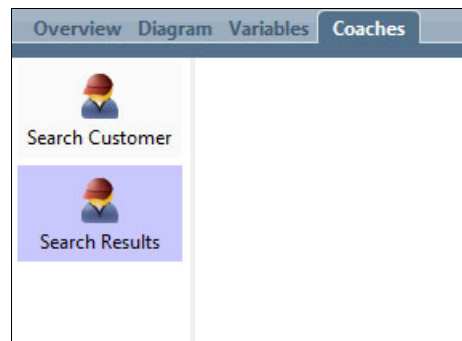


Figure 6-23 Search results

14. Construct the coach to show the search results that are returned from the MDM engine:
- Expand the **Section** drawer of the palette. Click the **Vertical Section** tool, and drag it to the canvas. Click the section on the canvas, and in the **Properties** view, change the Label to Search Result. See Figure 6-24.

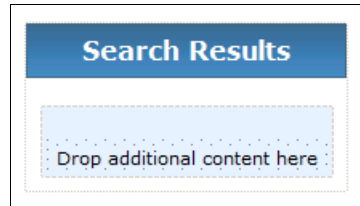


Figure 6-24 Search Results

- Expand the **Control** drawer of the palette, and select the **Table** tool. Drag it to the Search Results section, over the “Drop additional content here” text.

- c. Select the **new table** on the canvas. Then, for Binding, click **Select** (Figure 6-25). Expand **Local**, and then expand **Private**. Double-click the **CustomerSearchResults** variable.

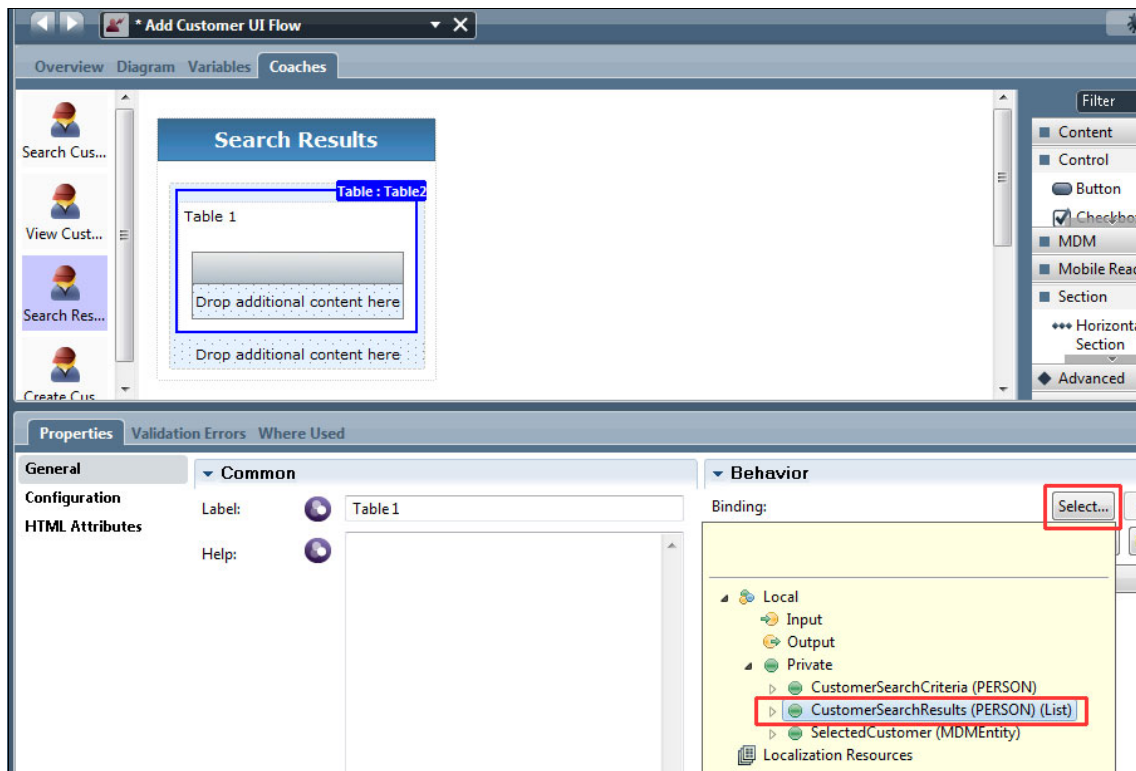


Figure 6-25 Select the new table

A table is now created that is bound to the CustomerSearchResults object. However, the table cannot detect which attributes to display in the table.

- d. Expand the **Variables** drawer on the palette, and then expand the **CustomerSearchResults** object.
- e. Expand the **LGLNAME** object.
- f. Select the **onmfirst** attribute, and drag it to the table on the canvas.
- g. Select the **onmlast** attribute, and drag it to the table on the canvas.
- h. Expand the **BIRTHDT** object.
- i. Select the **attrVal** property, and drag it to the table on the canvas.
- j. Select the **table**. In the **Properties** view, change the Label properties for the table and the fields.

Figure 6-26 shows how the coach looks now.

The screenshot shows a web-based interface titled "Search Results" in a blue header bar. Below the header is a light blue box containing the text "Search Results" in the top left corner. Inside this box is a table with three columns: "First Name", "Last Name", and "Birth Date". Each column has a corresponding text input field below it, with the same column name as a label. At the bottom of the light blue box is a dotted area with the text "Drop additional content here".

First Name	Last Name	Birth Date
<input type="text"/>	<input type="text"/>	<input type="text"/>

Drop additional content here

Figure 6-26 How the coach looks now

15. Press Ctrl+S to save your work.

6.2.5 Building the integration service

Now that you have built the UI windows, you create the integration logic behind the Execute Search on MDM node within the process. IBM Business Process Manager can integrate with InfoSphere MDM through several different connection mechanisms as described in Chapter 4, "Integration approaches and proven practices" on page 27.

This section provides an example of the Java code that can be written to code a service between BPM and MDM Standard Edition. This example is of one mechanism that you can use to run transactions against an MDM engine from BPM.

By using BPM, you can upload custom files into a business process to be used as resources by the business process. BPM provides support for Java archive files (JAR) to be uploaded that contain custom Java code that can be started within a business process. The BPM Java Integrator node can then be configured to point to this custom code so that it can be run at a particular step in the process. This Java Integrator mechanism is used in this example.

In this scenario, you create a simple searchMDM method inside a new Java class. Example 6-1 shows how the MDM Standard Edition Java API can be called from an integration service and then return objects back to the BPM in the form of a TWList to represent a list of TWObjects. Use your integrated development environment (IDE) to create the Java class and export it from your IDE packaged as a JAR file.

Example 6-1 Exporting a Java class as a JAR file

```
public class MDSAdapter {
/** * * Executes a memSearch command against an MDM Standard Edition engine, passing in search
criteria
* from a BPM process, and returns a list of search results in the form of a TWObject for
consumption
* by a BPM process.
*
* TWList search(
* @param connectionInfo * @param searchInput
* @param entType
* @param minScore
* @param maxRows
* ) * * @return searchResults
* @throws Exception
*/
public TWList search ( Object connectionObj, Object searchInputObj,
String entType, Integer minScore, Integer maxRows ) throws Exception {
TWObject connection = ( TWObject ) connectionObj; TWObject searchInput = ( TWObject )
searchInputObj;
Context context = new Context( connection.getPropertyValue( "hostname"
).toString( ),
( Integer ) connection.getPropertyValue( "port" ), connection.getPropertyValue( "username"
).toString( ), connection.getPropertyValue( "password" ).toString( ) );
DicStore dicStore = new DicStore( context ); String segAttrFilter = "";
MemRowList inputRows = new MemRowList( ); MemRowList outputRows = new MemRowList( );
//MDM API mandates that there must be a MEMHEAD property.
MemHead memHead = new MemHead( 0, 0 );
populateMemRow( memHead, ( TWObject ) searchInput.getPropertyValue( "MEMHEAD" ) );
inputRows.addRow( memHead );
TWObject MEMHEAD = (TWObject)searchInput.getPropertyValue("MEMHEAD"); String srcCode =
(String)MEMHEAD.getPropertyValue("srccode");
/*
* Iterate over all the properties in the input object.
* Each property should refer to an MDS attribute.
*/
Set< String > properties = searchInput.getPropertyNames( ); Iterator< String > i =
properties.iterator( );
while ( i.hasNext( ) ) {
String attrCode = i.next( ); System.out.println("Attribute code is " + attrCode); if (
dicStore.getSegAttrByCode( attrCode ) != null ) {
```

```

if ( segAttrFilter.length( ) > 0 ) {
segAttrFilter += ",";
}
segAttrFilter += attrCode;
TWOBJ obj = ( TWOBJ ) searchInput.getPropertyValue( attrCode );
MemRow memRow = dicStore.createMemAttrRowByCode( attrCode, memHead ); populateMemRow( memRow,
obj ); inputRows.addRow( memRow );
}
}
IxnMemSearch memSearch = new IxnMemSearch( context ); memSearch.setEntType( entType );
memSearch.setMinScore( minScore.shortValue( ) ); memSearch.setMaxRows( maxRows );
memSearch.setRecStatFilter( "A,I" ); memSearch.setSegAttrFilter( segAttrFilter );
memSearch.setSegCodeFilter( "MEMHEAD,MEMATTRALL" );
boolean success = memSearch.execute( inputRows, outputRows, GetType.ASENTITY,
SearchType.ASMEMBER );
if ( !success ) {
System.err.println( memSearch.getErrText( ) );
return null;
}
TWList resultsList = TWOBJFactory.createList( searchInput.getTWClassName( ) );
Map< ?, ? > entityMap = MemRowListUtils.getEntityMap( outputRows ); Iterator< ? > l =
entityMap.keySet( ).iterator( );
while ( l.hasNext( ) ) {
Long entrechno = ( Long ) l.next( );
TWOBJ resultObj =
TWOBJFactory.createObject( searchInput.getTWClassName( ) );
Map< ?, ? > memberMap = ( Map< ?, ? > ) entityMap.get( entrechno ); MemRowList rowList =
MemRowListUtils.getListFromMemberMap( memberMap );
RowIterator rows = rowList.rows( );
while ( rows.hasMoreRows( ) ) {
MemRow row = ( MemRow ) rows.nextRow( ); addProperty( resultObj, row );
}

}
int index = linearSearch( resultsList, resultObj ); resultsList.addArrayData( index, resultObj
);
}
return resultsList;
}
protected void addProperty( TWOBJ resultObj, MemRow memRow ) {
// Only return the first value for any attribute
if ( resultObj.getPropertyValue( getAttrCode( memRow ) ) != null ) return; TWOBJ attrObj =
null;
try {
attrObj =
TWOBJFactory.createObject( memRow.getRowSegCode( ).toUpperCase( ) ); }
catch ( Exception e ) {
e.printStackTrace( );
return;
}
}

```

```

}
SegDef segDef = memRow.getSegDef( );
for ( int i = 0; i < segDef.getFldDefCnt( ); i++ ) {
FldDef fldDef = segDef.getFldDefByNo( i );
String fieldname = fldDef.getFldName( ).toLowerCase( );try {
memRow.getDateAsDS( fldDef );
memRow.getInt( fldDef );if( fldDef.getFldType()== FldType.DATE ){
attrObj.setPropertyValue( fieldname,
}
else if( fldDef.getFldType()== FldType.INT ){
attrObj.setPropertyValue( fieldname,
memRow.getLong( fldDef ));} elseif( fldDef.getFldType( )== FldType.LONG ){
attrObj.setPropertyValue( fieldname,( int )
memRow.getShort( fldDef));} elseif( fldDef.getFldType( )== FldType.SHORT ){
attrObj.setPropertyValue( fieldname,( int )
memRow.getString( fldDef));} elseif( fldDef.getFldType( )== FldType.STRING ){
attrObj.setPropertyValue( fieldname,
}
}
}
catch ( Exception e ) {
// An exception simply means that the BPM object does not use this
field,
// so just ignore it.
}
}
resultObj.setPropertyValue( getAttrCode( memRow ), attrObj );
}
protected String getAttrCode( MemRow memRow ) { if ( memRow instanceof MemHead ) { return
"MEMHEAD";
}
else {
MemAttrRow attrRow = ( MemAttrRow ) memRow; return attrRow.getAttrCode( );
}
}
protected int linearSearch( TWList list, TWObject key ) { try {
TWObject keyHead = ( TWObject ) key.getPropertyValue( "MEMHEAD" ); int keyScore = ( Integer )
keyHead.getPropertyValue( "matchscore" );
for ( int i = 0; i < list.getArraySize( ); i++ ) {
TWObject item = ( TWObject ) list.getArrayData( i );TWObject itemHead = ( TWObject )
item.getPropertyValue( "MEMHEAD"
);
int itemScore = ( Integer ) itemHead.getPropertyValue( "matchscore"
);
if ( keyScore > itemScore )
return i;
}
}
catch ( Exception e ) {
}
}

```


Importing the integration service into your process

After you write your Java integration service and it is packaged in a JAR file, import it into your process:

1. Click the **+** button next to the **Files** category on the palette on the left.
2. In the window that opens, navigate to the JAR file that you created that contains your search service.

The JAR file and the search service are now available to be used by the process.

The Execute Search on MDM node calls the searchMDM Java service to run a search and bring back a list of search results.

3. From the palette on the left side, click the **+** button next to **Implementation**. Select **Integration Service** (Figure 6-27).

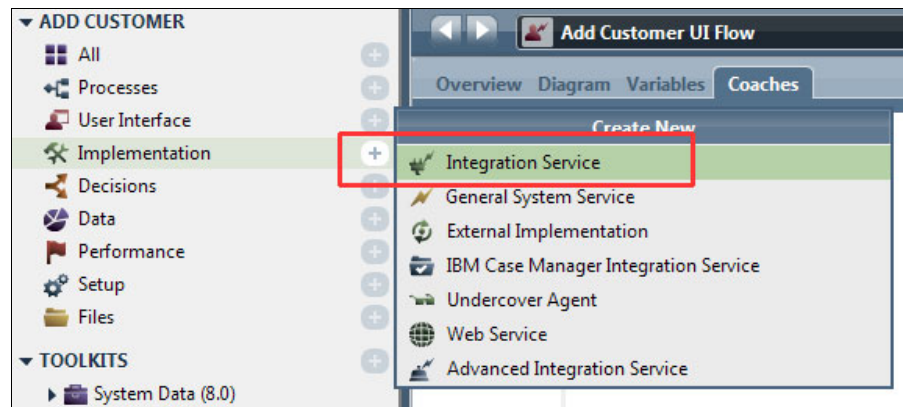


Figure 6-27 Search results

4. Enter a name of SearchMDM for your new Integration Service, and then click **Finish** (Figure 6-28).

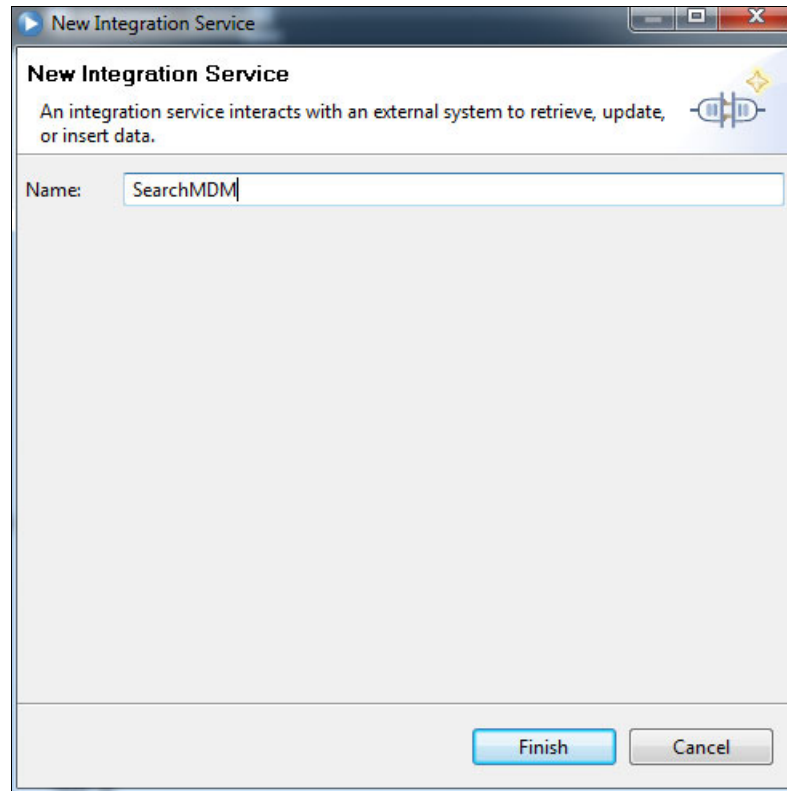


Figure 6-28 Enter a name

A new blank canvas opens so that you can define the integration components.

Defining the integration components

To define the integration components:

1. Click the **Variables** tab at the top of the editor.
2. Create a private variable to hold the details for connection into the MDM engine (Figure 6-29):
 - a. Click **Add Private**.
 - b. Enter a name of connectionDetails.
 - c. Click **Select** and select the type **MDM_Connection** from the window.
 - d. Select the **Has Default** check box.

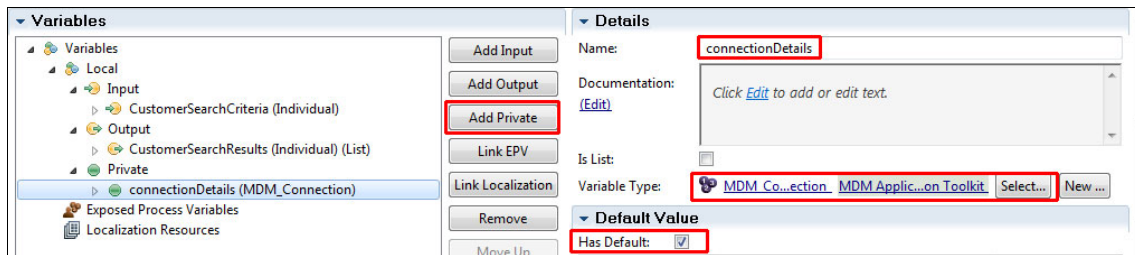


Figure 6-29 Define the integration components

The InfoSphere MDM Application Toolkit made available the MDM_Connection type that you selected.

You selected the **Has Default** check box to allow use of this editor to specify default values. In this exercise, the connection details that are required by the MDM engine are hardcoded.

3. Specify the default values for the *connectionDetails* variable you created, as shown in Figure 6-30.

hostname	The host name of your MDM Standard Edition server (can be "localhost").
port	The port on which your MDM Standard Edition server is running (default is 16000).
username	The administrator user name of your MDM engine.
password	The administrator password of your MDM engine.
serverType	The type of InfoSphere MDM server you are using (MDS for standard or MDMS for advanced).

▼ **Default Value**

Has Default: ☒

Property	Value
hostname	"localhost"
port	16000
username	"system"
password	"system"
serverType	"MDS"

Figure 6-30 Specify the default values

4. Create an input variable to store the search criteria that is passed to the integration node from the search window:
 - a. Click **Add Input**.
 - b. Enter a name of CustomerSearchCriteria.
 - c. Under Variable Type, click **Select**, and click **Individual** (Figure 6-31).

▼ **Variables**

- Variables
 - Local
 - Input
 - CustomerSearchCriteria (Individual)

▼ **Details**

Name: CustomerSearchCriteria

Documentation: Click [Edit](#) to add or edit text.

Is List: ☐

Variable Type: **Individual** MDM Application Toolkit Select... New ...

▼ **Default Value**

Has Default: ☐

Figure 6-31 Creating an input variable

5. Create an output variable to store the search results that are returned from the call to the MDM engine to allow these results to be passed into the second coach window (Figure 6-32):
 - a. Click the **Add Output** variable.
 - b. Enter a name of CustomerSearchResults.
 - c. Under Variable Type, click **Select**, and then click **Individual**.
 - d. Select the **Is List** check box.

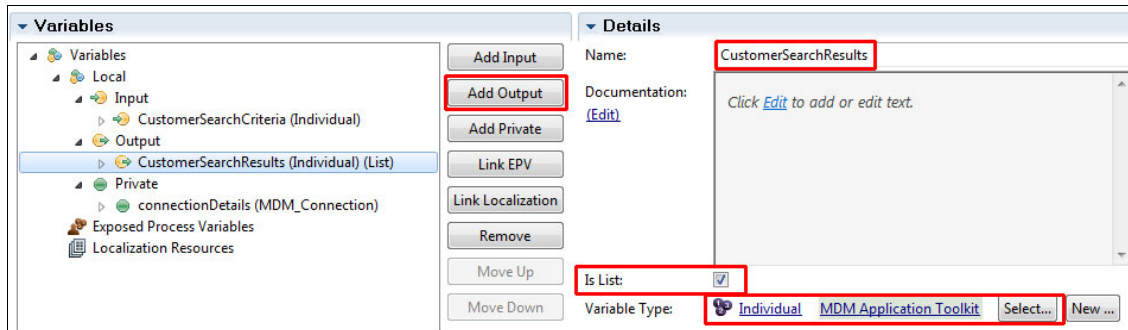


Figure 6-32 Creating an output variable

6. Press Ctrl+S to save your work.
7. Click the **Diagram** tab at the top of the page to return to the blank canvas.

8. Add the integration node that will call a service on the MDM engine and return the search results (Figure 6-33):
 - a. From the palette on the right side of the canvas, click the **Java Integration** tool, and drag it to the canvas.
 - b. Select the **new node** on the canvas.
 - c. In the Properties window, at the bottom of the canvas, enter the name `Call MDM`.

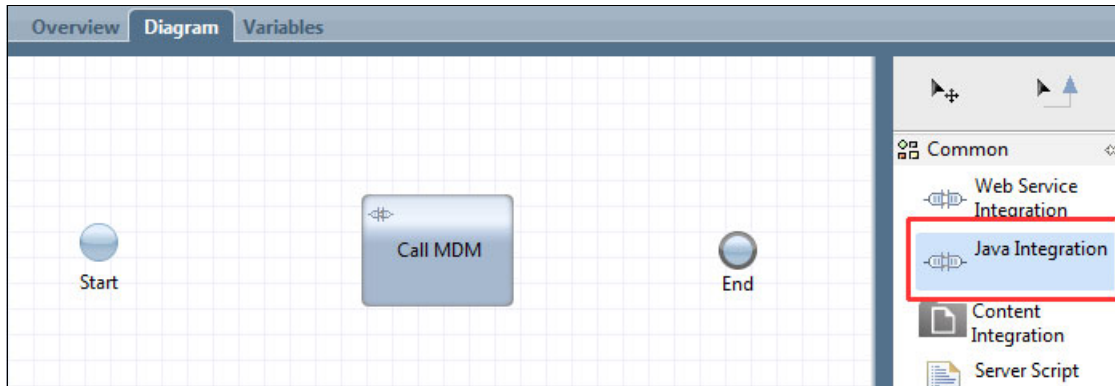


Figure 6-33 Enter a name

9. Configure the Java integration node to point to the searchMDM service on the MDM engine:
 - a. With the Call MDM node selected, click the **Definition** tab.
 - b. Click **Select** from the Java Class Property.
 - c. In the window that opens, expand the JAR file you uploaded previously, and select the class that contains your searchMDM method (Figure 6-34).

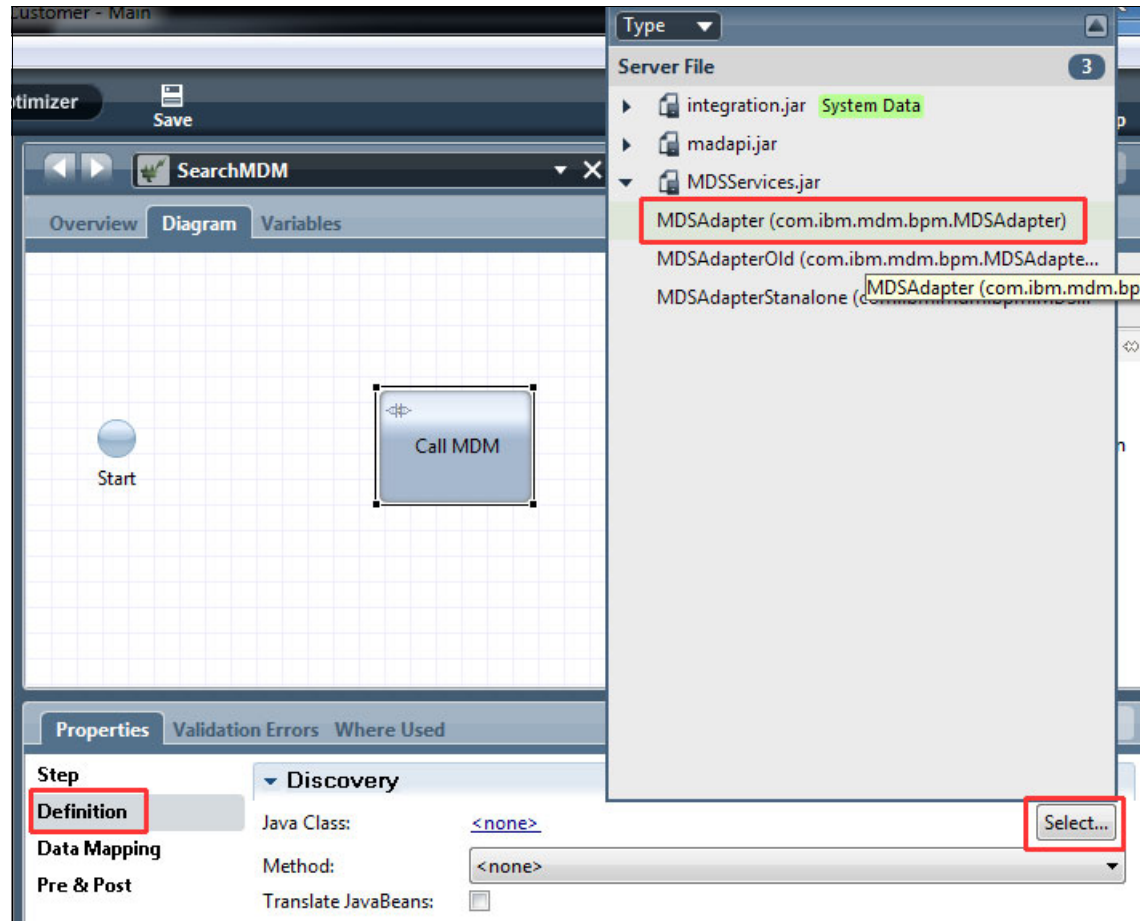


Figure 6-34 Select the class method

10. For Method, select the **TWList search(Object, Object, String, Integer, Integer)** method (Figure 6-35). Alternatively, select the name of your method if you did not use the code sample.

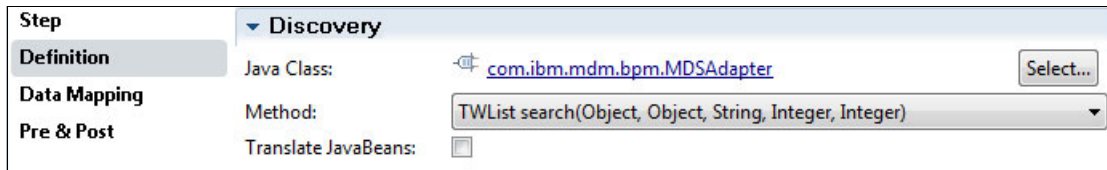


Figure 6-35 Select the TWList search

The integration node is now configured to call the search service.

Defining the variables to pass in and out of the call to MDM

Define the variables for the integration node to pass in and out of the call to the MDM engine:

1. In the **Properties** view, click the **Data Mapping** tab.

The search method that is selected expects five parameters to be passed into it. BPM provides five fields to represent these five parameters.

In this example, the parameters to be passed into the searchMDM service are the parameters that were passed into the searchMDM method from the code sample that is shown in Example 6-1 on page 82. If you write your own search service, you might have different parameters to pass in to the service.

2. Next to Parameter 1, click the **icon**.
3. In the pop-up window, double-click the **connectionDetails** variable:
! SearchService-SelectConnectionDetailsVariable.png!
This variable passes the host name and authentication credentials into the service.
4. Next to Parameter 2, click the **icon**.
5. In the pop-up window, double-click the **CustomerSearchCriteria** variable.
This variable passes the search criteria into the search service.
6. For Parameter 3, enter "id" in the field (including the quotation marks (" ")).
This value indicates to the search service the type of MDM entity to search on. This value might be different depending on your MDM hub configuration. Assume that the Individual domain supplied as a sample with MDM Standard edition is used.
7. For Parameter 4, enter 0 in the field. This value indicates to the search service the minimum match score to use in the search service.

- For Parameter 5, enter 10 in the field. This value indicates to the search service the number of results to return from the MDM engine.

Setting the output mapping

Set the output mapping. This task is required to map the search results that are returned from the service to the local variable created to store the search results.

- For the Return Value property, click the **icon**. In the pop-up window, double-click the **CustomerSearchResults** object. Figure 6-36 shows how the **Data Mapping** tab looks now.

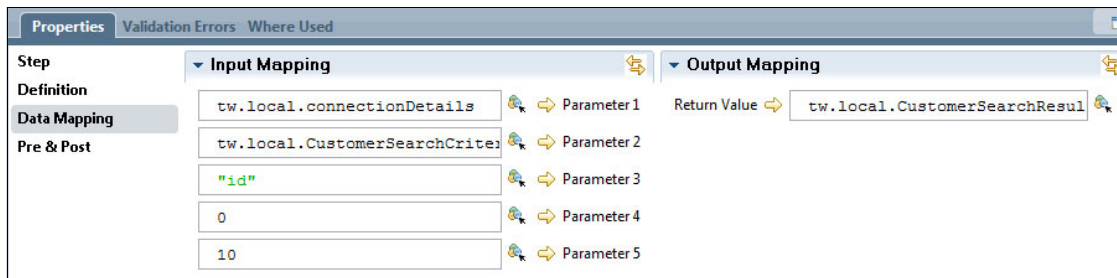


Figure 6-36 The Data Mapping tab

- To complete the integration service, connect the nodes (Figure 6-37):
 - Click the **tool** from the palette on the right side.
 - Draw a relationship from the Start node to the Call MDM node.
 - Draw another relationship from the Call MDM node to the End node.

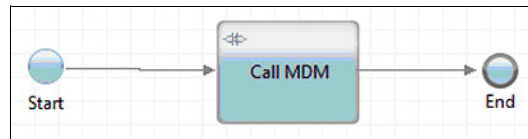


Figure 6-37 Complete the integration service

- Press Ctrl+S to save your work.

6.2.6 Connecting the UI coaches and integration service

Now that the UI coaches are built and the integration service is defined, join them together and pass the variables between each node:

1. From the palette on the left side, under ADD CUSTOMER, click **User Interface**. Then, in the window that opens, under Human Service, double-click **Add Customer UI Flow** (Figure 6-38).

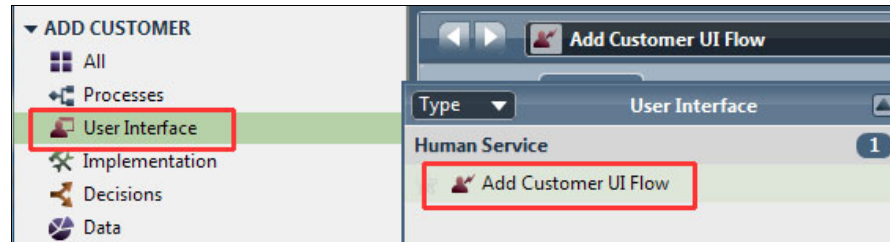


Figure 6-38 Select the Add Customer UI Flow human service

2. Configure the Execute Search on MDM node to point to the SearchMDM Service (Figure 6-39):
 - a. Switch back to the **Diagram** view.
 - b. Click **Execute Search** on MDM node from the diagram.
 - c. From the **Properties** view at the bottom of the window, click the **Implementation** tab.
 - d. Next to Attached Nested Service, click **Select**.
 - e. In the pop-up window, select the **SearchMDM** integration service.

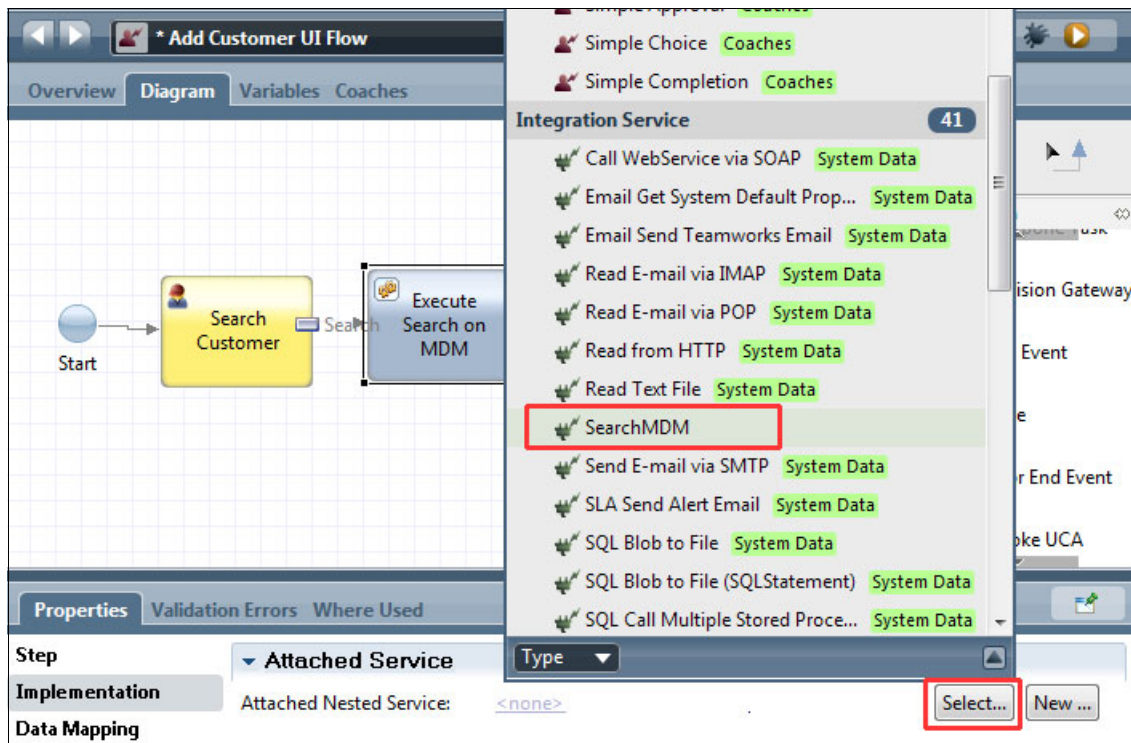


Figure 6-39 Select the integration service

The Execute Search on MDM is now pointed to the Search MDM integration service, which calls the Java class to run a search on the MDM engine.

3. Join each node of the Human Service together.
 - a. Select the **palette** on the right side.
 - b. Create the following relationships:
 - Join the Start node with the Search Customer node.
 - Join the Search Customer node with the Execute Search on MDM node.
 - Join the Execute Search on MDM node with the Search Results node.

You do not need to join the Search Results node to the End node because a button is not defined on the Search Results coach to go anywhere.

Figure 6-40 shows the Human Service now.

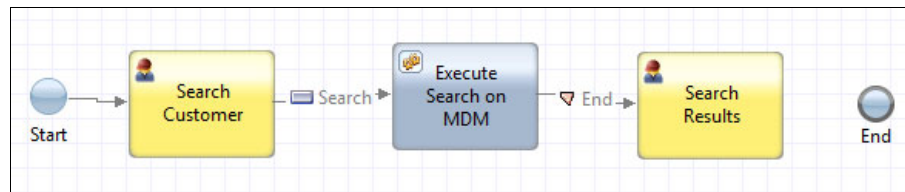


Figure 6-40 The human service

4. From the Human Service, define which of the local variables to pass in and out of the Execute Search on the MDM node:
 - a. Click the **icon** at the top of the palette to switch back to the selection mode.
 - b. Click **Execute Search** on the MDM node.
 - c. In the **Properties** view, click the **Data Mapping** tab.

On the **Data Mapping** tab, two properties are defined, one for the Input Mapping and one for the Output Mapping. These properties are displayed because in the SearchMDM integration node was defined as an Input and Output variable. Now map these properties to variables that exist in the Human Service.

- i. Click the **Input Mapping icon**.
- ii. Double-click the **CustomerSearchCriteria** variable.
- iii. Click the **Output Mapping icon**. Double-click the **CustomerSearchResults** variable.

With these mappings defined, the search criteria that is entered into the Search Customer coach is passed to the Execute Search on to MDM integration service. In addition, the search results that are returned from the integration service are passed to the Search Results coach.

5. Press Ctrl+S to save your work.

6.2.7 Running the process

The process is now complete and you can test it in the Process Designer. Process Designer provides a fast way to run and test new processes immediately, rather than requiring you to install the process to a BPM instance.

1. Click the **icon** in the upper right corner of the Process Designer. Your web browser opens, showing the Search Customer coach window.
2. In the Search Customer coach window:
 - a. In the FirstName field, enter Anna.
 - b. In the LastName field, enter Franklin.
 - c. Click **Search**.

Customer name: Anna Franklin is the name of a customer that exists in the Individual data set that provided with InfoSphere MDM Standard Edition. If you are not using the data set that comes with InfoSphere MDM Standard Edition, you must enter the name of customer that exists in your data set.

The list of search results is then returned from the MDM engine and shown in the second coach.

For this scenario, the focus is on the integration between MDM and BPM, and not on how the windows look and function. By using Process Designer, you can design and build coach windows that have all the design aspects that are required of a typical business application.

If no search results are returned from the server, an error occurred. The error might be as simple as no data was found in the MDM engine that matched the search criteria. In this case, confirm that the data exists in your hub by using the IBM Initiate Inspector application. For more information about IBM Initiate Inspector, go to:

<http://www.ibm.com/software/data/infosphere/inspector>

For a list of error codes, see “Error codes for IBM Initiate® Enterprise Viewer” in the IBM Initiate Master Data Service, Version 9.5 Information Center at:

http://pic.dhe.ibm.com/infocenter/initiate/v9r5/topic/com.ibm.vwrinstall.doc/topics/c_vwrinstall_error_codes.html

6.2.8 Checkpoint

The tasks in 6.2, “Creating the business process definition” on page 63, demonstrate how to build a simple business process that interacts with master data, by using IBM Business Process Manager and IBM InfoSphere MDM. You

built a simple search and search results window and connected them together by using variables that represent the shape of the master data. You used the Individual object that is provided by the InfoSphere MDM Application Toolkit to work with the MDM objects within BPM. You also saw how to write and package a simple search service inside a JAR file to include in your process. Next, you use the MDM Tree widget, provided by the InfoSphere MDM Application Toolkit, to provide a hierarchical view of the customer data.

6.3 Integrating the business process with MDM by using the MDM Tree

In this section, first you extend the process so that a user can select a customer record from the search results. Then, you display that record in a window by using the MDM Tree widget that is provided by the InfoSphere MDM Application Toolkit for BPM.

6.3.1 Extending the existing human service

To extend the existing Add Customer UI Flow human service with a new coach view to display the selected customer record and the MDM Tree:

1. From the **Diagram** tab in the Add Customer UI Flow human service, click the **Coach** tool from the palette on the right side of the window, and drag it to the canvas.
2. Enter a name of View Customer Details, as shown in Figure 6-41.

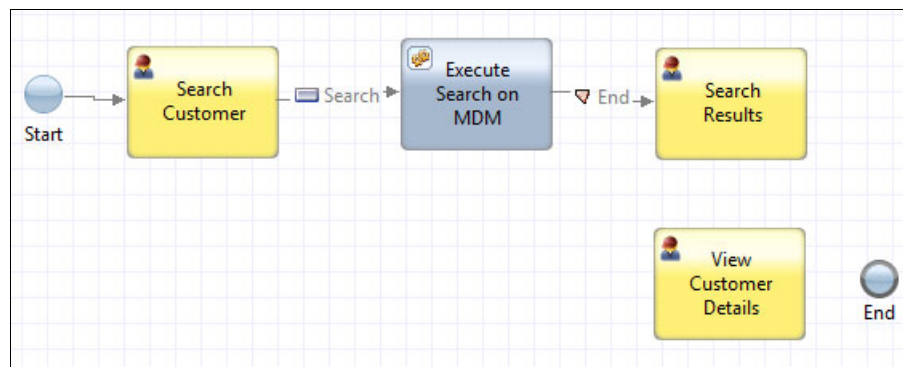


Figure 6-41 Add Customer UI Flow human service

3. Use a component from the Application toolkit to display the selected customer record in the hierarchy.

4. Click the **Nested Service** tool from the palette on the right side, and drag it to the canvas.
5. Enter a name of Build MDM Entity. You might need to scroll down in the palette window to see the Nested Service tool.
6. On the **Implementation** tab from the palette view, for Attached Nested Service, click **Select**. Select the **Get MDM Entity Integration** service (Figure 6-42).

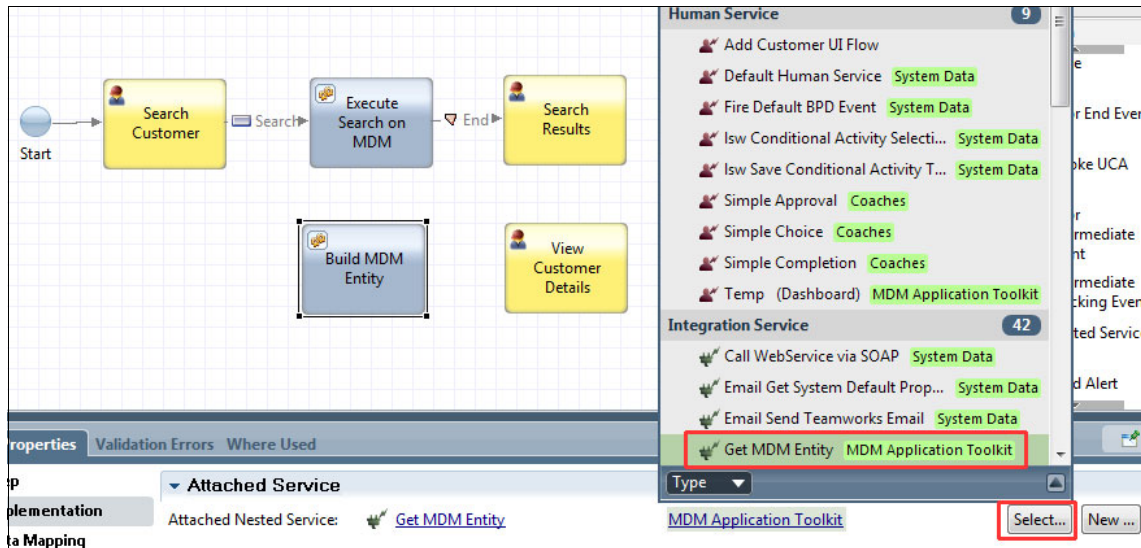


Figure 6-42 Use a component from the Application Toolkit

6.3.2 Configuring the nested service

By using the Get MDM Entity service, you can pass in the selected Customer record. You can make an extra call into the MDM engine to retrieve all of the relationship data for that customer and return it in a format that can be used by a hierarchy widget. This capability demonstrates the power that the InfoSphere MDM Application Toolkit brings to application developers. They do not need to make extra calls to MDM to retrieve the correct relationship data nor manually convert the data into the format for the hierarchy. The Get MDM Entity service that is provided by the InfoSphere MDM Application Toolkit does these tasks already.

Now you must configure the service. For this exercise, the properties that are required by the Build MDM Entity are hardcoded. Typically, the variables are defined to pass in the correct properties.

Create a variable to store the data that is returned from the Get MDM Entity service in the format that is required by the hierarchy widget:

1. On the **Variables** tab, click **Add Private**.
2. Enter a name of SelectedCustomer.
3. Click **Select**, and then set the Variable Type to **MDM Entity** (Figure 6-43).

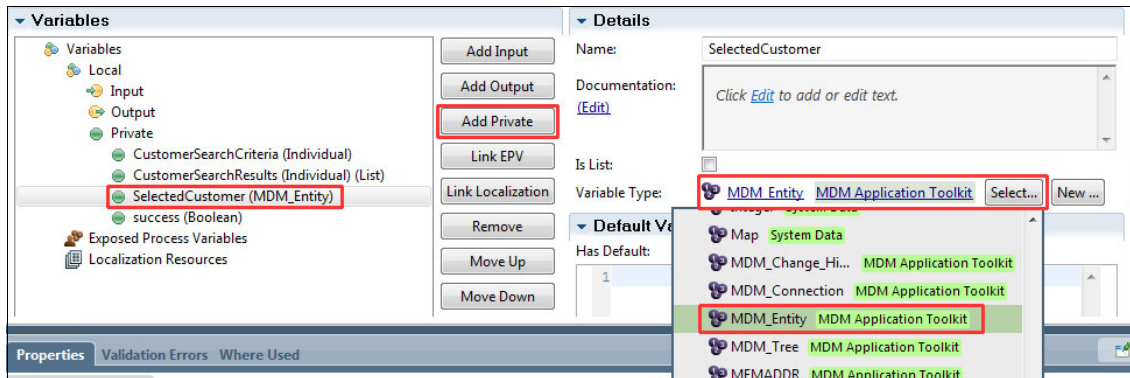


Figure 6-43 Configuring the nested service

4. Switch back to the Diagram view.
5. Click the **Build MDM Entity** node.
6. In the **Properties** view, on the **Data Mapping** tab:
 - a. For service, enter "MDSService" in the field (including the " ").

This value indicates to the Get MDM Entity service which service on the embedded REST interface to call. By default, the REST interface comes preconfigured with sample services, one for MDM Standard Edition and one for MDM Advanced Edition. In this exercise, the default service for InfoSphere MDM Standard Edition is used.

- b. For type, enter "id" in the field (including the " ").

This value indicates to the Get MDM Entity service the entity type that you want returned from the MDM engine. The id entity relates to the default entity type that is specified by the Individual domain that comes with InfoSphere MDM Standard Edition. If you are using a custom domain, specify your chosen entity type here.

7. Next to the id attribute, the **icon**.

8. In the pop-up window, expand **Private** → **CustomerSearchResults** → **listSelected** → **MEMHEAD** → **entrecno** (Figure 6-44).

Important: Navigate to the **listSelected** level and not to the top level because MEMHEAD is displayed on two levels.

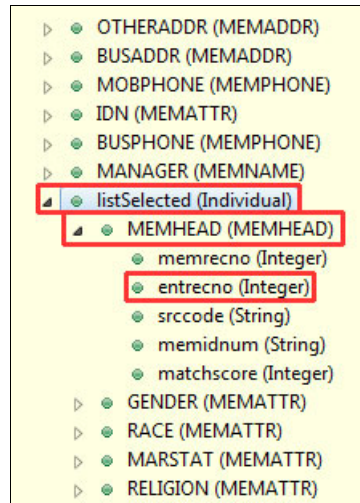


Figure 6-44 Navigating from private to entrecno

9. In the id field, add "+" to the beginning of the ID to convert the entrecno Integer into a String as required by the Get MDM Entity service. The edited the field value now looks like the following example:

```
""+tw.local.CustomerSearchResults.listSelected.MEMHEAD.entrecno
```

This property contains the MDM entity id of the customer record that was selected by the user in the previous step.

10. For port, enter "9080" in the field (including the " ").

This value indicates to the Get MDM Entity service the port number that the embedded REST interface is running on. The value 9080 is the default port. You might need to change this port to suit your BPM configuration.

11. For contextRoot, enter "MDMAT_BPM_REST" in the field (including the " ").

This value indicates to the Get MDM Entity service the location (context root) of the embedded REST interface. This value is the default value that is provided by the REST interface. If you changed the default context root, specify the new value here.

12. On the Output Mapping, click the icon for **response property**.

13. In the pop-up window, click the **SelectedCustomer** variable (Figure 6-45).

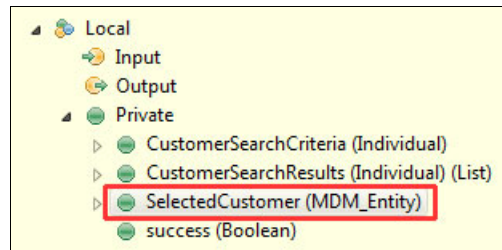


Figure 6-45 The SelectedCustomer variable

Figure 6-46 shows how the data mappings look now.



Figure 6-46 The Data Mappings

14. Press Ctrl+S to save your work.

6.3.3 Configuring the MDM Tree

Next, you build the new View Customer Details coach by adding the MDM Tree widget, provided by the InfoSphere MDM Application Toolkit for BPM, to display the customer record:

1. Open the View Customer Details coach designer.
2. From the palette on the right side, expand the **MDM** drawer. Select the **MDM Tree** tool and drag it to the canvas.
3. With the **Tree** selected, in the **Properties** view, click the **Configuration** tab. For the rootNode property, click **Select**.

4. From the pop-up window, double-click **SelectedCustomer** (Figure 6-47).

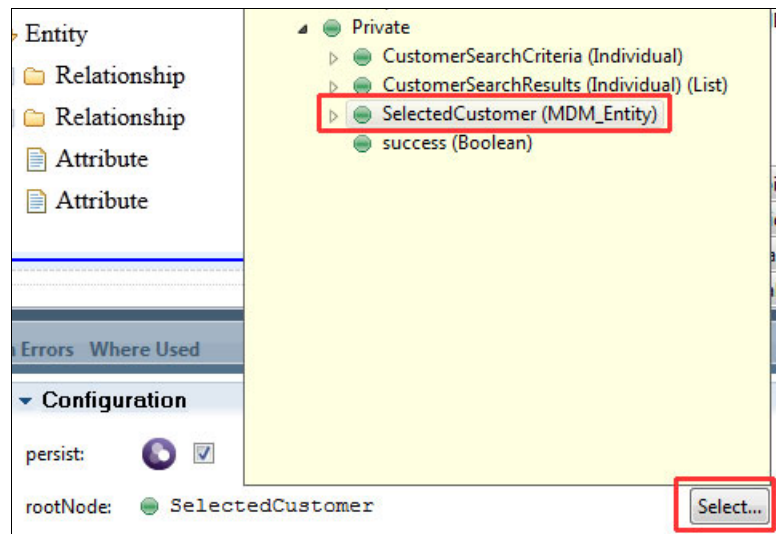


Figure 6-47 Configure the MDM Tree

Double-clicking the **Selected Customer** option signals the MDM Tree to set the root entry of the tree to the MDM Entity that is stored in the SelectedCustomer variable. The SelectedCustomer variable is the Customer record that was selected by the user in the previous window.

5. Set the URL property for the MDM Tree to MDMAT_BPM_REST. This property indicates to the MDM Tree the location of the underlying REST interface when lazy loading data when a user expands a node on the tree.

You can specify several other properties on the tree, but they are beyond the scope of this scenario. The other properties can be used to configure such capabilities as lazy loading of the tree.

6.3.4 Connecting the nodes with the service

Before you run and test the process, join all of the nodes within the Human Service:

1. Create a button on the Search Results coach so that the process can move on to the new View Customer Details coach inside the existing Search Results section.
2. Double-click the **Search Results** coach.
3. Select the **Button** tool from the palette and drag it to the canvas.

4. In the **Properties** view, on the **General** tab, rename the button to View Selected Customer (Figure 6-48).

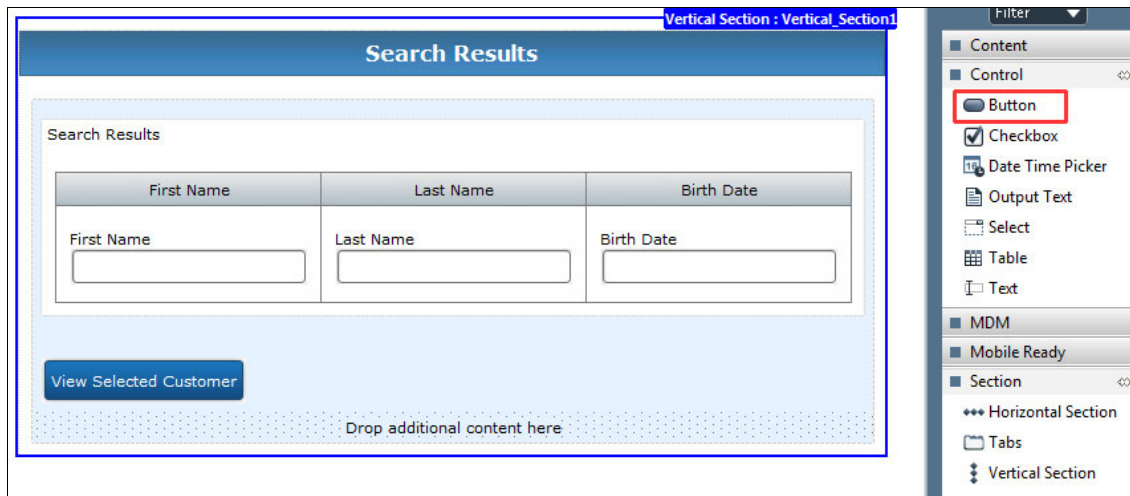


Figure 6-48 The View Selected Customer button

5. Switch back to the Diagram view.
6. Select the **tool** from the palette on the right side.
7. Create the following relationships:
 - Join the Search Results node with the Build MDM Entity node.
 - Join the Build MDM Entity node with the View Customer Details node.

Figure 6-49 shows the completed Human Service.

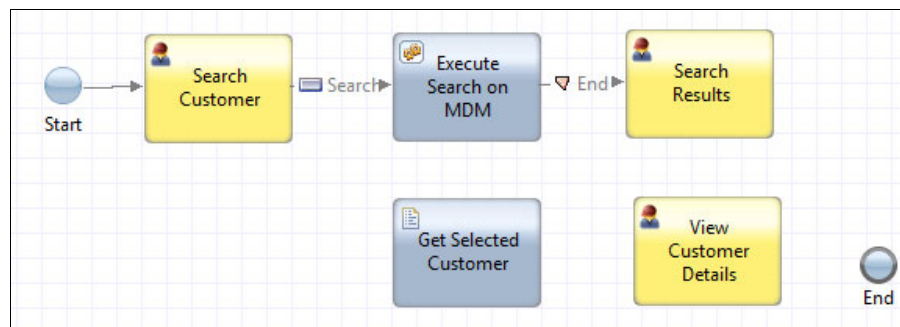


Figure 6-49 The completed Human Service

8. Press Ctrl+S to save your work.

9. Copy the `mdmbpm.zip` file from the InfoSphere MDM Application Toolkit into the local process. This compressed file contains the MDM Tree JavaScript code. Because of a defect, cannot be in the toolkit. Therefore, you must copy this file locally to your process application.
 - a. From the palette on the left, under the TOOLKITS category, expand **MDM Application Toolkit** and select the **Files** category.
 - b. In the pop-up window, right-click the **mdmbpm.zip** file.
 - c. In the pop-up window, click **Copy Item to** → **Other Process App**.
 - d. In the pop-up window, select your **process app** from the list of process applications.

6.3.5 Running your process

Now that the process is complete, you can test it in Process Designer:

1. Click the **icon** in the far upper-right corner of the Process Designer. Your web browser opens.
2. In the Search Customer coach window:
 - a. In the FirstName field, enter Anna.
 - b. In the LastName field, enter Franklin.
 - c. Click **Search**.

Customer name: Anna Franklin is the name of a customer that exists in the Individual data set that provided with InfoSphere MDM Standard Edition. If you are not using the data set that comes with InfoSphere MDM Standard Edition, you must enter the name of customer that exists in your data set.

The list of search results is then returned from the MDM engine and displayed in the second coach.

3. In the results table, select a row and click **View Selected Customer**. The MDM Tree loads, showing the attributes of the selected customer record.
4. Expand the nodes of the hierarchy.
5. Click the Options menu in the upper right corner of the window to show or hide the categories in the hierarchy.

6.3.6 Checkpoint

In this section, you extended the search application to use the MDM Tree that is provided by the InfoSphere MDM Application Toolkit to display a hierarchical view

of the customer data. You used the Get MDM Entity service that is provided by the InfoSphere MDM Application Toolkit to populate the MDM Tree and convert the selected record into a hierarchical format that is required to display that customer in a tree. By using the data types that are provided by the InfoSphere MDM Application Toolkit, you can work with the MDM data within the BPM environment in a simple and intuitive manner.

The last section of scenario expands the process further to capture the changes that a user might make to the entity that displayed in the tree and to persist those changes back to the MDM engine. This process takes advantage of another prebuilt integration node that is provided by the InfoSphere MDM Application Toolkit.

6.4 Extending the process to save updates to an entity

In this section, you further extend the process to persist any changes that are made to the data within the MDM Tree back to the MDM engine. The InfoSphere MDM Application Toolkit includes two prebuilt integration services so that you can save changes that are made to an MDM Tree back to the MDM engine. These prebuilt integration nodes handle the complexities of determining which services on the MDM engines to call to reflect the changes that are made to the MDM hierarchy.

The steps in this section use the Save MDS Tree integration service because the MDM Standard Edition is used. If you use MDM Advanced Edition, you would use the Save MDMS Tree integration service.

6.4.1 Extending the process

To extend the process:

1. Open the Add Customer UI Flow Human service, and click the **Diagram** tab. Extend this flow with a nested service that will point to the Save MDS Tree integration service.
2. From the palette on the right side, click the **Nested Service** tool and drag it to the canvas. Enter a name of Save Tree.
3. With your new node selected, from the Properties view, click the **Implementation** tab (Figure 6-50 on page 108).
4. For Attached Nested Service, click **Select**. In the pop-up window, select the **Save MDS Tree** integration service that is provided by the MDM Application Toolkit.

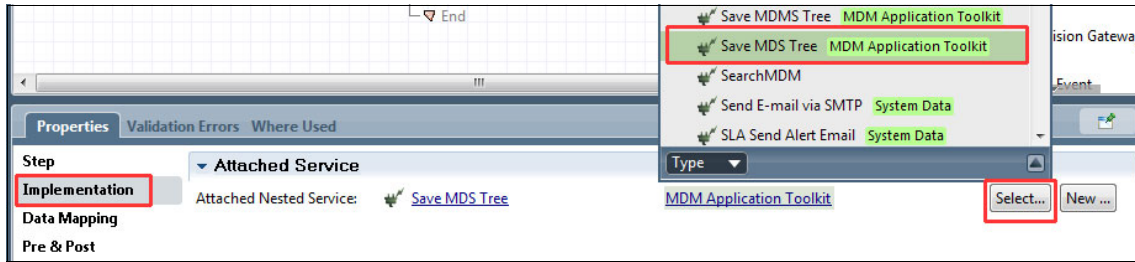


Figure 6-50 The MDS Tree integration service that is provided by the MDM Application Toolkit

6.4.2 Creating variables to hold the properties

Now, you create variables to hold the properties that are needed to pass into the new integration service. First, create a variable of type `MDM_Tree`. This type stores all the changes that are made by the user to the data that is stored in the MDM Tree on the user interface and is passed into the Save Tree service. The Save Tree service then uses the stored data to determine which services to call on the MDM engine to persist the changes that are made by the user.

To create the variables:

1. Switch to the **Variables** tab.
2. Click **Add Private** to create a private variable, and enter a name of Tree Data.
3. For Variable type, click **Select**, and select **MDM_Tree** provided by the MDM Application Toolkit (Figure 6-51).

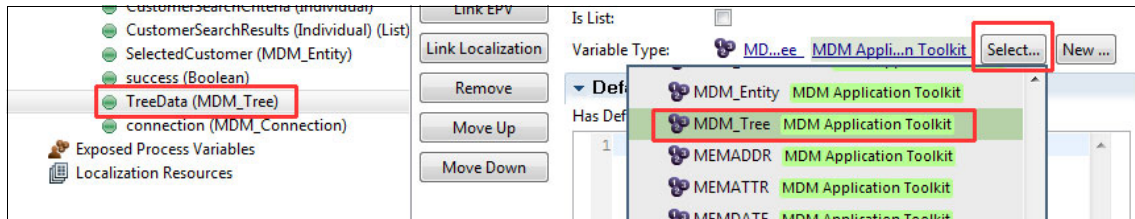


Figure 6-51 Creating a variable of type `MDM_tree`

6.4.3 Creating a connection variable

Next, create a connection variable. The connection variable contains the connection details of the MDM engine to be called to persist the changes that are made by the user.

To create a connection variable:

1. Click **Add Private**.
2. Enter a name of connection.
3. For Variable Type, click **Select**, and select **MDM_Connection** (Figure 6-52).

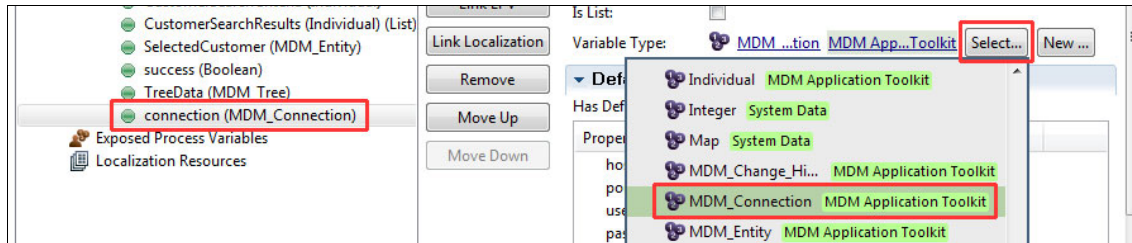


Figure 6-52 Creating a connection variable

4. Select the **Has Default** check box, and enter the following values (Figure 6-53):

hostname	The host name of your MDM Standard Edition server (can be "localhost").
port	The port on which your MDM Standard Edition server is running. The default is 16000.
username	The administrator user name of the MDM engine.
password	The administrator password of the MDM engine.
serverType	The type of InfoSphere MDM server that is being used (MDS for standard or MDMS for advanced).

Default Value	
Has Default:	<input checked="" type="checkbox"/>
Property	Value
hostname	"localhost"
port	16000
username	"system"
password	"system"
serverType	"MDS"

Figure 6-53 Defining the default variables

5. Pass the new variables that you defined to the Save Tree node on the process diagram:
 - a. Switch back to the **Diagram** view.
 - b. Select the **Save Tree** node.
 - c. In the **Properties** view (Figure 6-54), on the **Data Mapping** tab, click the **icon** next to the tree variable.
 - d. In the pop-up window, select the **TreeData** variable that you created.

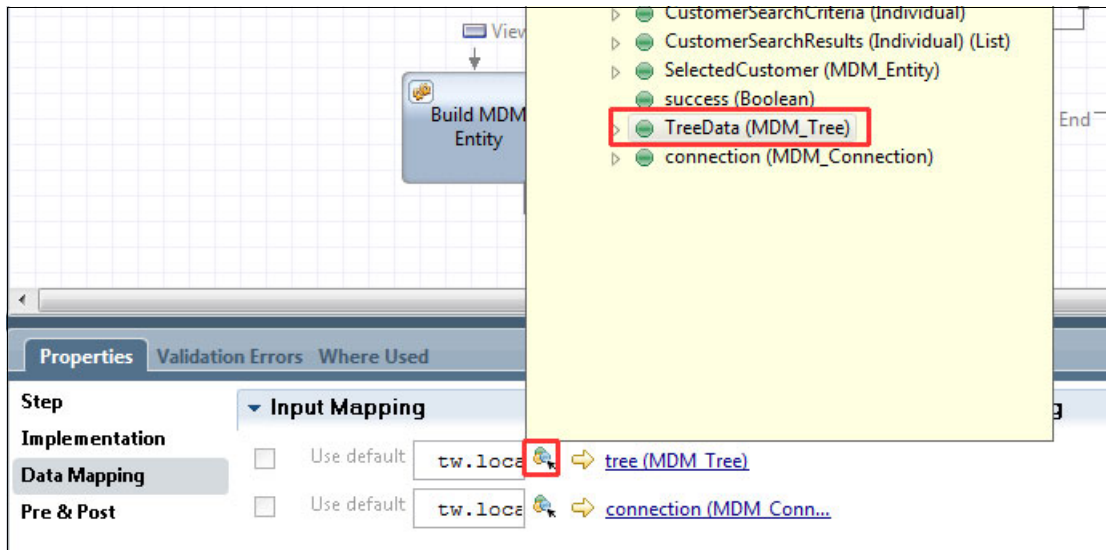


Figure 6-54 The TreeData variable

- e. Click the **icon** next to the connection variable.

- f. In the pop-up window, select the **connection variable** that you created (Figure 6-55).

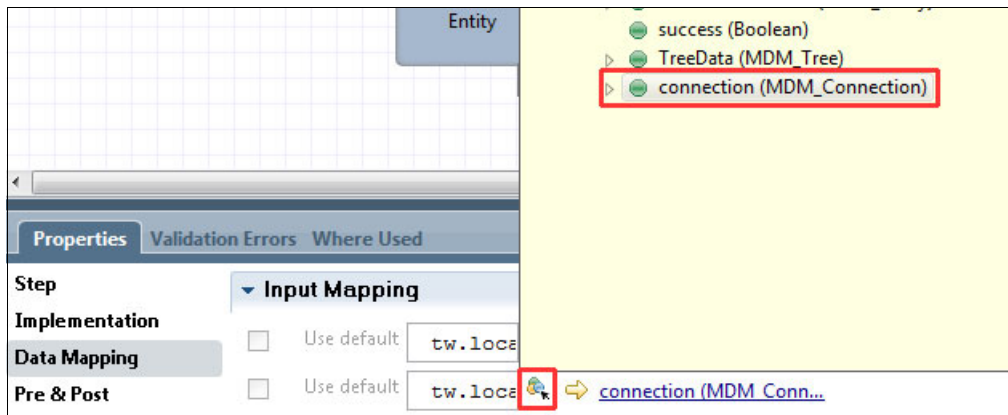


Figure 6-55 The connection variable

6.4.4 Binding a variable to the MDM Tree

The service integration node is now configured to save the changes that are made to the MDM Tree by a user. However, the TreeData variable that was created is not linked to the MDM Tree on the user interface.

To bind them together:

1. Open the **View Customer** node.
2. In the coach designer window, select the **MDM Tree**.

3. In the **Properties** view (Figure 6-56), on the **General** tab, next to the Binding property, click **Select**. In the pop-up window, select the **TreeData** variable that you created previously.

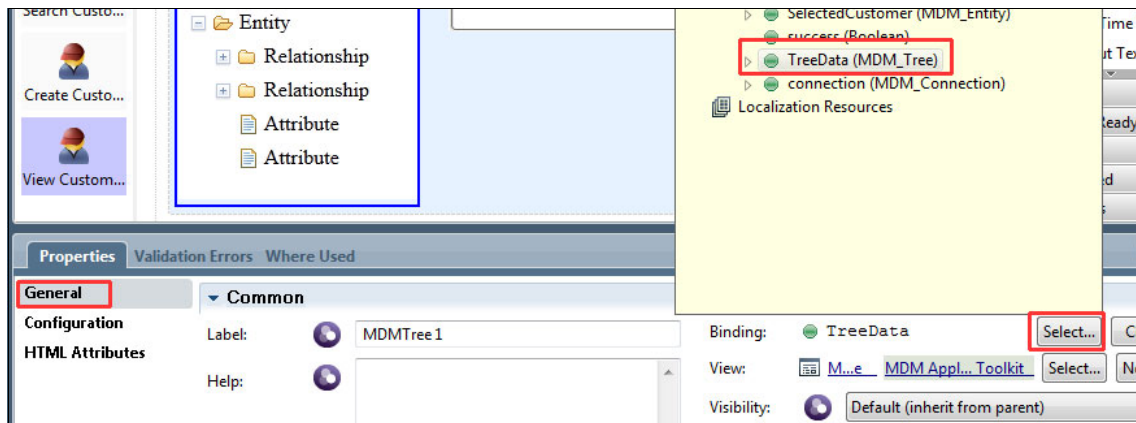


Figure 6-56 Linking the MDM Tree on the user interface

4. Add a **Save** button so that a user can save the changes that are made to the MDM Tree:
 - a. From the palette on the right side, expand the **Control drawer** and drag a **button** to the page designer.
 - b. Click the **new button**, and on the **Properties** tab, change the label to Save.

The MDM Tree control is now bound to the TreeData variable. Any changes that are made by a user to the data displayed in the MDM Tree widget are now stored in the TreeData variable. When a user clicks **Save**, the process passes this TreeData variable into the Save Tree integration service and persists the changes that are made by the user to the MDM engine.

6.4.5 Joining the process flow steps

To join the process flow steps together:

1. Switch back to the **Diagram** tab.
2. Select the **tool** from the palette on the right side.
3. Draw a relationship from the View Customer Details node to the Save Tree node. With the new relationship selected, in the **Properties** view, click the **Line** tab.

4. Ensure that End State Binding is set to the **Save** button (Figure 6-57). If it is not, click **Select**, and then click **Save** in the pop-up window.

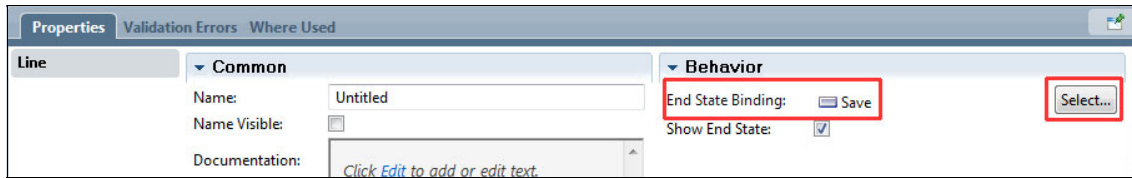


Figure 6-57 End State Binding

5. Create a relationship from the Save Tree node to the End node.

Figure 6-58 shows the completed process flow.

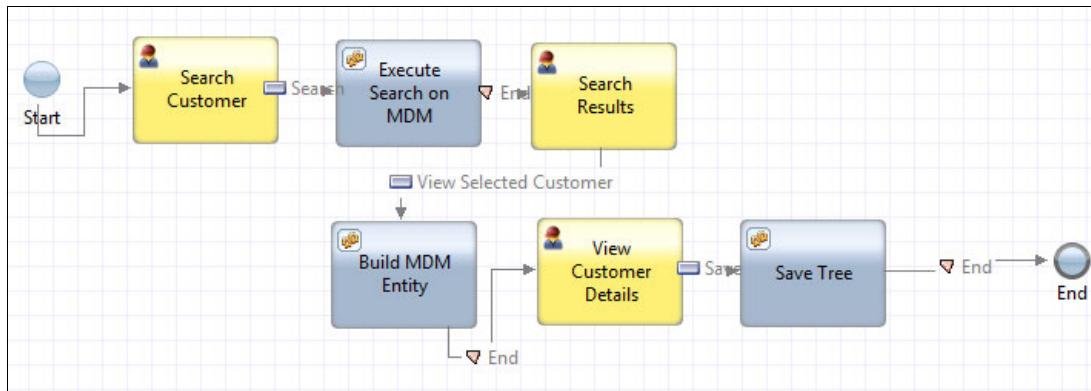


Figure 6-58 The completed process

6.4.6 Running the process

The process is now complete and can be tested within the Process Designer. To run the process:

1. Click the **icon** in the far upper-right corner of the Process Designer. Your web browser opens.
2. In the Search Customer coach window:
 - a. In the FirstName field, enter Anna.
 - b. In the LastName field, enter Franklin.
 - c. Click **Search**.

Customer name: Anna Franklin is the name of a customer that exists in the Individual data set that provided with InfoSphere MDM Standard Edition. If you are not using the data set that comes with InfoSphere MDM Standard Edition, you must enter the name of customer that exists in your data set.

The MDM engine returns a list of search results, which are displayed in the second coach.

3. Select a row in the results table, and then click **View Selected Customer**. When the MDM Tree loads, it shows the attributes of the selected customer record.
4. Expand the nodes of the hierarchy.
5. Drag the nodes to other parent nodes so that you manipulate the data in the hierarchy. Also, try deleting a node from the hierarchy.
6. After complete your changes, click **Save**. Your changes are then committed to the MDM engine.
7. Close your web browser, and run the process again. This time when you view the data in the hierarchy, you see the changes that you made.

6.4.7 Checkpoint

In this final section of the scenario, by using the integration services, you saw how to quickly extend the process so that changes to an MDM Tree can be persisted back into the MDM engine. The integration services handled the complexities about which services to call to reflect those changes on the MDM engine. The steps in this section demonstrated the configuration-based approach that BPM, MDM, and the InfoSphere MDM Application Toolkit offer to organizations that are looking to build process-orientated MDM powered applications.

6.5 Data stewardship processes within enterprise processes

This section highlights the use of data stewardship processes within enterprise processes.

6.5.1 Enterprise processes

Enterprise-wide processes include the following examples:

- ▶ Go to market
 - Use customer insights for target marketing
 - Dynamic pricing
- ▶ Order to cash
 - Account opening process automation
 - Automated order processing and fulfillment
- ▶ Product development
 - Reduce time to market of new products and services
 - Streamline production procurement sourcing

IBM Business Process Manager Advanced (BPM Advanced) supports the creation and implementation of enterprise-wide business processes. BPM Advanced includes a Business Process Execution Language (BPEL) engine and WebSphere Integration Designer for easy definition of processes by using enterprise-wide web services. For more information about IBM Business Process Manager Advanced, go to:

<http://www.ibm.com/software/integration/business-process-manager/advanced>

6.5.2 Integration of enterprise and MDM governance processes

Often MDM data governance-based processes can form a subset of tasks that make up part of greater enterprise-wide process, providing a vital role in the fundamental processes that run the wider organization. IBM InfoSphere MDM includes IBM Business Process Manager Express (BPM Express) to define and manage MDM governance-based processes.

Processes for BPM Advanced can call into processes for BPM Express so that an MDM data governance-based process can participate in wider processes. After you define the BPM Express process, you can import it into BPM Advanced Integration Designer. The entry and exit points for that process can then be integrated into the processes for BPM Advanced. The integrated process can then operate on a request-response model, where the BPM Advanced process starts the BPM Express process before waiting for a response. Alternatively, the integrated process operates on a request-only model, where the BPM Advanced process starts the BPM Express process and continues with the next steps in the process.

For more information about how BPM Express processes can integrate with BPM Advanced processes, see IBM Business Process Manager V8.0 Information Center at:

<http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/index.jsp>

At run time, when the BPM Advanced process is running, the BPM Advanced Engine initiates a call to start the BPM Express process. This call is initiated by using Service Component Architecture (SCA) over Internet Inter-ORB Protocol (IIOP) as illustrated in Figure 6-59.

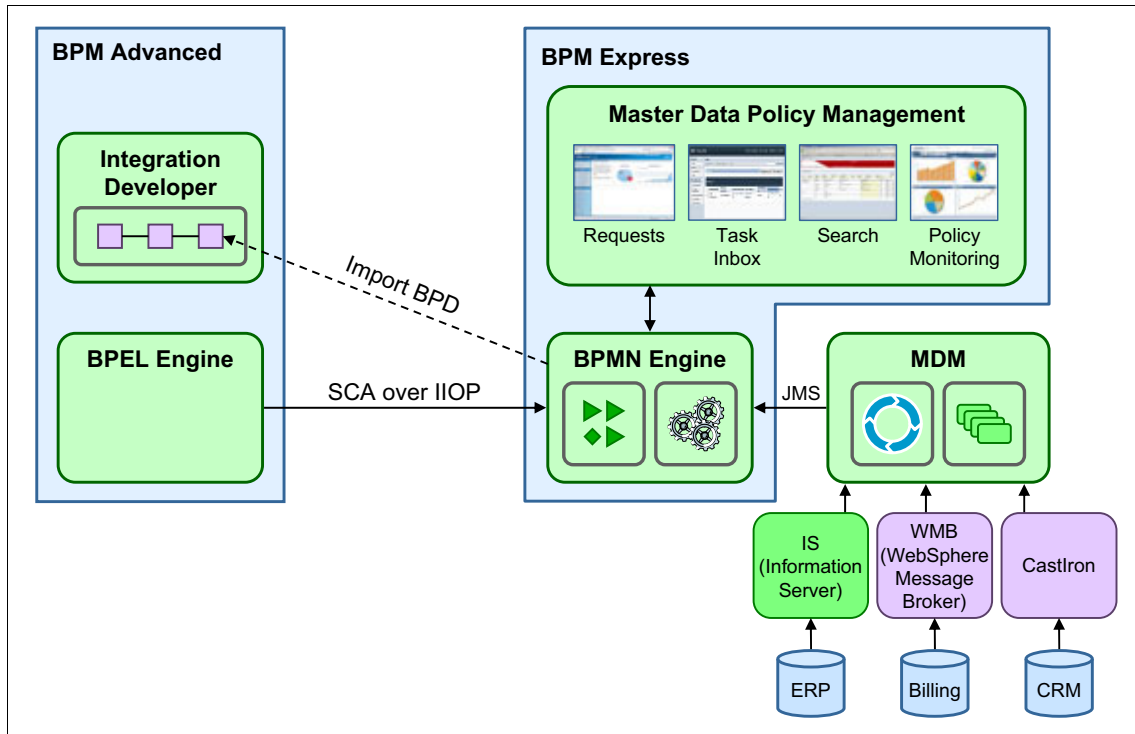


Figure 6-59 Starting the BPM Express process by using SCA over IIOP

6.6 Summary

This scenario demonstrated some of the capabilities that are provided by InfoSphere MDM Application Toolkit. The tasks showed how to use IBM Business Process Manager Process Designer to build an application that operates against mastered data that is stored in IBM InfoSphere MDM Standard Edition.

By using tools in the InfoSphere MDM Application Toolkit, you learned how to build process-orientated MDM powered applications that are based on a combination of drag-and-drop editing and wizard-based configurations. You saw how to use the prebuilt MDM data types, MDM integration services, and the MDM tree that is provided by the InfoSphere MDM Application Toolkit to build the application without any coding. This approach follows an iterative development process that facilitates greater buy-in from business users and lends itself to being incorporated into an Agile development project.

By using IBM InfoSphere MDM, you can simply and quickly build an MDM powered application without writing code. In addition, with InfoSphere MDM, you can place mastered data in the hands of business users and increase the return on investment of master data projects.



Master data policy enforcement with BPM Express

This chapter describes policy enforcement and how it fits with policy management. It includes examples that illustrate concepts from Chapter 3, “Master data governance and stewardship” on page 15. This chapter also describes the entitlements for use with the IBM Business Process Manager Express (BPM Express) supporting program. In addition, this chapter includes a look at the technical pattern that is used with policy enforcement to ensure master data policy (MDP) compliance. The key business drivers are introduced with guidance about what to listen for in requirements discussions.

This chapter introduces the following samples:

- ▶ Critical data change review and approval
- ▶ Policy remediation
- ▶ Entity consolidation

This chapter includes the following sections:

- Overview of policy enforcement
- Business case
- A technical perspective
- Assurances that master data is a trusted asset
- Business process scenarios

7.1 Overview of policy enforcement

Policy enforcement is one of the components that make up *policy management*. The other two components are *policy administration* and *policy monitoring*. Figure 7-1 highlights their complementary capabilities.

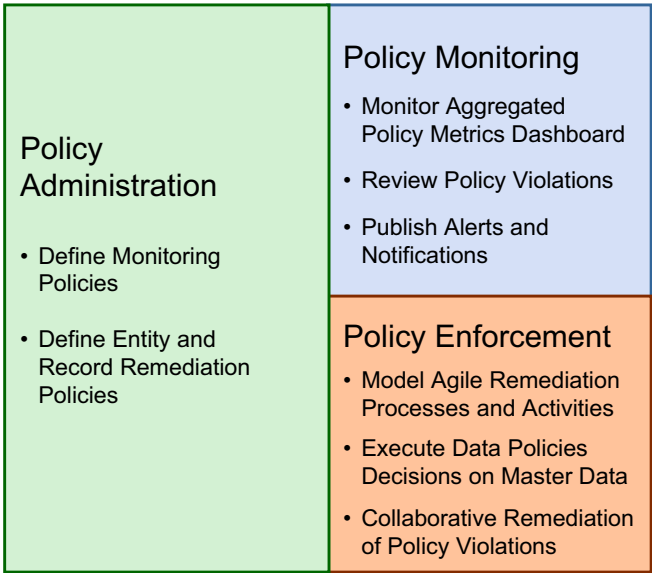


Figure 7-1 Logical architecture

The master data governance solution, which is based on a mix of methodology and technology, provides the assurances that master data can be trusted and that processes are empowered by the truth.

Before you explore each component, recall the definition of a *policy* as described in 3.1, “Definition and objectives of master data governance” on page 16. A *policy* is a characteristic of data that is required to satisfy one or more business processes or business outcomes. It must be concise and support measurement.

7.1.1 Components of master data governance

Policy administration captures and maintains an organization's policies for use in monitoring and enforcement. *Policy monitoring* provides visibility into master data health. *Master data health* is determined by translating a policy into a quantifiable measure of master data policy compliance.

Policies, when converted to metrics, show where data does not support the business. Then, business impact can be determined. Root cause analysis determines which policies to target for data quality improvement. These targeted policies are implemented in policy enforcement. *Policy enforcement* ensures improved data health by identifying noncompliant items and then bringing them into compliance through the appropriate remediation processes.

7.1.2 Data stewardship process example

A customer wants to trigger a data stewardship process whenever a patient, who is under age 18, does not have a guardian, is missing its social security number (SSN), or the SSN is invalid. The following policies are in place:

- ▶ Policy 1: If a patient is under age 18, the patient record must indicate a guardian for that patient.
- ▶ Policy 2: Patient records must indicate a valid SSN.

Records that do not comply with these policies are routed to the appropriate remediation process that is responsible for bringing the record into compliance.

7.1.3 Sales territory distribution and payouts example

A customer has a challenge in sales territories that are not correctly distributed and where payouts are misappropriated.

The root cause is that master data management (MDM) hierarchy changes are not appropriately approved or validated. In response, a policy, such as the following example, is defined and implemented in BPM Express:

“A data steward manager shall approve, reject, or modify changes to key customer hierarchy changes.”

The completion of this task by the data steward might result in alerting the sales territory management system so that the sales team can react to a significant merger.

7.2 Business case

Data quality improvement projects require a mechanism for enforcing critical policies. Policy enforcement supports these projects by providing assurances that master data is in compliance with these policies. Policy enforcement is built on BPM Express, which is an enterprise class BPM system. By using BPM Express, businesses can rapidly build process-centric applications through a What You See Is What You Get (WYSIWYG) widget (not coding based) design and deployment tool. This product is available to MDM customers free of charge to build MDM-centric applications. Note the following requirements:

- ▶ IBM Process Server Express
 - Use metric: Processor value unit (PVU)
 - Entitlement: 240 (~2 processors), with 75 - 100 users per processor
 - Use metric: Authorized user
 - Entitlement: 200
- ▶ IBM Process Center Express
 - Use metric: PVU
 - Entitlement: 120 (~1 processor)
- ▶ IBM Process Designer
 - Use metric: Authorized user
 - Entitlement: 2
- ▶ Support for warm failover on production: No clustering
- ▶ Express Edition runs on Windows, Linux, and IBM AIX®.
- ▶ Extra PVU, authorized users, AIX, or clustering support require more BPM V7.5 Standard edition licenses.

InfoSphere MDM v10.1 provides MDM enforcement samples to help users enforce review and approval of critical data change to remediate noncompliant policies, and to manage entity consolidation. Two samples work on the MDM Standard, Advanced, and Enterprise Editions:

- ▶ Critical data change
- ▶ Policy remediation

One sample, entity consolidation, works on MDM Advanced and Enterprise Addition. The following sections describe these samples in more detail.

7.3 A technical perspective

BPM Express is the heart of policy enforcement. BPM Express supports the interrogation of master data changes and identification of noncompliant items. It also brings noncompliant items into compliance, routes data changes for review and approval, and supports entity consolidation activities. The characteristics of a policy determine which type of process must be implemented:

- ▶ The process for a critical data change policy requires appropriate individuals to be notified. These individuals use windows to view the changes and then to approve or reject them.
- ▶ The process for remediating a noncompliant policy requires routing the appropriate individual. The process provides the individual a list of noncompliant policies in a window to correct this master data.
- ▶ A process for managing entity consolidation shows a list of suspected duplicates, from a master record, with UIs to link or collapse the data.

Each process might require manager review.

7.3.1 Aspects of policy enforcement

This section highlights the general aspects of policy enforcement. No single example can capture all the aspects. To begin, refer to the patient example that was introduced in 7.1.2, “Data stewardship process example” on page 121.

Policies are implemented as business rules called *Business Action Language* (BAL) rules. The BAL rules component provides a rule editor that rule designers use to author business rules by using natural language technology. Using natural language, instead of JavaScript, to author rules means that no programming expertise is required to create business rules, and the rules are easier for people to read and understand. Master data updates are fed to these rules for evaluation.

Processes are started based on the outcome of the BAL rules. In the example in 7.1.2, “Data stewardship process example” on page 121, a master record update results in an invalid SSN attribute. A process to bring the record back into compliance is started. By using automated tasks, the process reads more data from the hub or other systems that are necessary to support the UIs and business logic. UIs are displayed by using human tasks.

A designated user (for example, a data steward or sales representative) is made aware of an SSN violation by viewing the task inbox or by a notification the user received. This user selects the activity to view the policy violation and then edits the data, bringing it back into compliance. The outcome of this user action can be

sent by email to a designated individual for manual update in a source system. Alternatively, the outcome might be programmatically persisted back to the hub or source system. Manager approval or review can be added to the process as needed.

Figure 7-2 illustrates the physical architecture to support the aspects of policy enforcement processes.

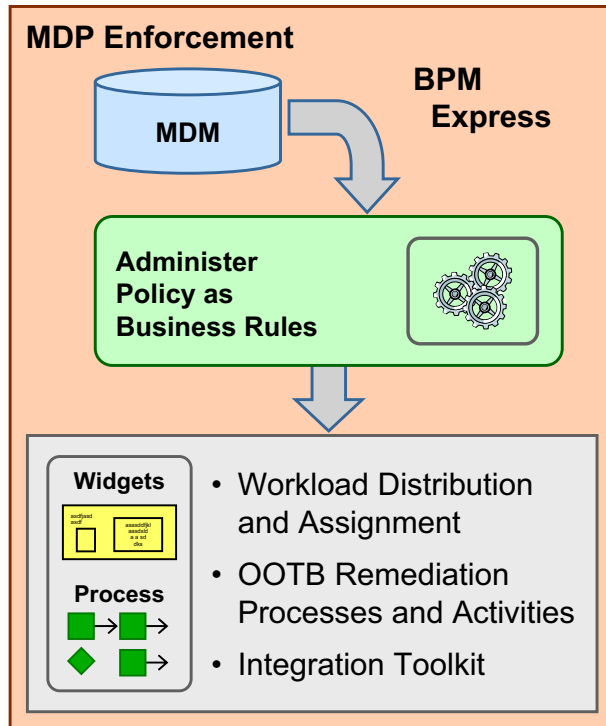


Figure 7-2 Physical architecture

7.4 Assurances that master data is a trusted asset

The need to implement policy enforcement has the following key business drivers:

- ▶ Business processes do not use master data because they do not trust it.
- ▶ Business processes are inefficient because the data they use is not accurate.
- ▶ Business process decisions cannot be made because data is missing.

You might be hearing the following phrases:

- ▶ “I do not trust the source of this data.”
- ▶ “This data does not contain the data that I require.”
- ▶ “This data is not in the format I require.”
- ▶ “This data is incomplete.”
- ▶ “This data has low quality.”

A key aspect to any operational MDM project is to consider the business processes that consume master data. If the success of your MDM projects depends on improving business processes, you must include the requirements of these business processes in the MDM requirements. Without understanding the needs of the business process that will ultimately consume the master data, you have no assurance that the master data will support the business process. Master data governance provides assurance by monitoring and enforcing the requirements of the business processes as policies.

7.5 Business process scenarios

This section highlights the following business process scenarios:

- ▶ Critical data change review and approval
- ▶ Policy remediation
- ▶ Entity resolution

7.5.1 Critical data change review and approval

For some organizations, certain data changes have a significant impact on the business. These attributes drive business process decisions that impact market campaigns, pricing discounts, compliance levels, and support contracts. The attributes are critical to the business, and changes to them must be reviewed. Through the critical data change review and approval process, owners of the consuming systems and process can review and approve updates to master data attributes with the control that they require.

For example, the customer service and account management departments both contribute data to the MDM hub of an organization. An individual's contact information is often updated by the customer service representative (CSR) when the individual calls in for help or questions.

For high profile accounts, such as for clients who receive a platinum level of service, a call from an individual might trigger downstream activities. For example, a new point of contact is assigned to a key account or the address of the account owner changes. In this case, the sales team might respond by

contacting a new customer contact, by assigning a new sales representative based on a new customer location, or by extending new offers that are based on recent questions. This approach can be wasteful and potentially risky if these activities are performed in error.

The account manager who is assigned to these key accounts can review the updates and then approve or reject them to ensure that the sales team responds only to accurate information.

Figure 7-3 illustrates the systems that act as a source for MDM data and a system that uses the MDM data.

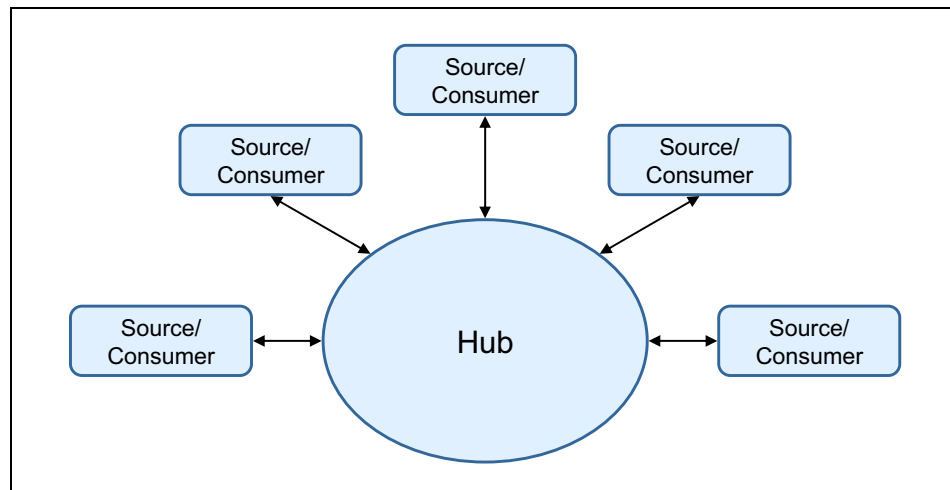


Figure 7-3 Sources and consumers for MDM data

In Figure 7-4, changes in one source system are submitted to MDM.

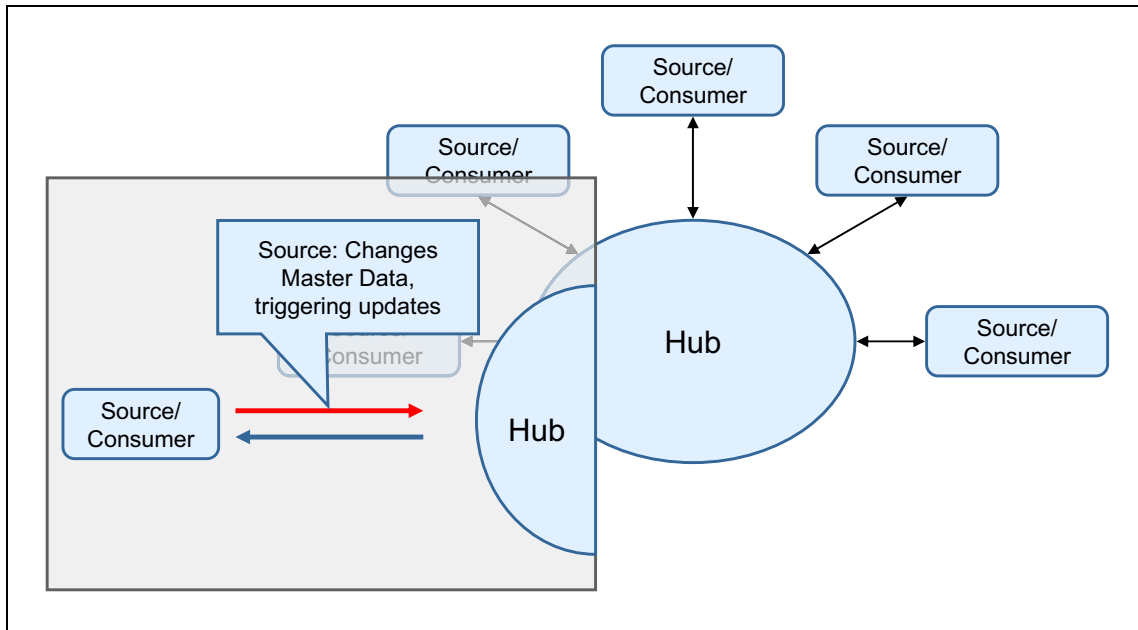


Figure 7-4 Submitting source changes to MDM

Figure 7-5 shows the business rules that are applied to the update. In the previous example, the account manager sees only updates for key customers. These business rules can use any single or multiple set of attributes as criteria for defining the information that should be reviewed. After an update is identified as one that needs review, a business rule determines the right individual or group to whom to route the update. A process automatically prioritizes and routes the work, guiding users through the appropriate decisions.

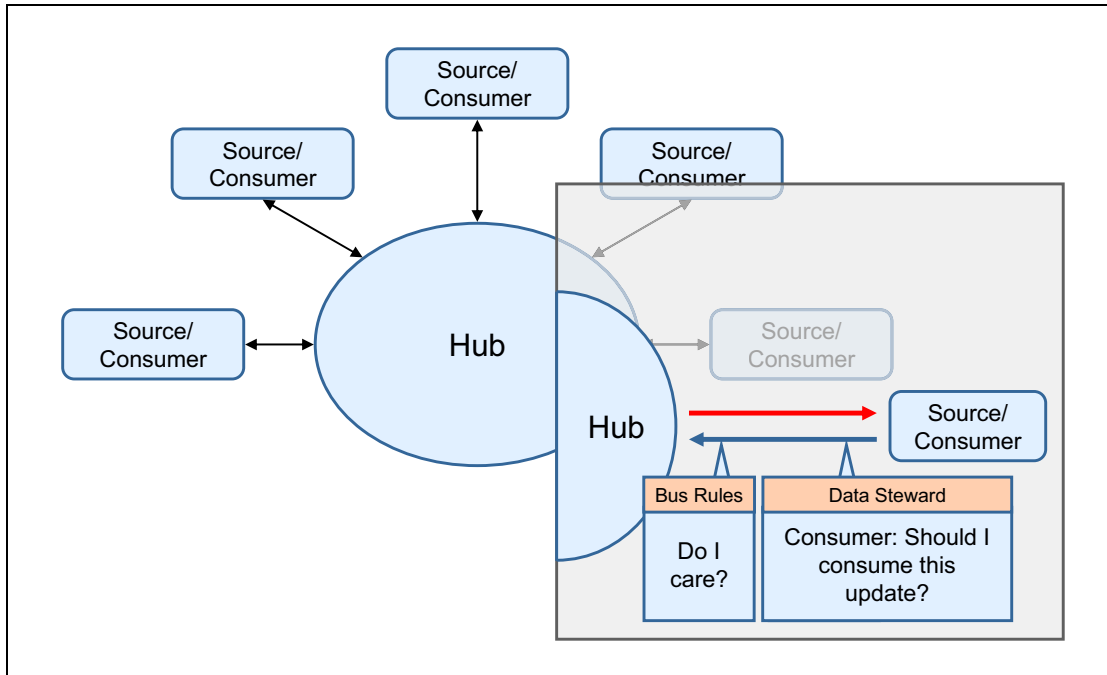


Figure 7-5 Applying business rules to updates

7.5.2 Policy remediation

For some organizations, consumption of MDM data by business processes is contingent on trust. The business owners who are responsible for these processes must trust in the data before they are willing to include it. Trust is more than just being accurate and complete. They want to ensure that their business rules and constraints are represented. Business owners must trust that MDM data is compliant with their requirements (or the requirements of the data governance council). Policy remediation provides the assurances that MDM data is compliant.

Continuing with the example from the previous section, the organization has a policy that all platinum accounts have an account manager assigned. The

organization also requires a customer's SSN and address to be valid. In addition, if a patient is under age 18, they must have guardian.

Figure 7-6 shows an MDM update that results in a change to the name, ID, and address of the golden record.

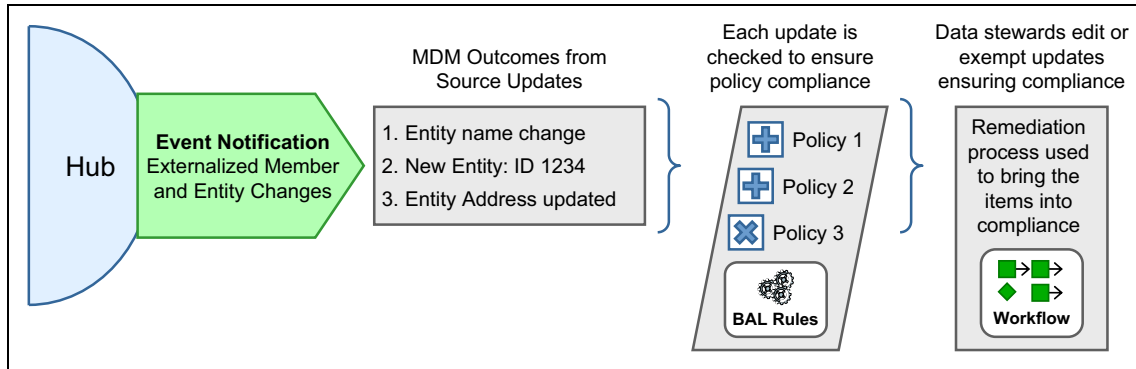


Figure 7-6 Updating a business policy

Policies are implemented as business rules and are used to identify noncompliance by interrogating MDM changes. Noncompliant updates are routed to data stewards. In prescriptive UIs, data stewards have the choice to bring the update into compliance by modifying or exempting the update.

7.5.3 Entity resolution

For some organizations, automated record resolution (or consolidation) is not enough. Investment in manual consolidation and review is necessary when the high profile customer of an organization has duplicate records. Choosing which class of service, billing address, and primary contact is the correct one is critical to maintain high customer satisfaction. Correct linking of high profile customer accounts across lines of business ensures a consist customer experience. Through the entity consolidation process, by editing the golden record directly, data stewards can preview the automated collapsed record and identify areas of potential conflict. They can also manually select the attributes that survive or provide new attributes. Business rules should be added to this process to ensure that only high priority items receive data steward attention.

For example, two source systems (A and B) each create an account, A1 and B1. MDM patching identifies that account A1 and account B1 are the same customer. Account A1 has a platinum level of service, and account B1 has a bronze level of service. Because of the priority of the customer, the data steward is notified of this task and works to collapse the two accounts into the golden record. The data

steward decides to use the contact information from the A1 account based on personal experience with the customer. The Account Manager is notified of their changes to the customer account and can then validate these changes.

Figure 7-7 shows three source system records in the green rectangle in the upper left corner of the figure. The lower two records in the green rectangle are newly added to the MDM hub. MDM matching identifies these three records as suspects or candidates (probable matches of each other). The entity consolidation process routes this task to the data steward to validate which items are matches and to select the attributes from the source records that should survive in the golden record.

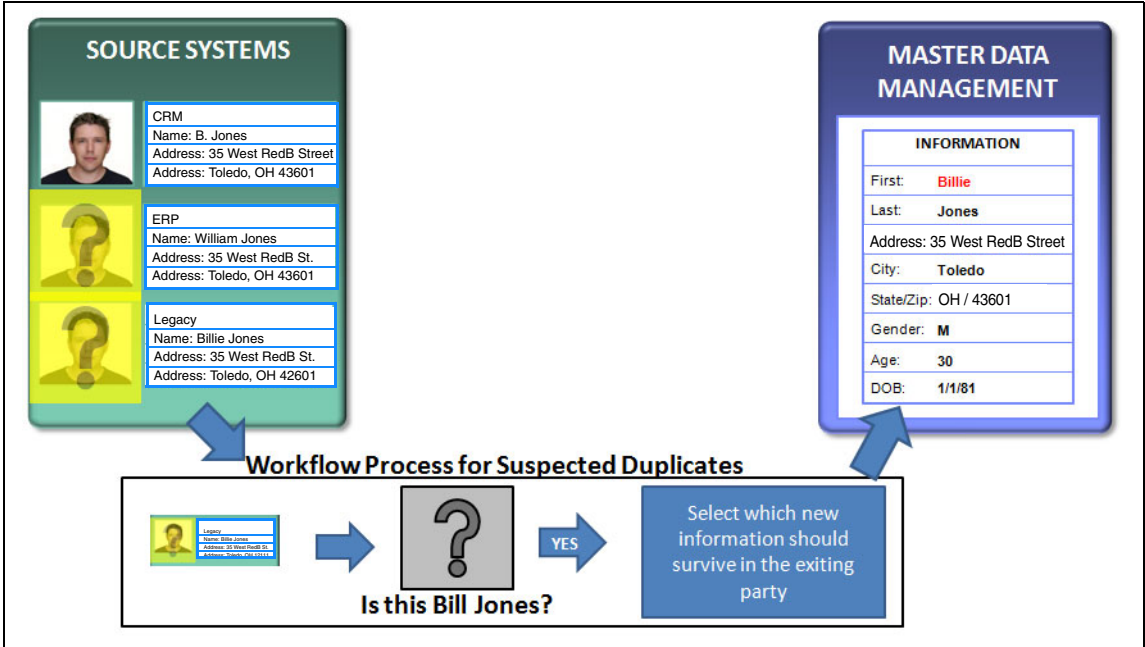


Figure 7-7 Entity consolidation process



Using and extending MD policy enforcement samples

This chapter describes the aspects of using the IBM InfoSphere Master Data Management (MDM) Server and IBM Business Process Manager (BPM) to implement business processes around master data. IBM provides the following set of samples on BPM-based data stewardship and policy enforcement process through the IBM Web Membership community:

- ▶ MDM Entity Resolution (MDMSER)
- ▶ Policy Enforcement (PE)
- ▶ Critical Data Change Verification (CDVP)

You can find the process for downloading these samples by searching for *installing master data policy enforcement* in the IBM InfoSphere MDM Version 10.1 Information Center at:

<http://pic.dhe.ibm.com/infocenter/mdm/v10r1/index.jsp>

In this chapter, any references to the term *samples* refer to the samples provided by IBM through the IWM, unless otherwise explicitly stated. Although the samples are referenced to explain some of the aspects of development, the patterns and mechanism described apply to any business process that was developed around MDM.

Before you import and customize the sample, install and configure the MDM and BPM software. For detailed installation instructions, see the product documentation. Although both MDM and BPM products can be installed on the same physical system, for demonstrating the capabilities of BPM and MDM working together, install these products in a distributed manner on different hardware to achieve optimal performance. This approach is important in a typical production setup where the business critical data is being processed.

This chapter includes the following sections:

- ▶ Configuring MDM Standard Edition for messaging
- ▶ Configuring InfoSphere MDM Advanced Edition for messaging
- ▶ Importing samples
- ▶ Configuring process applications
- ▶ Integrating BPM in the enterprise
- ▶ Extending the samples based on a business case
- ▶ Reporting for success

8.1 Configuring MDM Standard Edition for messaging

The event handler publishes the event message from the external system to the BPM WebSphere Application Server.

8.1.1 Configuring the event handler

To configure the event handler:

1. Create an InfoSphere MDM Standard Edition server instance. When you create the instance, select the option to use the embedded event manager of the instance. For instructions on how to create an instance, see the IBM InfoSphere MDM Version 10.1 Information Center at:

<http://pic.dhe.ibm.com/infocenter/mdm/v10r1/index.jsp>

2. Open a command window and navigate to the `\scripts` directory in the MDM home directory, for example `C:\IBM\Initiate\Engine10.1.0\scripts`.
3. Use the **madconfig** utility to configure the event handler:

```
madconfig configure_instance_eventhandler
```

When you configure the event handler, see Table 8-1 on page 133 as a reference. This table shows the prompts that you see when you run the **madconfig** utility, and samples of the values for each prompt.

Table 8-1 Configuration utility prompts

Prompt	Sample value	Description
Enter an existing IBM Initiate Master Data Engine instance name	mdsbpmone	Type the name of the existing IBM Initiate Master Data Engine instance that you want to configure to use the event handler.
Enter the IBM Initiate Event Manager JMS connection factory name	javax.jms.QueueConnectionFactory	Type the Java Naming and Directory Interface (JNDI) name of the queue connection factory that you created when you configured the BPM server to deploy the message-driven bean.
Enter the IBM Initiate Event Manager publishing destination name	eventqueue	Type the destination queue name, which is how the destination is referred to in an MDM environment. This name can be any string name. The value is used to configure the event notification settings in the MDM workbench. The value should match publish.destination.name in the com.initiate.server.event.cfg configuration file of the MDM Standard Edition instance.
Enter the IBM Initiate Event Manager publishing destination user	admin	Type the administrative user ID for the BPM process center.
Enter the password scheme that you will be using for the IBM Initiate Event Manager publishing destination	plain	
Enter the plain text password for the IBM Initiate Event Manager publishing destination	admin	Type the administrative user password for the BPM process center.
Would you like to add another IBM Initiate Event Manager publishing destination	n	

For more information, see Table 2, “Embedded event handler configuration worksheet,” in the *Event notification worksheets* topic of the IBM InfoSphere MDM Version 10.1 Information Center at:

<http://pic.dhe.ibm.com/infocenter/mdm/v10r1/index.jsp>

8.1.2 Event handler settings

After you configure the event handler, the `com.initiate.server.event.cfg` file has the entries that are shown in Table 8-2. This configuration file is in the `\conf` directory in the MDM server instance. In Table 8-2, the required configuration entries were modified manually to point to the correct Initial Context Factory and the provider URL. The provider URL is the WebSphere Application Server instance where BPM is running, and Context Factory is the WebSphere Application Server default Context Factory.

Table 8-2 Event handler configuration file entries

Entry	Value	Required or Optional
<code>#publish.environment.initialcontextfactory.key</code>	<code>java.naming.factory.initial</code>	Required
<code>#publish.environment.initialcontextfactory.value</code>	<code>com.ibm.websphere.naming.WsnInitialContext Factory</code>	Required
<code>#publish.environment.jndiprovider.key</code>	<code>java.naming.provider.url</code>	Required
<code>#publish.environment.jndiprovider.value</code>	<code>corbaname:iiop:localhost:2809</code>	Required
<code>#publish.environment.key.1</code>	<code>java.naming.security.authentication</code>	Optional
<code>#publish.environment.value.1</code>	<code>simple</code>	Optional
<code>#publish.environment.key.2</code>	<code>java.naming.security.principal</code>	Optional
<code>#publish.environment.value.2</code>	<code>cn=Manager,o=IBM,c=US</code>	Optional
<code>#publish.environment.key.3</code>	<code>java.naming.security.credentials</code>	Optional
<code>#publish.environment.password.3</code>	<code>admin</code>	Optional
<code>#publish.environment.key.4</code>		Optional
<code>#publish.environment.value.4</code>		Optional

Table 8-3 shows similar entries for an MQ queue as presented for event handler in Table 8-2.

Table 8-3 Event handler configuration file entries for MQ queues

Entry	Value
<code>#publish.environment.initialcontextfactory.key</code>	<code>java.naming.factory.initial</code>
<code>#publish.environment.initialcontextfactory.value</code>	<code>com.ibm.mq.jms.context.WMQInitialContextFactory</code>
<code>#publish.environment.jndiprovider.key</code>	<code>java.naming.provider.url</code>

Entry	Value
#publish.environment.jndiprotider.value	<hostname:port>/MDMBPMSRVCONCHANNEL

8.1.3 Deploying the Event Handler

After the event handler is configured, deploy the event handler to the instance. To deploy the event handler, run the **madconfig deploy_instance_eventhandler** batch file command. Then, respond to the event handler prompts, which are listed in Table 4.

Table 4 Prompts for the madconfig command

Prompt	Sample value	Description
Enter an existing Master Data Engine instance name:	mdsbpmone	Type the name of the existing Master Data Engine instance that you want to configure to use the event handler.
Enter the directory that contains Event Publisher JAR files:	C:\Temp\	Specify the path for the Java archive (JAR) files. The JAR files are used to create the Initial Context Factory and to create the queue connections and sending messages to the Queue. The JAR files are different for WebSphere Application Server and WebSphere MQ. The following JAR files make up the event handler for WebSphere Application Server JMS implementation: <ul style="list-style-type: none"> ▶ bootstrap.jar ▶ com.ibm.ffdc.jar ▶ com.ibm.hpel.logging.jar ▶ com.ibm.tx.jta.jar ▶ com.ibm.tx.util.jar ▶ com.ibm.ws.admin.core.jar ▶ com.ibm.ws.bootstrap.jar ▶ com.ibm.ws.emf.jar ▶ com.ibm.ws.runtime.jar ▶ com.ibm.ws.sib.client.thin.jms_8.0.0.jar ▶ com.ibm.ws.sib.server.jar ▶ com.ibm.ws.sib.utils.jar ▶ com.ibm.wsspi.extension.jar ▶ j2ee.jar ▶ org.eclipse.core.runtime_.jar ▶ org.eclipse.emf.common.jar ▶ org.eclipse.emf.commonj.sdo.jar ▶ org.eclipse.emf.ecore.jar ▶ org.eclipse.equinox.common_.jar ▶ org.eclipse.equinox.registry.jar

Prompt	Sample value	Description
(Continued) Enter the directory that contains Event Publisher JAR files:	C:\Temp\	<p>The bootstrap.jar and j2ee.jar files are in the <WAS_HOME>/lib folder.</p> <p>The com.ibm.ws.sib.client.thin.jms_8.0.0.jar file is in the <WAS_HOME>/runtimes folder.</p> <p>All the other JAR files are in the <BPM_INSTALL_ROOT>/plugins folder (for example, C:\BPM\V8\plugins). The event handler is deployed and initialized by these JAR files. The process is optimized because these JAR files take less time than when given the entire \runtimes folder of WebSphere Application Server 8 to deploy.</p>

8.1.4 Configuring the hub model

For MDM Standard Edition, to send notifications, configure the hub model to enable event notifications. MDM workbench can be used to configure the hub model to create notifications in the XML format that BPM can use.

To enable event notification on hub model:

1. Select **Events**.
2. On the **Event Notification** tab:
 - a. For the Destination Name, enter a unique name for the destination, for example, eventqueue.
 - b. For the Publisher Name, enter the JNDI name of the destination queue, for example, jms/eventqueue.
 - c. In the Publisher Arguments, enter the required arguments to construct the notification message in BPM format. Example 8-1 shows the format that BPM expects for the message.

Example 8-1 BPM format

```

<eventmsg>
  <!-- The process app acronym and event name are required: The
  snapshot and UCA name are optional -->
  <event processApp="[acronym]" snapshot="[snapshot_name]"
  ucaname="[UCA_name]"> [event_name] </event>
  <!--Optional: The name of the queue for the event to run in-->
  <queue> [queue name] </queue>
  <!--Any parameters the UCA may require--
  <parameters>
    <parameter>
      <key>param1</key>
      <value>![CDATA[value1]]</value>

```



```

        </parameter>
    </parameters>
</eventmsg>

```

The processApp, eventName, and key are mandatory, and other attributes are optional. The arguments should be provided as name-value pairs that are separated by the carat (^) symbol, for example:
processApp=PE^eventName=PolicyEnforcementMessage^key=MDMEventMessage
In (Figure 8-1).

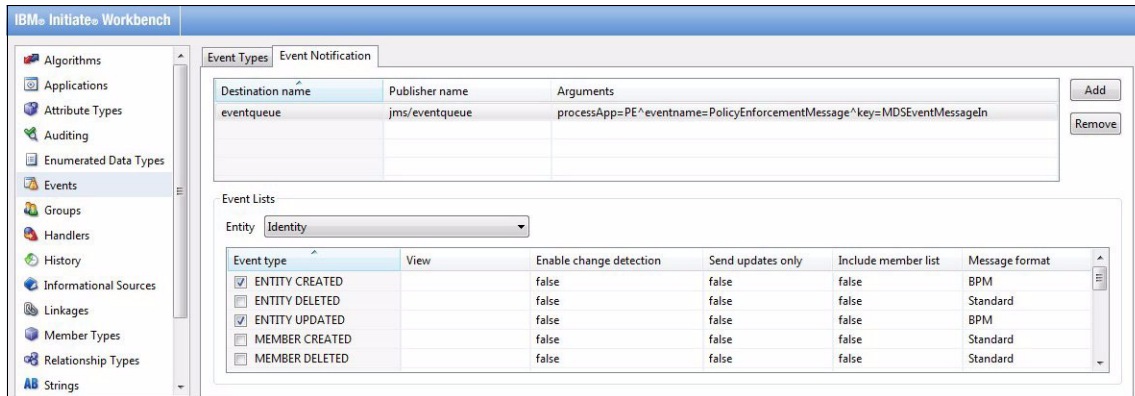


Figure 8-1 Configuring the hub model

3. Select only the required events for which notification should be enabled, for example, **ENTITY UPDATED**.
4. For Message Format, ensure that only **BPM** is selected.
5. Deploy the hub configuration.

8.1.5 Configuring IBM BPM Express

To enable IBM BPM Express Edition to receive messages from InfoSphere MDM Standard Edition, on BPM:

1. Set the Provider End Point URL of `javax.jms.QueueConnectionFactory` to `bootstrapMessaging`:
 - a. Open the BPM WebSphere Application Server administrative console.
 - b. Click **Resources** → **JMS** → **Queue Connection Factories**, and select **Queue connection factory** with the JNDI name of **javax.jms.QueueConnectionFactory**.
 - c. Set Target Inbound transport chain to `InboundBasicMessaging`.
 - d. Set the Provider Endpoints to `BootstrapBasicMessaging`.

2. Disable Secure Sockets Layer (SSL) on BPM version 8.0:
 - a. From the navigation panel, click **Security** → **Global Security** → **CSlv2 outbound communications** → **Client Certificate Authentication** → **Supported and Transport** → **SSL-Supported**.
 - b. In the WebSphere Application Server administrative console, click **Service Integration** → **Buses**.
 - c. Locate the bus name, for example, PROCSSVR.localhostNode06Cell.Bus. Then click **Security**.
 - d. Click **Allow the use of all defined transport channel chains**, and then click **Apply and Save**.
3. Restart BPM.

8.2 Configuring InfoSphere MDM Advanced Edition for messaging

The BPM samples are packaged with a set of Python scripts that can be used to configure the WebSphere Application Server instances where MDM Advanced Edition and BPM are deployed. The complete script content is shown in at the end of this section.

8.2.1 MDM Advanced Edition configurations

Configuring MDM Advanced edition required creation of Queues, Queue Connection Factory, Alias Destination, and Foreign Bus Connection on the WebSphere Application Server instance on which MDM is deployed. The following steps can be used to complete the configuration:

1. Go to the <MDM_SERVER_INSTALL_HOME>\profiles\<ProfileName>\bin folder. In this example, we go to the C:\Program Files (x86)\IBM\WebSphere\AppServer\profiles\AppSrv03\bin folder.
2. From a command line, run the **MDMBPMSetupApp.py** script. In the script, as shown in the following example, substitute the values for your environment:

```
wsadmin -lang jython -f MDMBPMSetupApp.py <MDMClusterName>
<MDMNodeName> <MDMServerName> <MDMSIBusName> <BPMSIBusName>
<BPMEventDestinationName> <BPMMessagingEngineName>
<BPMSIBEndpointPort>
```

The following example shows the script with the substituted values:

```
wsadmin -lang jython -f C:\Temp\MDMBPMSetupApp.py None MDMNode02
server1 MDM.SIB.server1 PROCSVR.BPMNode05Cell.Bus
eventqueueDestination.BPMNode05.server1
BPMNode05.server1-PROCSVR.BPMNode05Cell.Bus 7283
```

8.2.2 Configuring metadata

InfoSphere MDM Advanced Edition can send notification messages in the XML format that BPM can use. You can enable this feature by using metadata that defines the notifications to send to the BPM destination.

The metadata is configured in the BPMNOTIFICATIONTYPE table (Table 8-5).

Table 8-5 BPM notifications

Column name	Comment
NOTIF_TYPE	The primary key of the notification type that is defined in NOTIFICATIONTYPE table. It is a foreign key to the NOTIFICATIONTYPE table.
BPM_PROCESS_APP_NAME	Acronym of the process application name that is defined in the BPM server.
BPM_PROCESS_SNAPSHOT_NAME	Snapshot name of the process application that is defined in the BPM server.
BPM_UCA_NAME	Name of the undercover agent on the BPM server to which the notification is intended.
BPM_EVENT_NAME	Name of the event as understood by the undercover agent on the BPM server.
UCA_EVENT_KEY	Name of the key to set in the notification message for the parameter value. This key is the XML element type variable that is associated with the UCA. If the value is populated, MDM uses mdmNotificationXML as the key.
BPM_QUEUE_NAME	Name of the queue to which the undercover agent is listening.
DESCRIPTION	A short description of the process.
EXPIRY_DT	Expiration date of the BPM notification type, such as the date after which the BPM notification type is not valid.
LAST_UPDATE_DT	The date on which the record was updated last.
LAST_UPDATE_USER	The user ID who modified the record last.

Update this table manually by using SQL because the InfoSphere MDM Server does not provide any services to populate this metadata table.

8.2.3 Configuring IBM BPM Express

To enable IBM BPM Express Edition to receive messages from InfoSphere MDM Advanced Edition, complete the following configuration steps on BPM. These steps create foreign bus connection on the IBM WebSphere Application Server instance on which BPM is deployed.

1. Go to the <BPM_SERVER_INSTALL_HOME>\profiles\<ProfileName>\bin folder. In this example, we go to the C:\IBM\BPM\v8.0\profiles\ProcCtr02\bin folder.
2. From a command line, run the **BPMSetupApp.py** script. In the script, as shown in the following example, substitute the values for your environment:

```
The usage is wsadmin -user admin -password admin -lang jython -f
BPMSetupApp.py <BPMClusterName> <BPMNodeName> <BPMServerName>
<BPMSIBusName> <MDMSIBusName> <MDMMessagingEngineName>
<MDMSIBEndpointPort> <AdminUserIdOfBPM>
```

The following example shows the script with the substituted values:

```
wsadmin -lang jython -f C:\Temp\BPMSetupApp.py None BPMNode05
server1 PROCSVR.BPMNode05Cell1.Bus MDM.SIB.server1
MDMNode02.server1-MDM.SIB.server1 7281 admin
```

8.3 Importing samples

You can import the samples into BPM Process Designer. They can be used to import any IBM BPM process application.

1. Open BPM Process Designer V8.0.
2. Click **Import Process application**.
3. In the Import Process App dialog box (Figure 8-2):
 - a. Click **Browse**.



Figure 8-2 Import a sample application

- b. Select the process application file (.twx extension) that you want to import.
- c. Click **OK**.

The process application is listed in Process Designer after is imported (Figure 8-3).

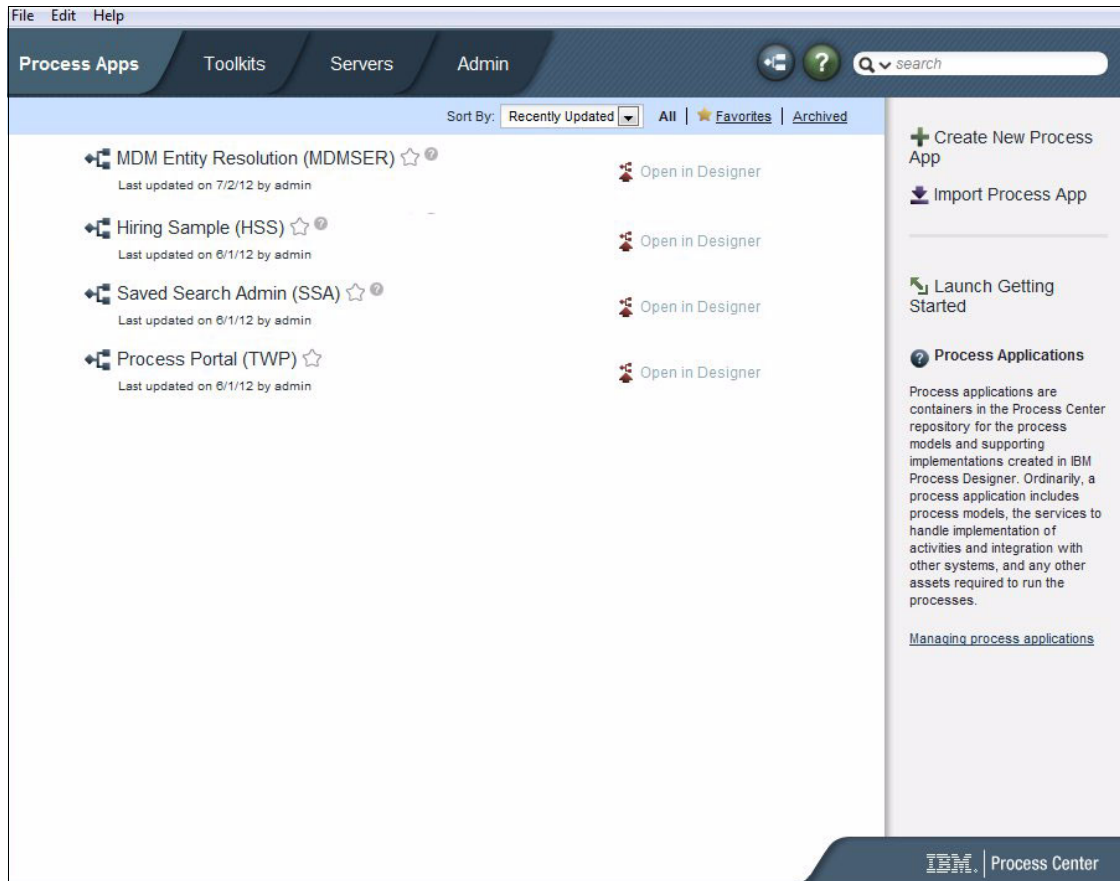


Figure 8-3 Process applications list

4. To open the application, in Process Designer, click **Open** in Designer link for customization and editing. Figure 8-4 shows the opened application.

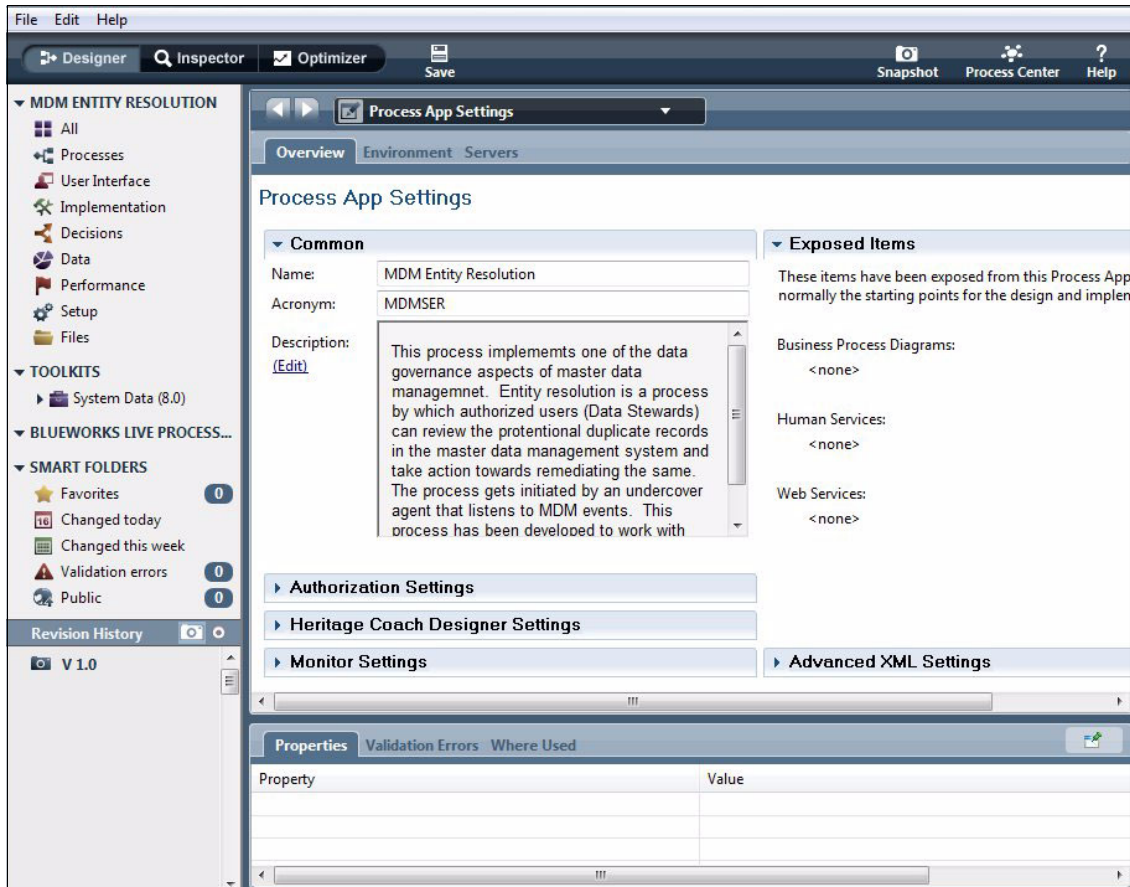


Figure 8-4 Opened application

8.4 Configuring process applications

You can implement the business process so that it is configurable. IBM BPM software provides a mechanism to make process applications configurable. A simple and easy mechanism is provided through exposed process values (EPV), which are global variables that are available throughout the process application. As a common practice, you can use this feature for process-level configurations and to define constants such as field labels, service endpoint URLs, and other environment details. To edit the EPVs, open the process application in Process

Designer. You can access the EPVs from the Data category in the left pane of Process Designer (Figure 8-5).

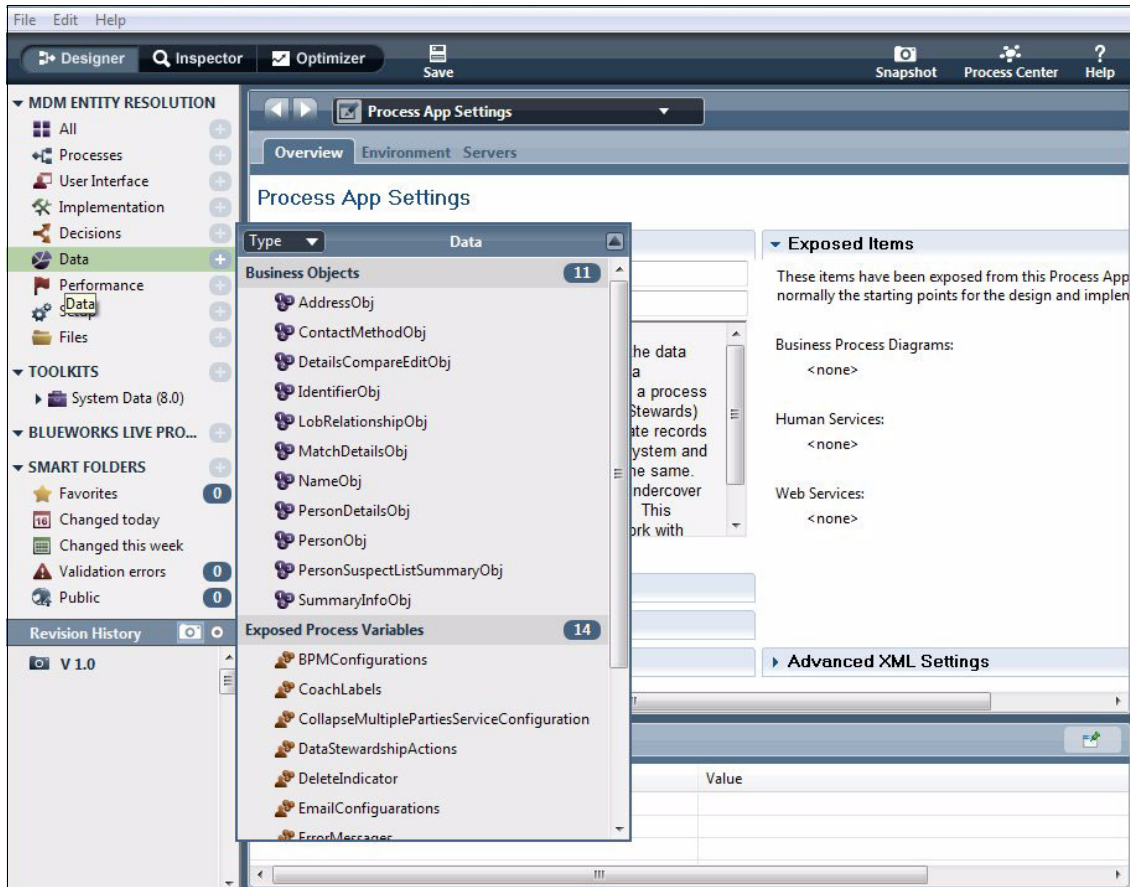


Figure 8-5 Editing EPVs

By using the samples that are provided by IBM, you can configure process applications easily in your environment. For more information about configuring the samples, search for *developers samples for InfoSphere MDM Server* in the IBM InfoSphere MDM version 10.0 Information Center at:

<http://pic.dhe.ibm.com/infocenter/mdm/v10r0m0/index.jsp>

8.5 Integrating BPM in the enterprise

The IBM Business Process Manager server provides open standards to integrate with other components within the enterprise IT environment. The following integration touch points are supported by BPM:

- ▶ Web services
- ▶ Java integration
- ▶ Integration through JMS messaging

BPM also supports easy integration with the SMTP mail server and WebSphere Operational Decision Management (WODM) rules engine.

This section describes the integration techniques and mechanisms that were used in implementing the samples. The samples primarily used the integration features, such as web services, Java, and JMS messaging that are available for immediate use. Because InfoSphere MDM Server Standard and Advanced Editions expose well-designed SOA services and JMS channels, it is common for BPM and MDM to be integrated through these protocols.

8.5.1 Integration patterns used in the samples

The samples that are provided by IBM use web services, Java, and JMS integration to implement the process applications.

MDM Entity Resolution

In the MDM Entity Resolution (MDMSER) process implementation, BPM and MDM Advanced Edition are integrated through JMS messaging and Java. How this integration was implemented is described later in this chapter. After you import this process into BPM Process Designer, you have to configure only the ServiceEndPointURL variable in MDMConfigurations EPV. The process implementation contains prebuilt integration nodes for all the necessary web service invocation.

The following MDM services are used in this process:

- ▶ GetAllOperationalCodeTypesByLangId
- ▶ GetParty
- ▶ GetAllPartySuspects
- ▶ ComparativePreviewCollapseMultipleParties
- ▶ CollapseMultipleActiveParties
- ▶ UnmarkPartiesAsSuspect
- ▶ UpdateSuspectStatus

In this process, a generated SOAP client is used to start MDM web services. You can generate the SOAP client and types by using the **wsimport** command. For information about using the **wsimport** command, see the official Java documentation at:

<http://docs.oracle.com/javase/6/docs/technotes/tools/share/wsimport.html>

You can also generate the SOAP types and client by using IBM Rational® Software Architect. For more information, see the IBM Rational Software Architect Information Center at:

<http://pic.dhe.ibm.com/infocenter/rsahelp/v8/index.jsp>

The communication for the InfoSphere MDM Server to BPM happens through JMS messaging. InfoSphere MDM Standard Edition supports configurations to send messages through a JAM implementation that is supported by WebSphere Application Server. These configurations are available for immediate use with InfoSphere MDM Server.

Policy Enforcement

The Policy Enforcement (PE) process implementation sample demonstrates the integration of InfoSphere MDM Standard Edition through JMS messaging and web services. InfoSphere MDM Standard Edition exposes web services through the web services toolkit and ESOA toolkit. In this sample, the services that are exposed through the web services toolkit were used for integration. These services are more generic in nature and are hub model agnostic.

Critical Data Change Verification

The Critical Data Change Verification (CDCV) process implementation sample demonstrates the integration of InfoSphere MDM Standard Edition through JMS messaging and web services. In this sample, the services that are exposed through the ESOA toolkit were used for integration. These services are more hub-specific, and therefore, offer a higher ease of use when handling data. This process also showcases the use of history data services of InfoSphere MDM Standard Edition through Java integration.

8.6 Extending the samples based on a business case

This section describes how to extend and customize the samples. The processes are kept at a generic level so that they apply in general to any process application. You can use BPM Process Designer to achieve the required customization in most of the scenarios. If you are using Java integration or web services, the development of the Java or web services code requires more

development environments such as IBM Rational Software Developer or Rational Software Architect.

Most of the process development or process customization for MDM-specific stewardship implementation includes the following tasks among others:

- ▶ Adding a field in the UI
- ▶ Creating a services integrator node to start an MDM service
- ▶ Modifying the display test of a label
- ▶ Retrieving more details from MDM server about a particular entity

The following sections describe the different aspects of the defining and customizing processes.

8.6.1 Adding and removing an attribute

Stewardship activities or any human task requires the data to be displayed on the UI. During the process flow, data is retrieved from MDM Standard or Advanced Editions through the exposed services. The retrieved data can be displayed on the UI for review as part of a human task implementation.

The manner in which the data is presented varies depending on the purpose. For example, if you want to view details about an individual or an organization with related details, such as name and address identifiers, the best way is to display it in a form format. If you want to compare the data of two entity records, such as names or addresses, you can use a tabular view with each column to see the details of a specific entity, with attributes that match each other. Alternatively, if you want to view a collection of entities, you can use a tabular view, where each row of the table shows a record.

Often the data that is displayed for stewardship processes includes MDM entities, such as an individual or organization that consists of numerous attributes. Not all attributes are of importance to a data steward or for a process application. Depending on the goal of the process application, a set of attributes should be presented to the user. This subset of attributes might change over time based on the needs of the implemented business process. It requires implementing a user interface that is a flexible to support such changes.

The same data that is retrieved from InfoSphere MDM Server might have to be displayed in multiple ways, depending on the requirement. Often only a subset of data needs to be shown on the user interface. Therefore, as a good practice, define a coach-specific data model to decouple the format in which the data is received and the format in which to display the data. Additionally, this approach helps to parallel development of a process, the coaches, and independent testing of the coaches with stubbed data.

Adding more attributes

To add more attributes for display:

1. Decide on the set of attributes that is required to be displayed.

This list depends on the goal of business process. The decision drives, for example, the right choice of service invocation and inquiry level for InfoSphere MDM Server Advanced Edition. Based on this decision, you can retrieve details from the InfoSphere MDM Server at the right volume.

2. Modify the data objects that are associated with user interface.

Modify the data objects that are defined within the BPM environment accordingly. This aspect depends on the manner in which the data object is designed in the BPM environment. For example, when data is to be displayed in a tabular view as name-value pairs, the associated data model can be a list of name-value pair objects. Any additional MDM attribute that is required to be displayed requires another name-value pair object to be added to the list.

3. Modify the script node that composes the object.

Displaying the data on the user interface required the data object associated with the use interface to be composed. Modify the script that is associated with the data object to be composed.

4. Modify the user interface.

The need to change the user interface is driven mostly by the design of the user interface. For example, if the user interface is designed as a form view, add the additional attributes to the user interface. Because BPM supports UI customization, use care to ensure that it does not inadvertently introduce regression.

Removing attributes that are currently being displayed follows similar steps. Again, the removal of attributes is driven by the design of the user interface. For a form view, modifying the files alone from the UI is sufficient. If the UI design is a tabular view to display list of objects, modify the manner in which the list object is created to filter the objects that do not need to be displayed. Use care when you remove any attributes from the data object structure that is associated with the user interface because the data object might be used at other places in the process.

To elaborate more on the steps, consider the sample MDM Entity Resolution (MDMSER). Consider the requirement of displaying more details in the party summary view.

1. Identify the data element to be displayed. Display the party line of business (LOB) relations in the user interface. The required attribute of MDM to be displayed in this case is RelatedLobValue. Also, the label for this attribute in the UI should be the value of the MDM Field LobRelationshipValue.

2. Change the data objects. In this case, the data objects already provide support to store details of LOB relations of party entity. If the support is not provided, modify the data object. Create LobRelationshipObj with the required attributes. Alter the PersonObj to contain a list of objects of type LobRelationshipObj.
3. Modify the script that composes the object to display in the UI. To change the scripts:
 - a. Open the Human Service implementation named **Entity Remediation Service** (Figure 8-6).

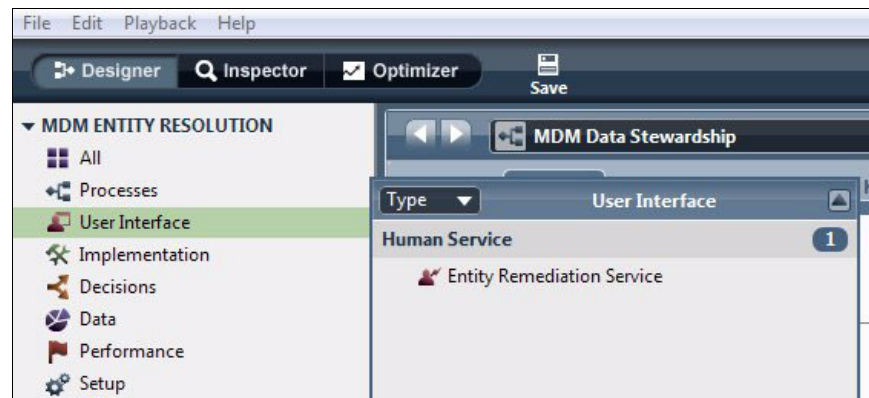


Figure 8-6 Remediation Service

- b. Open the script node named **Prepare Summary Object for UI**. The script in the node consists of the data object that is displayed in the model (Figure 8-7).

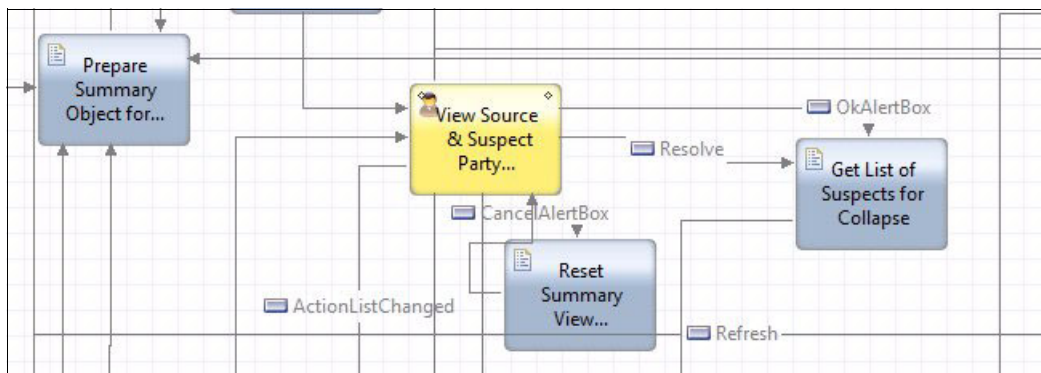


Figure 8-7 Opening the script node

- c. Follow the same pattern as other attributes in the script node to set the LOB details. Declare a list variable with the name `lobRelationshipsInfo` of type `SummaryInfoObj`. Example 8-2 shows the script to be added.

Example 8-2 Script snippet

```
contactMethodType = null;
}

//***** New code - Begin *****/
tw.local.lobRelationshipsInfo = new tw.object.listOf.SummaryInfoObj();
var lobRelationshipsListIndex = 0;
var lobRelationshipsType = null;
for(var i=0; i < tw.local.sourcePersonObj.lobRelationships.listLength ; i++){
    if ((tw.local.sourcePersonObj.lobRelationships[i].lobRelationshipValue != null)
    &&
        (tw.local.sourcePersonObj.lobRelationships[i].lobRelationshipValue != ""))
    {
        lobRelationshipsType =
tw.local.sourcePersonObj.lobRelationships[i].lobRelationshipValue;
    }
    else
    {

if(tw.local.suspectPersonObjList[tw.local.suspectPartyNum].lobRelationships[i] !=
null) {
        if
((tw.local.suspectPersonObjList[tw.local.suspectPartyNum].lobRelationships[i].lobRela
tionshipValue != null) &&
(tw.local.suspectPersonObjList[tw.local.suspectPartyNum].lobRelationships[i].lobRelat
ionshipValue != ""))
        {
            lobRelationshipsType =
tw.local.suspectPersonObjList[tw.local.suspectPartyNum].lobRelationships[i].lobRelati
onshipValue;
        }
    }

    if ((lobRelationshipsType != null) && (lobRelationshipsType != ""))
    {
        tw.local.lobRelationshipsInfo[lobRelationshipsListIndex] = new
tw.object.SummaryInfoObj();
```

```

        tw.local.lobRelationshipsInfo[lobRelationshipsListIndex].attributeLabel =
lobRelationshipsType;
            if
                ((tw.local.sourcePersonObj.lobRelationships[i].relatedLobValue != null)
                && (tw.local.sourcePersonObj.lobRelationships[i].relatedLobValue !=
                ""))
            {

tw.local.lobRelationshipsInfo[lobRelationshipsListIndex].sourcePartyAttributeValue =
tw.local.sourcePersonObj.lobRelationships[i].relatedLobValue;
            }

if(tw.local.suspectPersonObjList[tw.local.suspectPartyNum].lobRelationships[i] !=
null) {
            if
                ((tw.local.suspectPersonObjList[tw.local.suspectPartyNum].lobRelationships[i].related
LobValue != null) &&
                (tw.local.suspectPersonObjList[tw.local.suspectPartyNum].lobRelationships[i].relatedL
obValue != ""))
            {

tw.local.lobRelationshipsInfo[lobRelationshipsListIndex].suspectPartyAttributeValue =
tw.local.suspectPersonObjList[tw.local.suspectPartyNum].lobRelationships[i].relatedLo
bValue;
            }
        }

        lobRelationshipsListIndex = lobRelationshipsListIndex+1;
    }

    lobRelationshipsType = null;
}
//***** New code - End *****/

tw.local.suspectActionList = new tw.object.listOf.String();

```

This script node handles scenarios when the person entity does not have any suspects that are associated with it.

d. Insert the following snippet as indicated.

Example 8-3 Code snippet for person entity with no associated suspects

```
contactMethodListIndex = contactMethodListIndex+1;
    }
}

//***** New code - Start *****/

tw.local.lobRelationshipsInfo = new tw.object.listOf.SummaryInfoObj();
var lobRelationshipsListIndex = 0;
for(var i=0; i < tw.local.sourcePersonObj.lobRelationships.listLength ; i++){
    if ((tw.local.sourcePersonObj.lobRelationships[i].lobRelationshipValue != null)
    &&
        (tw.local.sourcePersonObj.lobRelationships[i].lobRelationshipValue != ""))
    {
        tw.local.lobRelationshipsInfo[lobRelationshipsListIndex] = new
tw.object.SummaryInfoObj();
        tw.local.lobRelationshipsInfo[lobRelationshipsListIndex].attributeLabel =
tw.local.sourcePersonObj.lobRelationships[i].lobRelationshipValue;
        if ((tw.local.sourcePersonObj.lobRelationships[i].relatedLobValue != null) &&
(tw.local.sourcePersonObj.lobRelationships[i].relatedLobValue != ""))
        {

tw.local.lobRelationshipsInfo[lobRelationshipsListIndex].sourcePartyAttributeValue =
tw.local.sourcePersonObj.lobRelationships[i].relatedLobValue;
        }

        lobRelationshipsListIndex = lobRelationshipsListIndex+1;
    }
}

//***** New code - End *****/

tw.local.lobRelationshipsInfo = new tw.object.listOf.SummaryInfoObj();
```

4. Modify the user interface. Insert a new table in the coach below the “Contact Methods” table control. Here also the table setting follows a similar setting as the other table controls. Associate the new table with the variable lobRelationshipsInfo, which was previously declared.

Figure 8-8 shows the altered coach.

Identifiers		
<small>.attributeLabel</small>	<small>.sourcePartyAttributeValue</small>	<small>.suspectPartyAttributeValue</small>
Repeating		
Contact Methods		
<small>.attributeLabel</small>	<small>.sourcePartyAttributeValue</small>	<small>.suspectPartyAttributeValue</small>
Repeating		
LOB Relationship		<small>tw.local.lobRelationshipsInfo</small>
<small>.attributeLabel</small>	<small>.sourcePartyAttributeValue</small>	<small>.suspectPartyAttributeValue</small>
Repeating		
ActionListChanged Refresh OkAlertBox CancelAlertBox		
<script type="text/javascript">		

Figure 8-8 Modifying the user interface

After you make the customizations, when the process is run, the additional attributes are available on the user interface. Figure 8-9 shows the modified user interface.

Addresses		
Primary Residence	2146 Summerfield Ave., ELIOT, Massachusetts, 03903	
Business		92 Enterprise Park, Bar Harbour, Massachusetts, 03907
Primary Residence	1 Main Street, BERWICK, Massachusetts, 39019	
Secondary Residence		2 Baker Street, CAPE NEDDICK, Massachusetts, 03907
Summer Residence		1 Main Street, BERWICK, Massachusetts, 39019
Business	92 Enterprise Park, Bar Harbour, Massachusetts, 03907	
Identifiers		
Passport Number	US8128424	US8128424
Driver's Licence Number	K6150 7869-610101	K6150 7869-610101
Social Security Number	251 762 592	251 762 592
Contact Methods		
Personal Email		akutch@yahoo.com
Home Telephone	207 402 4751	207 402 4751
Personal Email	annie@aol.com	
Home Telephone		207 952 4862
LOB Relationship		
Owner		Life Insurance
Owner	Retail Banking	
Owner	Group Insurance	

Figure 8-9 Customized user interface

8.6.2 Changing the attribute label

When customizing process applications, often you are required to change the attribute labels depending on the use case that is being implemented. For example, a front-desk process receives data about an individual and completes a registration. This process is common and has applicability in multiple use cases, such as patient registration in a hospital or customer registration in a bank. In these use cases, the type of data that is collected includes attributes such as name, address, phone number, and date of birth. However, in a patient registration system, it is relevant to have the attribute label as “Patient name” rather than “Customer name.” BPM software provides easy ways to accomplish such requirements.

Also, other considerations can exist for localizations, such as displaying locale-specific label text on the user interface. BPM provides a mechanism to manage and accomplish localization requirements.

In some use cases, the display labels must be populated dynamically, and the display text is modeled as text controls rather than label controls. BPM provides an easy-to-use mechanism to accomplish these requirements as well.

Localization resources

A *localization resource* is a mechanism that is supported by BPM to manage translated label strings and display the label text based on the locale. You can create localization resources within BPM and associate the labels on the user interface. BPM also supports export and import of the localization resource to edit them out side of BPM environment.

Creating localization resources

To create localization resources for the needs of the process application:

1. From the Library pane in Process Designer (Figure 8-10), click the + icon next to **Setup**.

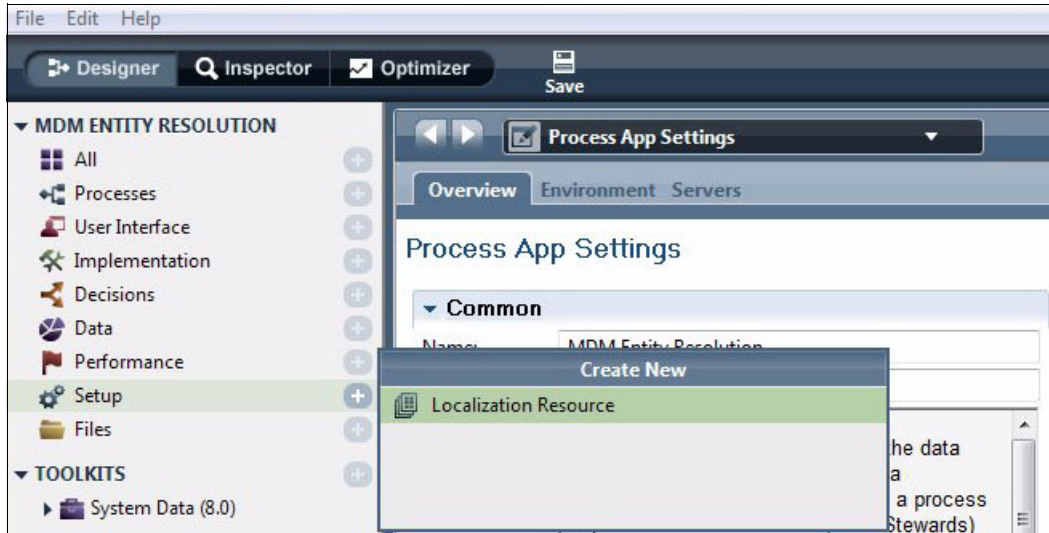


Figure 8-10 Using localization resources

2. In the Create New window, select **Localization Resource**.
3. Enter a valid name for the localization resource, for example CoachLabels, and then click **Finish**.

The newly created resource is opened for editing (Figure 8-11).

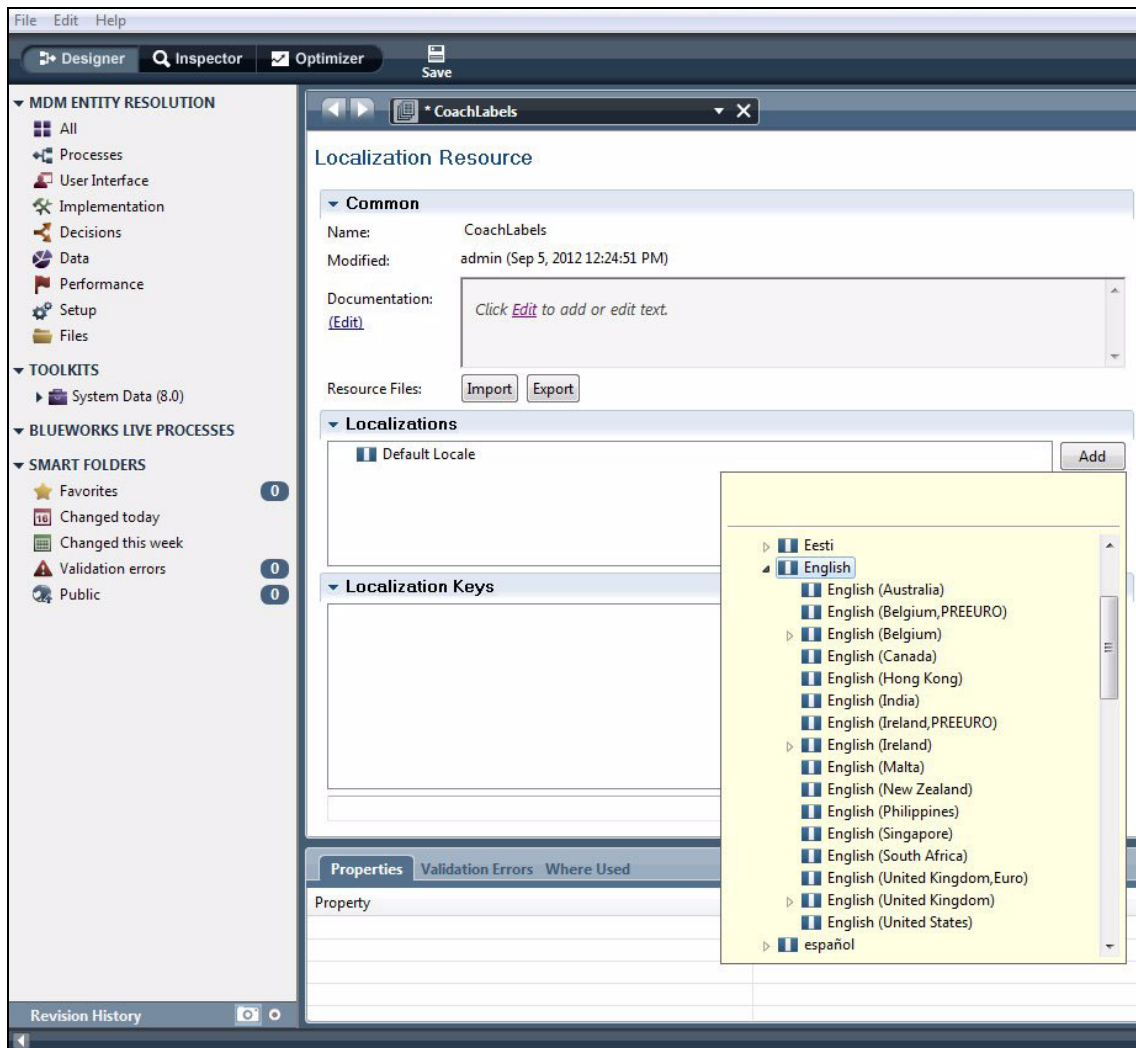


Figure 8-11 Localization Resource

4. In the Localization Resource pane (Figure 8-12):
 - a. In the Localizations section, click **Add** to add the required locales.
 - b. In the Localization Keys section, click **Add** to add the required keys. The keys that are defined are associated with the labels in the user interface.

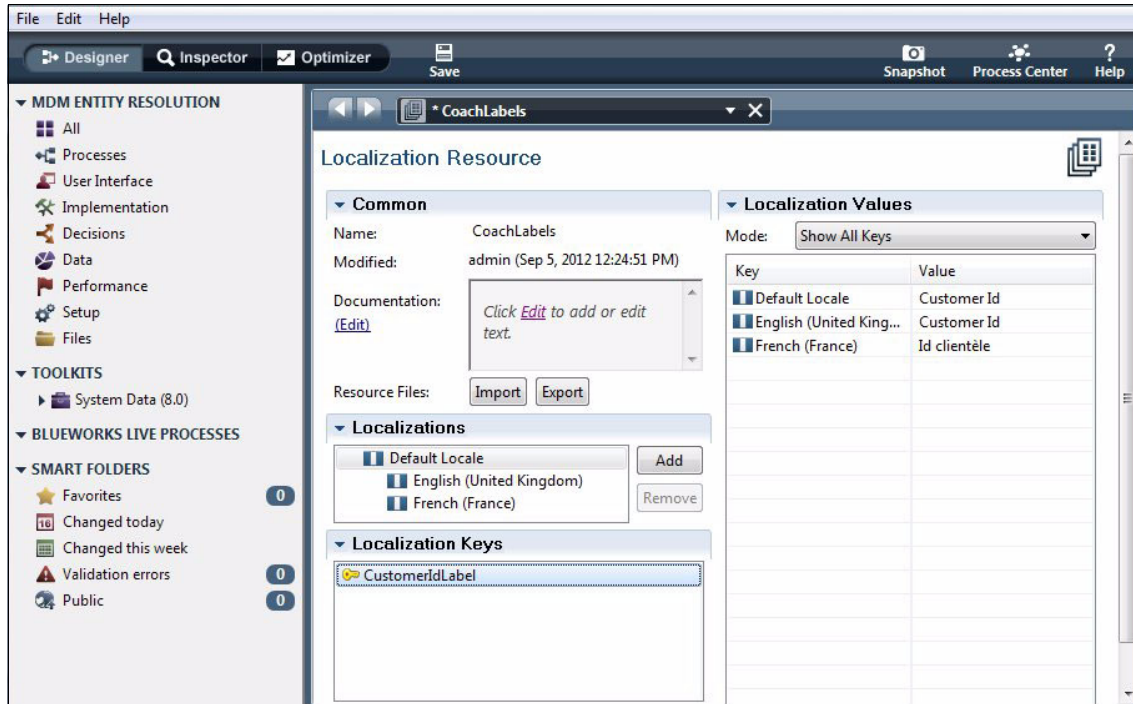


Figure 8-12 Define localization keys

5. In the Localization Values section, add the local specific label text for the keys defined.

Associating localization resources with the attribute label

The created localization resources can be associated with the attribute label. The attribute label shows the locale-specific text, depending on the locale in which the process application is run (Figure 8-13).

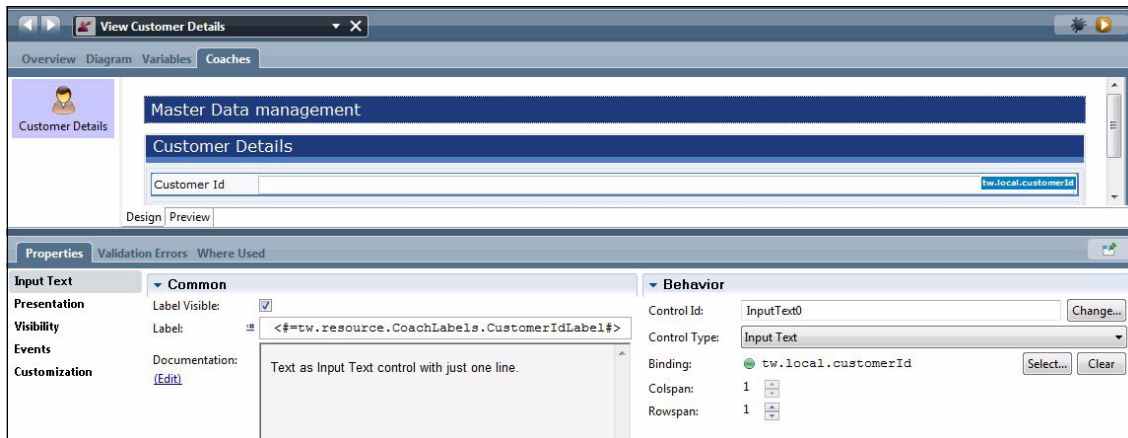


Figure 8-13 Localization text

To associate the resource key with the label:

1. On the **Variables** tab, click **Link Localization** to link the required localization resource.
2. On the **Coaches** tab, select the attribute that you want to localize.
3. Click the **Properties** view, and select the **Label Visible** check box.
4. In the Label text box, specify the key that you want to apply to the Label Control.

By using this mechanism, you can easily manage translated text for the labels and modify the label text. The label is not hardcoded in the process application user interface. Therefore, modifications to the values in the localization resources that are associated with the label are reflected automatically without changing any other artefact.

Exposed process values

You can use EPV variables to provide flexibility to change attribute names. EPVs are useful in scenarios where localization support is not required. They are also useful where attribute names are not designed as label controls and the labels must be populated dynamically. Before you can use an EPV variable, you must define and link it to the UI or service implementations in which it is used.

For more information about EPVs, search for *Creating exposed process values (EPVs)* in the IBM Business Process Manager, V8.0 Information Center at:

<http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/index.jsp>

Consider a use case where you have to display a list of values and the label for these values depends on the metadata of the attribute. Most of the times, the metadata is cryptic, and a display label for the attribute might be different. In such cases, using EPVs is convenient. The standard display labels can be defined as EPVs. By using a script, you can create a map between the attributes metadata and the display label to help to display the label dynamically based on the metadata of the attribute. The flexibility of changing the attribute display labels can be achieved easily by changing the EPV variable.

In scenarios where localization support is required and the display text must be populated dynamically, use a combination of EPVs and localization resources.

In the samples that IBM bundles with the InfoSphere MDM Server, the EPVs are used extensively to define the attribute labels.

EPVs in MDM Entity Resolution

Consider the MDM Entity Resolution (MDMSER) sample in Figure 8-14.

The screenshot displays the 'CoachLabels' configuration page for an 'Exposed Process Value'. The interface is divided into several sections:

- Common:** Contains fields for 'Name' (CoachLabels), 'Modified' (admin (Sep 9, 2012 7:34:45 PM)), and 'Documentation' (Click [Edit](#) to add or edit text. (Edit)).
- Details:** Contains fields for 'Feedback E-mail Contact' and 'External Description' (Click [Edit](#) to add or edit text. (Edit)).
- Exposed Process Value Variables:** A list of variables with 'Add' and 'Remove' buttons. The list includes: GENDER (String), ADDRESS_LINE_ONE (String), PREFERRED_LANGUAGE (String), PREFERRED_CONTACT (String), IDENTIFIER_NUMBER (String), DECEASED_DATE (String), **FAMILY_NAME (String)** (selected), SUSPECT_COUNT_MESSAGE_SUSPECTS (String), SUSPECT_DUPLICATE_STATUS (String), CONTACT_METHOD_TYPE (String), and CITY (String).
- Variable Details:** Contains fields for 'External Name' (FAMILY_NAME), 'Variable Name' (FAMILY_NAME), 'External Description' (Click [Edit](#) to add or edit text. (Edit)), 'Default Value' (Family name), 'In-Progress Tasks Use New Values' (checkbox), and 'Variable Type' (Script, with buttons for 'Select...' and 'New ...').
- Exposing:** A section at the bottom of the configuration page.

Figure 8-14 Use of EPVs in Entity Resolution

The attribute labels are defined in the EPV named CoachLabels. Consider the FAMILY_NAME variable in this EPV. This EPV variable is used in the user interfaces that are defined in the process. This variable holds the “Family name” value. The label on the user interface is displayed as shown in Figure 8-15.

Master Data Management

Manage Suspected Duplicates

Resolve

1 of 3 Suspects

Previous

Next

	Source Party Id 2000000000000001	Suspected Duplicate Party Id 2000000000000004
Given name	Annabelle	Annabelle
Family name	Kutcher	Kutcher
Match score		63 - 0 (G,G1Name,SName,DOB,Addr,SSN)
Match category		A1
Suspect duplicate status		Parties are Suspect Duplicates

Figure 8-15 Using the Family name variable

To change the display text to “Last name,” modify the value of EPV variable FAMILY_NAME to Last name as shown in Figure 8-16.

CoachLabels

Exposed Process Value

Common

Name: CoachLabels

Modified: admin (Sep 9, 2012 7:30:00 PM)

Documentation: [Click Edit to add or edit text.](#) ([Edit](#))

Exposed Process Value Variables

- GENDER (String)
- ADDRESS_LINE_ONE (String)
- PREFERRED_LANGUAGE (String)
- PREFERRED_CONTACT (String)
- IDENTIFIER_NUMBER (String)
- DECEASED_DATE (String)
- FAMILY_NAME (String)**
- SUSPECT_COUNT_MESSAGE_SUSPECTS (String)
- SUSPECT_DUPLICATE_STATUS (String)
- CONTACT_METHOD_TYPE (String)

Variable Details

External Name: FAMILY_NAME

Variable Name: FAMILY_NAME

External Description: [Click Edit to add or edit text.](#) ([Edit](#))

Default Value: Last name

In-Progress Tasks Use New Values: ☐

Variable Type: [Sy...ta](#) [Select...](#) [New ...](#)

Figure 8-16 Changing the name variable

With this change, all the labels that are bound to this EPV variable automatically show the modified label text (Figure 8-17).

Master Data Management		
Manage Suspected Duplicates		
Resolve	1 of 3 Suspects	Previous Next
	Source Party Id 2000000000000001	Suspected Duplicate Party Id 2000000000000004
Given name	Annabelle	Annabelle
Last name	Kutcher	Kutcher
Match score		63 - 0 (G,G1Name,SName,DOB,Addr,SSN)
Match category		A1
Suspect duplicate status		Parties are Suspect Duplicates

Figure 8-17 Modified label

Another use case where use of EPVs are helpful is when using attribute metadata as an attribute label. In many scenarios, the metadata must be mapped to more human readable text that needs to be displayed as a label.

EPVs in MDM Policy Enforcement

Consider the MDM Policy Enforcement (PE) sample. In this process, EPVs are used to map the metadata to the display text that is used as a label. InfoSphere MDM Server Standard Edition provides data types that can be readily used to construct the member models. (A detailed description about the MDM element types and creation of member models is outside the scope of this book.)

When the member record is retrieved from InfoSphere MDM Server Standard Edition, the member record is returned with the metadata. Typically, a member record contains segments, and each segment contains attributes. Member records can have multiple segments at the same time. For example, LGLNAME (Legal name) and MAIDENNAME (Maiden name) are two segments that are of type MEMNAME in the member record. Further, the MEMNAME type has atomic attributes. For example, “onmFirst” corresponds to “First name.” Even though all the metadata is available with the member record, for human readability “onmFirst” is not suitable. To overcome this issue, the MDM Policy Enforcement process implementation uses EPVs to provide a mapping of segment name and attributes name to human readable display labels.

The EVPs MDSegmentNames and SegmentNameConfiguration define the mappings for the segment metadata to the display labels. The EPV MDSegmentNames defines the segment metadata that is used within the samples (Figure 8-18).

The screenshot shows a web application window titled "MDSegmentNames". The main content area is divided into several sections:

- Exposed Process Value**: A header section with a user icon in the top right.
- Common**: Contains fields for "Name" (MDSegmentNames), "Modified" (admin (Aug 1, 2012 7:37:28 PM)), and "Documentation" (a text area with a placeholder "Click [Edit](#) to add or edit text." and an "(Edit)" link).
- Details**: Contains fields for "Feedback E-mail Contact" (empty), "External Description" (a text area with a placeholder "Click [Edit](#) to add or edit text." and an "(Edit)" link), and "Default Value" (LGLNAME).
- Exposed Process Value Variables**: A list of variables with "Add" and "Remove" buttons. The list includes: LGLNAME (String), GENDER (String), MARSTAT (String), BUSADDR (String), SSN (String), BIRTHDT (String), HOMEADDR (String), and MAIDENNAME (String).
- Variable Details**: Contains fields for "External Name" (LGLNAME), "Variable Name" (LGLNAME), "External Description" (a text area with a placeholder "Click [Edit](#) to add or edit text." and an "(Edit)" link), "Default Value" (LGLNAME), "In-Progress Tasks Use New Values" (checkbox), and "Variable Type" (a dropdown menu with "Sy...ta" selected, and "Select..." and "New ..." buttons).
- Exposing**: A section at the bottom of the window.

Figure 8-18 Defining mappings for segment metadata

The EPV SegmentNameConfiguration defines the display names of the segments that are used within the samples. The display name of LGNAME segment is LEGAL NAME (Figure 8-19).

The screenshot shows a window titled "SegmentNameConfiguration" with a standard Windows-style title bar. The window is divided into several sections:

- Exposed Process Value**: A header section at the top right.
- Common**: A section on the left containing:
 - Name**: SegmentNameConfiguration
 - Modified**: admin (Aug 1, 2012 7:37:26 PM)
 - Documentation**: A text area with the placeholder "Click [Edit](#) to add or edit text." and an [\(Edit\)](#) link below it.
- Exposed Process Value Variables**: A section below Common containing a list of variables:
 - SSN (String)
 - BUSADDR (String)
 - GENDER (String)
 - MAIDENNAME (String)
 - HOMEADDR (String)
 - MARSTAT (String)
 - LGLNAME (String)** (highlighted)
 - BIRTHDT (String)There are "Add" and "Remove" buttons to the right of the list.
- Details**: A section on the right containing:
 - Feedback E-mail Contact**: An empty text field.
 - External Description**: A text area with the placeholder "Click [Edit](#) to add or edit text." and an [\(Edit\)](#) link below it.
- Variable Details**: A section below Details containing:
 - External Name**: LGLNAME
 - Variable Name**: LGLNAME
 - External Description**: A text area with the placeholder "Click [Edit](#) to add or edit text." and an [\(Edit\)](#) link below it.
 - Default Value**: LEGAL NAME
 - In-Progress Tasks Use New Values**: An unchecked checkbox.
 - Variable Type**: A dropdown menu showing "Sy...ta" with "Select..." and "New ..." buttons.
- Exposing**: A section at the bottom left, currently collapsed.

Figure 8-19 Defining the display names

Similarly, the EVPs MDSAttributeNames and AttributeNameConfiguration define the mappings for the attribute metadata to the display labels. The EPV MDSAttributeNames defines the attribute metadata that is used in the samples, as shown in Figure 8-20.

The screenshot shows a web-based configuration interface for 'MDSAttributeNames'. The window has a title bar with navigation icons and a close button. The main content area is divided into several sections:

- Exposed Process Value**: A header section with a user icon.
- Common**: Contains fields for 'Name' (MDSAttributeNames), 'Modified' (admin (Aug 1, 2012 7:37:27 PM)), and 'Documentation' (a text area with a placeholder 'Click [Edit](#) to add or edit text.').
- Details**: Contains fields for 'Feedback E-mail Contact' (empty), 'External Description' (a text area with a placeholder 'Click [Edit](#) to add or edit text.'), and an '(Edit)' link.
- Exposed Process Value Variables**: A list of variables with 'Add' and 'Remove' buttons. The list includes: IdNumber (String), attrVal (String), state (String), country (String), city (String), zipCode (String), **onmFirst (String)** (selected), stLine1 (String), and onmLast (String).
- Variable Details**: Contains fields for 'External Name' (onmFirst), 'Variable Name' (onmFirst), 'External Description' (a text area with a placeholder 'Click [Edit](#) to add or edit text.'), 'Default Value' (onmFirst), 'In-Progress Tasks Use New Values' (checkbox), and 'Variable Type' (a dropdown menu with options 'Sy...ta', 'Select...', and 'New ...').
- Exposing**: A section at the bottom of the interface.

Figure 8-20 Define the attribute metadata

The EPV AttributeNameConfiguration defines the display names of the segments that are used in the samples. The display name of the “onmFirst” attribute is “First Name” as shown in Figure 8-21.

The screenshot shows the 'AttributeNameConfiguration' window. It has several sections:

- Common:** Name: AttributeNameConfiguration, Modified: admin (Aug 1, 2012 7:37:27 PM), Documentation: Click [Edit](#) to add or edit text.
- Details:** Feedback E-mail Contact: (empty), External Description: Click [Edit](#) to add or edit text.
- Exposed Process Value Variables:** A list of variables including attrValGEN (String), onmFirst (String), attrValMARSTAT (String), attrValBIRTHDT (String), zipCode (String), country (String), idNumber (String), stLine1 (String), onmLast (String), and state (String). There are 'Add' and 'Remove' buttons.
- Variable Details:** External Name: onmFirst, Variable Name: onmFirst, External Description: Click [Edit](#) to add or edit text, Default Value: First Name, In-Progress Tasks Use New Values: (checkbox), Variable Type: Sy...ta, Select..., New ...
- Exposing:** Exposed to: <none>, Select..., New ...

Figure 8-21 Define segment display names

These EPVs are used in the script implementation. Open the Retrieve Entity Details service implementation. In the script node, Build Generic Object, the EPVs are used to construct the object to display on the user interface (Example 8-4).

Example 8-4 Construct object to display in the user interface

```
//Add Name Details
if( tw.local.members[j].memName != null ){
    for( var i = 0 ; i < tw.local.members[j].memName.listLength ; i++ ){
        var nameSegement = new tw.object.MemberObject();
        nameSegement.attributes = new tw.object.Map();

        //Add LegalName
        if(tw.local.members[j].memName[i].attrCode ==
String(tw.epv.MDSSegmentNames.LGLNAME)){
```

```

nameSegement.attributes.put(String(tw.epv.MDSAttributeNames.onmFirst),
tw.local.members[j].memName[i].onmFirst);
        nameSegement.attributes.put(String(tw.epv.MDSAttributeNames.onmLast),
tw.local.members[j].memName[i].onmLast);

genericMemberData.segments.put(tw.local.members[j].memName[i].attrCode,
nameSegement);
        //Construct map for DisplayName of each segment and attribute

genericMemberData.segmentNameConfiguration.put(tw.local.members[j].memName[i].attrCo
de,String(tw.epv.SegmentNameConfiguration.LGLNAME));

genericMemberData.attributeNameConfiguration.put("onmFirstLGLNAME",String(tw.epv.Attr
ibuteNameConfiguration.onmFirst));

genericMemberData.attributeNameConfiguration.put("onmLastLGLNAME",String(tw.epv.Attr
ibuteNameConfiguration.onmLast));
        }
        else
        //Add Maiden Name
        if(tw.local.members[j].memName[i].attrCode ==
String(tw.epv.MDSSegmentNames.MAIDENNAME)){

nameSegement.attributes.put(String(tw.epv.MDSAttributeNames.onmFirst),
tw.local.members[j].memName[i].onmFirst);
        nameSegement.attributes.put(String(tw.epv.MDSAttributeNames.onmLast),
tw.local.members[j].memName[i].onmLast);

genericMemberData.segments.put(tw.local.members[j].memName[i].attrCode,
nameSegement);
        //Construct map for DisplayName of each segment and attribute

genericMemberData.segmentNameConfiguration.put(tw.local.members[j].memName[i].attrCo
de, String(tw.epv.SegmentNameConfiguration.MAIDENNAME));

genericMemberData.attributeNameConfiguration.put("onmFirstMAIDENNAME",String(tw.epv.
AttributeConfiguration.onmFirst));

genericMemberData.attributeNameConfiguration.put("onmLastMAIDENNAME",String(tw.epv.A
ttributeNameConfiguration.onmLast));

        }

```

```
    }//for  
  }//if
```

Exposed process values can be used as a mechanism to define the display labels, which in turn, you can use to modify easily without requiring any extensive code change.

8.6.3 Adding and changing a policy

Enterprises that implement master data solutions establish data governance policies. A critical aspect of data governance is to ensure that the data that gets into the system adheres to those policies. Data governance must also provide ways to detect any noncompliance of the established policies so that it can be remediated. Enterprises implement these policies in multiple ways depending on the complexity of the policies. One widely used implementation of governance policies is through rules.

IBM BPM comes with a BAL rule component to create rules. This component provides a rule editor that rule designers can use to author business rules by using natural language technology. Using natural language instead of JavaScript to author rules means that no programming expertise is required to create business rules. Also, the rules are easier for people to read and understand.

By using Process Designer, non-developers can express business logic by using BAL in business rules. In most cases, business rules can be fully developed and implemented by using Process Designer. However, in some situations, the business logic might reach levels of complexity that are not supported within IBM Business Process Manager. In this case, the business logic can be exported without modifications to IBM WebSphere Operational Decision Management JRules Rule Studio. The export procedure produces a complete business rules project suitable for continued work within JRules. After the rules are exported, they are imported into Rule Studio, and the rules are deployed on a Rule Execution Server. The business process in Process Designer can use the resulting rule application by using a remote decision service that is provided by JRules.

For more information about BAL rules, search for *Adding a BAL Rule component to a service* in the IBM Business Process Manager V8.0 Information Center at:

<http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/index.jsp>

In the samples that IBM bundles with the InfoSphere MDM Server, the BAL rules are user-defined policies through data validation.

BAL rules in MDM Policy Enforcement

The PE sample showcases the use of BAL language for policy implementation. It also used an XML file for textual description of the policies that are wrapped in a Java adapter to retrieve the content of the XML.

To enhance the sample, by defining a new rule for a policy, you can create a Decision service by using Process Designer. For information about creating a Decision service, search for *Adding a BAL Rule component to a service* in the IBM Business Process Manager V8.0 Information Center at:

<http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/index.jsp>

In the diagram of the new Decision service that you create, click **BAL Rule** in the component palette and drag the **BAL Rule component** icon from the palette to the service diagram. Link any other service implementation in BPM. In the Decision service, you can also map the input and output variables, which can be accessed within the BAL rule language. You can use the pre-execution and post-execution assignments to extract the data element for processing by the BAL rule and construct a data object that represents the result of the rule execution.

The Decision service can then be nested in a General System service and used in the business process. In this sample, the text descriptions for the policy are provided through a simple XML script (Example 8-5).

Example 8-5 Policy descriptions

```
<?xml version="1.0" encoding="UTF-8"?>
<p:PolicyValidations
  xmlns:p="http://www.ibm.com/mdm/datagovernance/PolicyValidation"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ibm.com/mdm/datagovernance/PolicyValidation PolicyValidation.xsd">
  <policyValidation>
    <policyID>1</policyID>
    <policyShortDesc>SSN value cannot be Null</policyShortDesc>
    <policyLongDesc>SSN value for the entity record cannot be
null</policyLongDesc>
    <policyRemediationApproach>Remediation
approach</policyRemediationApproach>
  </policyValidation>
```

You can update this XML and the file in the business process application library. Ensure that the policyID is a unique string. The sample provides data object definitions that can hold the details that are provided through XML.

8.6.4 Adding an activity

Activities are the individual business tasks within a process that make up the larger business goal. When you define the business process by using Process Designer, these activities are typically a service implementation. The activities are a set of service implementation types that are supported by BPM. The type of service that you create depends upon the requirements of the activity. For example, if an activity requires integration with an external system, such as a database, you can create an Integration service. If an activity requires that call center personnel enter data about customer requests, you can create a Human service with a coach.

You can choose from the following activities:

- | | |
|-------------------------------------|---|
| Decision service | Use when you want a condition to determine the implementation that is started. For example, when a condition evaluates to true, IBM BPM implements the JavaScript expression that you provide. Decision services cannot include Java or Web service integrations directly. You can call a Decision service from any other type of service, and a Decision service can call other nested services. |
| Human service | Use to create an interactive service. A Human service is the only type of service that can contain coaches and postpones. Human services generate tasks in IBM Process Portal. |
| Ajax service | Use when you want to include a control in a coach to implement dynamic data selection such as automatically populating drop-down lists and completing edit boxes. An Ajax service can pull data dynamically from a connected data source, such as a database. You cannot call an Ajax service from other types of services, but an Ajax service can call other nested services. |
| Integration service | Use to integrate with an external system. An Integration service is the only type of service that can contain a Java or Web service integration. You can call an Integration service from any other type of service, and an Integration service can call other nested services. |
| Advanced Integration service | Use to integrate with a service that is created in Integration Designer. |

IBM Case Manager Integration service

Use to integrate with an IBM Case Manager server.

General System service

Use to coordinate other nested services or to manipulate variable data. For example, to implement data transformations or generate HTML for a coach, you can use a General System service. General System services cannot include Java or Web service integrations directly. You can call a General System service from any other type of service, and a General System service can call other nested services.

Process Designer provides a context-sensitive component palette from which you can drag the components to construct the individual activities and the process definition.

For information about how to define a complete process, search for *modeling processes* in the IBM Business Process Manager V8.0 Information Center at:

<http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/index.jsp>

8.6.5 Adding activity UIs

A business process consists of a set of tasks that are implemented through the various service implementations. A data steward or any user interacts with the business process through well-defined user interfaces. These interactions are implemented as Human services. Human services represent the actions that these users perform within the process. In BPM terminology, user interfaces are referred to as *coaches*.

The coaches in IBM Business Process Manager V8.0 are different in construction than the coaches in previous versions of IBM BPM. In versions before V8.0, the coaches of the previous versions of IBM BPM are referred to as *heritage coaches*. The primary difference between coaches and the heritage coaches of previous versions is that the new coaches consist of one or more *coach views*. Coach views are reusable collections of user interfaces that are frequently bound to a data type.

Process Designer provides a context-sensitive component palette from where you can drag the components to construct the coaches. Also, BPM supports complete customization of coaches, which means a custom hand-coded user interface can seamlessly be used with human services implementation.

For more information about defining coaches and implementing human services, search on *creating user interfaces* in the IBM Business Process Manager V8.0 Information Center at:

<http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/index.jsp>

For example, to add a review and approval step to the Policy Enforcement and Remediation sample, drag the required controls from the component palette and wire them to the samples. Assuming you want to reuse the remediation UI, you can copy and then paste the existing coach controls.

8.7 Reporting for success

Enterprises that implement business processes need to monitor the processes from a performance and statistical perspective to further improve and optimize the process. IBM Business Process Manager provides ways to collect and use process performance information. The process applications can be designed so that they can be tracked by the monitoring framework of BPM.

In IBM Business Process Manager V8, the reporting function was deprecated. However, this function is compatible with earlier versions. The reporting function is not enabled by default.

To enable reporting, in Process Designer, click **File** → **Preferences** → **IBM BPM** → **Capabilities**. Then, enable the capabilities that are compatible with earlier versions. By using reports, you can analyze business data that is specific to your processes. You can specify the variables to track and define the report to query your tracked data. Users can view the resulting report scoreboards from the Dashboards page in Process Portal. However, in BPM V8, you can use more mechanisms for monitoring, tracking, and reporting purposes.

By using the Performance Data Warehouse, you can create customized reports by using third-party reporting tools in IBM Business Process Manager. To use this feature, you must identify the data elements that are required for tracking in the process. You send this tracking definition to the Performance Data Warehouse.

To track data in a business process definition (BPD), you can use the facilities of autotracking, tracking groups, or both. Autotracking automatically captures data from tracking points at the entry and exit of each item in a BPD (for example, services, activities, and gateways). Tracking groups provide more control over tracked data. You use them to track a selected group of process variables across multiple BPDs or process applications and to store tracking points for a timing interval. You can take advantage of both tracking methods in a single BPD. Use

discretion when you model process definitions because the number of events that are captured for reporting impacts the process performance.

For more information about the configurations and steps for process monitoring, search for *enabling processes for tracking and reporting* in the IBM Business Process Manager V8.0 Information Center at:

<http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/index.jsp>



A

MDM Advanced Edition configuration script

Use the script in Example A-1 to achieve the required configuration in IBM InfoSphere Master Data Management (MDM) Advanced Edition for messaging integration.

Example A-1 MDM Advanced Edition configuration script for messaging integration

```
#####  
# Content of file MDMBPMSSetupApp.py  
#####  
import sys  
#####  
#  
# Functions:  
#  
# getPort  
# createSIBAlias  
# createForeignBus  
# createForeignBusLink  
# createSIBJMSQueue  
# createSIBJMSConnectionFactory  
# createDefaultJMSProvider
```

```

def getPort (portName, nodeName, serverName):
    lineSeparator = java.lang.System.getProperty('line.separator')
    node=AdminConfig.getid("/Node:"+nodeName+"/")
    serverEntries = []
    serverEntries=AdminConfig.list("ServerEntry",node).split(lineSeparator)
    for serverEntry in serverEntries:
        sName = AdminConfig.showAttribute(serverEntry,"serverName")
        if sName == serverName:
            specialEndPoints = []
            temp = AdminConfig.showAttribute(serverEntry,"specialEndpoints")
            specialEndPoints =
AdminConfig.showAttribute(serverEntry,"specialEndpoints").split(" ")
            for specialEndPoint in specialEndPoints:
                specialEndPoint = specialEndPoint.replace("]",",")
                endPointNm = []
                endPointNm = AdminConfig.showAttribute(specialEndPoint,"endPointName")
                if endPointNm == portName:
                    ePoint = AdminConfig.showAttribute(specialEndPoint,"endPoint")
                    vhPort = AdminConfig.showAttribute(ePoint,"port")
                    #print "\nGet the Port - "+portName+" = "+vhPort
                    break
    return vhPort

def createSIBAlias
(clusterName,nodeName,serverName,SIBAliasName,SIBusName,targetBusName,targetDestinati
onName ):

    #-----
    # up globals
    #-----

    global AdminConfig
    global AdminTask

    #-----
    # Create SIB queue
    #-----

    print ""
    print "-----"
    print "          Create SIB Alias          "
    print "-----"
    print " SIB Alias:          "

```

```

print "    name                "+SIBAliasName+"                "
print " SI bus:                "
print "    name                "+SIBBusName+"                "
print "-----"
print ""
#lineSeparator = java.lang.System.getProperty('line.separator')
#arr=AdminConfig.list("SIBAlias").split(lineSeparator)
#for queue in arr :
#identifier = AdminConfig.showAttribute(queue,"identifier")
#if identifier == SIBAliasName:
#    print " The "+SIBAliasName+" SIB queue already exists"
#    return

#-----
# Create SIB queue
#-----

print " Creating a new SIB Alias named "+SIBAliasName+"..."
if clusterName != "None":
    params="[-bus "+SIBBusName+" -name "+SIBAliasName+" -type Alias -cluster
"+clusterName+" -targetBus "+targetBusName+" -targetName "+targetDestinationName+"]"
else:
    params="[-bus "+SIBBusName+" -name "+SIBAliasName+" -type Alias -node
"+nodeName+" -server "+serverName+" -targetBus "+targetBusName+" -targetName
"+targetDestinationName+"]"
    try:
        AdminTask.createSIBDestination(params)
    except:
        print " The "+SIBAliasName+" SIB Alias already exists"

def createForeignBus (clusterName,nodeName,serverName,SIBBusName,foreignBusName):

#-----
# up globals
#-----

global AdminConfig
global AdminTask

#-----
# Create Foreign Bus
#-----

```

```

print ""
print "-----"
print "                Create Foreign Bus                "
print "-----"
print " Foreign Bus:                                     "
print "     name                "+foreignBusName+"                "
print " SI bus:                                     "
print "     name                "+SIBusName+"                "
print "-----"
print ""

#-----
# Create Foreign Bus
#-----

print " Creating a new Foreign Bus named "+foreignBusName+"..."
if clusterName != "None":
    params="[-bus "+SIBusName+" -name "+foreignBusName+" -routingType Direct -type
SIBus ]"
else:
    params="[-bus "+SIBusName+" -name "+foreignBusName+" -routingType Direct -type
SIBus]"
try:
    AdminTask.createSIBForeignBus(params)
except:
    print " The "+foreignBusName+" foreign bus already exists"

def createForeignBusLink
(clusterName,nodeName,serverName,SIBusName,foreignBusName,foreignBusMessagingEngineNa
me,SIBLinkName,endpoint):

#-----
# up globals
#-----

global AdminConfig
global AdminTask

#-----
# Create Foreign Bus
#-----

```



```

print ""
print "-----"
print "          Create Foreign Bus          "
print "-----"
print " Foreign Bus:          "
print "   name                "+foreignBusName+"          "
print " SI bus:                "
print "   name                "+SIBusName+"          "
print "-----"
print ""

#-----
# Create Foreign Bus
#-----

if clusterName != "None":
    localMessagingEngine = clusterName+".000-"+SIBusName
else:
    localMessagingEngine = nodeName+"."+serverName+"-"+SIBusName

print " Creating a new SIB Link named "+SIBLinkName+"..."

print " Foreign messaging engine name is "+foreignBusMessagingEngineName

params="[-bus "+SIBusName+" -messagingEngine "+localMessagingEngine+" -name
"+SIBLinkName+" -foreignBusName "+foreignBusName+" -remoteMessagingEngineName
"+foreignBusMessagingEngineName+" -bootstrapEndpoints "+endpoint+"]"

try:
    AdminTask.createSIBLink(params)
except:
    print " The "+SIBLinkName+" SIB Link already exists"

def createSIBJMSQueue
(clusterName,nodeName,serverName,jmsQName,jmsQJNDI,jmsQDesc,SIBQName,SIBusName):

#-----
# up globals
#-----

global AdminConfig

```

```

global AdminTask

#-----
# Create SIB JMS queue
#-----

print ""
print "-----"
print "          Create SIB JMS queue          "
print "-----"
print "    SIB          "+SIBusName+"          "
print " SIB JMS queue:          "
print "    name          "+jmsQName+"          "
print "    JNDI          "+jmsQJNDI+"          "
print "    description    "+jmsQDesc+"          "
print "    SIB queue      "+SIBQName+"          "
print "-----"
print ""

#-----
# Create SIB JMS queue that references a SIB queue
#-----
if clusterName != "None":
    scope = AdminConfig.getid("/ServerCluster:"+clusterName+"/")
else:
    scope = AdminConfig.getid("/Node:"+nodeName+"/Server:"+serverName+"/")

#for queue in AdminTask.listSIBJMSQueues(scope):
#name = AdminConfig.showAttribute(queue,"name")
#if name == jmsQName:
#    print " The "+jmsQName+" SIB JMS queue already exists"
#    return

print " Creating a new SIB JMS queue named "+jmsQName+"..."
params = "[-busName "+SIBusName+" -name \" "+jmsQName+"\" -jndiName "+jmsQJNDI+"
-description \" "+jmsQDesc+"\" -queueName "+SIBQName+"]"
try:
    AdminTask.createSIBJMSQueue(scope,params)
except:
    print " The "+jmsQName+" SIB JMS queue already exists"

def createSIBJMSConnectionFactory
(clusterName,nodeName,serverName,jmsCFName,jmsCFJNDI,jmsCFDesc,jmsCFType,authAlias,SIBusName,endpoint):

```

```

#-----
# up globals
#-----

global AdminConfig
global AdminTask

#-----
# Create SIB JMS connection factory
#-----

print ""
print "-----"
print "          Create SIB JMS connection factory          "
print "-----"
print " Connection factory:                                     "
print "     name              "+jmsCFName+"                  "
print "     JNDI              "+jmsCFJNDI+"                  "
print "     description       "+jmsCFDesc+"                  "
print "     type              "+jmsCFTYPE+"                  "
print "     auth alias        "+authAlias+"                  "
print " SI bus:              "
print "     name              "+SIBusName+"                  "
print "-----"
print ""
if clusterName != "None":
    jmsCF = AdminConfig.getid
    ("/ServerCluster:"+clusterName+"/J2CResourceAdapter:SIB JMS Resource
Adapter/J2CConnectionFactory:"+jmsCFName)
else:
    jmsCF = AdminConfig.getid ("/Node:"+nodeName+"/J2CResourceAdapter:SIB JMS
Resource Adapter/J2CConnectionFactory:"+jmsCFName)

print ""
print " JMS Connection Factory ID is "+jmsCF
print ""
if len(jmsCF) != 0:
    print " The "+jmsCFName+" JMS connection factory already exists"
    return

#-----
# Create the SIB JMS connection factory
#-----
if clusterName != "None":

```

```

    scope = AdminConfig.getid("/ServerCluster:"+clusterName+"/")
else:
    scope = AdminConfig.getid("/Node:"+nodeName+"/Server:"+serverName+"/")

print "  Creating a new SIB JMS connection factory named "+jmsCFName+"..."
print scope
params = []
params.append(["-name",jmsCFName])
params.append(["-jndiName",jmsCFJNDI])
params.append(["-busName",SIBusName])
if jmsCFDesc != "":
    arg = ["-description",jmsCFDesc]
    params.append(arg);

endArg = ""
if endpoint != "":
    endArg = " -providerEndpoints " + endpoint

if jmsCFType != "":
    arg = ["-type",jmsCFType]
    params = ["-name \""+jmsCFName+"\" -jndiName "+jmsCFJNDI+" -type "+jmsCFType+"
-busName "+SIBusName+ endArg+"]"
    elif authAlias != "":
        arg = ["-authDataAlias",authAlias]
        params = ["-name \""+jmsCFName+"\" -jndiName "+jmsCFJNDI+" -busName
"+SIBusName+" -authDataAlias "+authAlias+ endArg+"]"
    else:
        params = ["-name \""+jmsCFName+"\" -jndiName "+jmsCFJNDI+" -busName
"+SIBusName+ endArg+"]"
    print ""
    AdminTask.createSIBJMSConnectionFactory(scope,params)

def createDefaultJMSProvider
(clusterName,nodeName,serverName,SIBusName,targetBusName,targetDestinationName,target
MessagingEngineName,endpointPort):

    print ""
    print "Creating Default Messaging Provider"

    print ""
    print "Setting endpoint port"

```

```

endpoint = ""
if clusterName != "None":
    clusterId = AdminConfig.getId("/ServerCluster:"+clusterName+"/")

    lineSeparator = java.lang.System.getProperty('line.separator')
    members = AdminConfig.list("ClusterMember",clusterId).split(lineSeparator)

    for id in members:
        srv = AdminConfig.showAttribute(id,"memberName")
        node = AdminConfig.showAttribute(id,"nodeName")
        nodeId = AdminConfig.getId("/Node:"+node+"/")
        host = AdminConfig.showAttribute(nodeId,"hostName")
        port = getPort("SIB_ENDPOINT_ADDRESS", node, srv)

        endpoint1 = host+": "+port+":BootstrapBasicMessaging"
        if endpoint == "":
            endpoint = endpoint1
        else:
            endpoint = endpoint + ","+endpoint1
    else:
        endpointPort = getPort("SIB_ENDPOINT_ADDRESS", nodeName, serverName)

        print "endpoint port : "+endpointPort
        if endpointPort != "7276":
            endpoint = "localhost:" + endpointPort + ":BootstrapBasicMessaging"

jmsCFName = "MDMBPMQueueConnectionFactory"
jmsCFJNDI= "notification/MDMBPMQueueConnectionFactory"
jmsCFDesc= "MDM BPM Queue Connection Factory"
jmsCFType = "Queue"
secAuthAlias = ""
createSIBJMSConnectionFactory
(clusterName,nodeName,serverName,jmsCFName,jmsCFJNDI,jmsCFDesc,jmsCFType,secAuthAlias
,SIBusName,endpoint)

#-----
# Create JMS Queue
#-----

jmsQName= "MDMBPMQueue"
jmsQJNDI = "notification/MDMBPMQueue"
jmsQDesc= "MDM BPM Queue"
AliasQName = "MBMBPMAlias"

```

```

        createSIBJMSQueue(clusterName,nodeName,serverName,jmsQName,jmsQJNDI,jmsQDesc
,AliasQName,SIBusName)
        createSIBAlias(clusterName,nodeName,serverName,AliasQName ,SIBusName,
targetBusName,targetDestinationName)

#-----
# Create Foreign Bus
#-----
endpoint = "localhost:" + endPointPort + ":BootstrapBasicMessaging"
SIBLinkName="MDM_BPM_LINK"
createForeignBus (clusterName,nodeName,serverName,SIBusName,targetBusName)
createForeignBusLink
(clusterName,nodeName,serverName,SIBusName,targetBusName,targetMessagingEngineName,SIBLinkName,endpoint)

```

```

#####
# Prints the usage message
#####
def printUsage ():
    print ""
    print "Usage:"
    print ""
    print "    wsadmin -f MDMBPMSetupApp.py"
    print "    ClusterName"
    print "    NodeName"
    print "    ServerName"
    print "    SIBusName"
    print "    targetBusName"
    print "    targetDestinationName"
    print "    targetMessagingEngineName"
    print "    targetEndPointPort"

    sys.exit(1)

```

```

#####
#                                     Main                                     #
#####

```

```

print "Number of arguments:"
print len(sys.argv)
if len(sys.argv) != 8:
    printUsage()
    sys.exit()

ClusterName = sys.argv[0];
NodeName = sys.argv[1];
ServerName = sys.argv[2];
SIBusName= sys.argv[3];
targetBusName = sys.argv[4];
targetDestinationName = sys.argv[5];
targetMessagingEngineName=sys.argv[6];
targetEndPointPort=sys.argv[7];

print ""
print "Setting up application server for MDM BPM Integration ..."
print " Cluster name           : "+ClusterName
print " Node name                : "+NodeName
print " Server name                : "+ServerName
print " SI Bus name                : "+SIBusName
print " Target Bus name            : "+targetBusName
print " Target Destination Name     : "+targetDestinationName
print " Target Messaging Engine Name : "+targetMessagingEngineName
print " Target End Point Port       : "+targetEndPointPort
print ""

#~~~~~#
#                                     #
#                               WAS or MQ JMS Queue Connection Factories      #
#-----#

createDefaultJMSProvider(ClusterName,NodeName,ServerName,SIBusName,targetBusName,targetDestinationName,targetMessagingEngineName,targetEndPointPort);

#~~~~~#
# Save server configuration          #
#-----#
print ""
if ClusterName != "None":
    print " Saving configuration changes for cluster '"+ClusterName+"'...."

```

```
# ~~~~~#
# Done
# -----#
print ""
print "  Done setting up environment."
print ""
```




B

BPM Express Edition for messaging integration

Use the script in Example B-1 to achieve the required configuration on IBM Business Process Manager Express Edition for messaging integration with IBM InfoSphere Master Data Management (MDM) Advanced Edition.

Example B-1 BPM Express Edition configuration for messaging integration with MDM

```
#####  
# Content of file BPMSetupApp.py  
#####  
import sys  
#####  
#  
# Functions:  
#  
#  
# createJAASAuthData  
# setupDeployServer  
# createJDBCProvider  
# createDatasource  
# setOracleDatasourceCustomProperties  
# getJMSProvider  
# createMQQueueConnectionFactory
```

```

# createMQQueue
# createMQTopicConnectionFactory
# createMQTopic
# createWASQueueConnectionFactory
# createWASQueue
# createWASTopicConnectionFactory
# createWASTopic
# createMessageListenerPort
# setWASVariablesForMQ
# createAlias
# printUsage
#

def createForeignBus
(clusterName,nodeName,serverName,SIBusName,foreignBusName,userId):

    #-----
    # up globals
    #-----

    global AdminConfig
    global AdminTask

    #-----
    # Create Foreign Bus
    #-----

    print ""
    print "-----"
    print "          Create Foreign Bus          "
    print "-----"
    print " Foreign Bus:          "
    print "   name                "+foreignBusName+"          "
    print " SI bus:                "
    print "   name                "+SIBusName+"          "
    print "-----"
    print ""

    #-----
    # Create Foreign Bus
    #-----

    print "  Creating a new Foreign Bus Link named "+foreignBusName+"..."

```

```

        if clusterName != "None":
            params="[-bus "+SIBusName+" -name "+foreignBusName+" -routingType Direct -type
SIBus ]"
        else:
            params="[-bus "+SIBusName+" -name "+foreignBusName+" -routingType Direct -type
SIBus -inboundUserId userId]"
        try:
            AdminTask.createSIBForeignBus(params)
        except:
            print " The "+foreignBusName+" foreign bus already exists"

```

```

def createForeignBusLink
(clusterName,nodeName,serverName,SIBusName,foreignBusName,foreignBusMessagingEngineName,SIBLinkName,endpoint):

```

```

#-----
# up globals
#-----

global AdminConfig
global AdminTask

#-----
# Create Foreign Bus
#-----

print ""
print "-----"
print "          Create Foreign Bus          "
print "-----"
print " Foreign Bus:          "
print "   name              "+foreignBusName+"          "
print " SI bus:              "
print "   name              "+SIBusName+"          "
print "-----"
print ""

#-----
# Create Foreign Bus
#-----

```

```

    if clusterName != "None":
        localMessagingEngine = clusterName+".000-"+SIBusName
    else:
        localMessagingEngine = nodeName+"."+serverName+"-"+SIBusName

    print "  Creating a new SIB Link named "+SIBLinkName+"..."

    params="[-bus "+SIBusName+" -messagingEngine "+localMessagingEngine+" -name
"+SIBLinkName+" -foreignBusName "+foreignBusName+" -remoteMessagingEngineName
"+foreignBusMessagingEngineName+" -bootstrapEndpoints "+endpoint+"]"

    try:
        AdminTask.createSIBLink(params)
    except:
        print "  The "+SIBLinkName+" SIB Link already exists"

#####
# This method create WAS6 Default JMS Provider.
#####
def createDefaultJMSProvider
(clusterName,nodeName,serverName,SIBusName,targetBusName,foreignBusMessagingEngineNam
e,endPointPort,userId):

    #-----
    # Create Foreign Bus
    #-----
    endpoint = "localhost:" + endPointPort + ":BootstrapBasicMessaging"
    SIBLinkName="MDM_BPM_LINK"
    createForeignBus (clusterName,nodeName,serverName,SIBusName,targetBusName,userId)
    createForeignBusLink
(clusterName,nodeName,serverName,SIBusName,targetBusName,foreignBusMessagingEngineNam
e,SIBLinkName,endpoint)

#####
# Prints the usage message
#####
def printUsage ():
    print ""
    print "Usage:"
    print ""
    print "  wsadmin -f BPMSetupApp.py"
    print "      ClusterName"
    print "      NodeName"

```

```

print      "ServerName"
print      "SIBusName"
print      "ForeignBusName"
print      "ForeignBusMessagingEngineName"
print      "ForeignEndPointPort"
print      "AdminUserIdOfBPM"

sys.exit(1)

#####
#                               Main                               #
#####
print "Number of arguments:"
print len(sys.argv)
if len(sys.argv) != 8:
    printUsage()
    sys.exit()

ClusterName = sys.argv[0];
NodeName = sys.argv[1];
ServerName = sys.argv[2];
SIBusName= sys.argv[3];
targetBusName=sys.argv[4];
targetBusMessagingEngineName=sys.argv[5];
targetBusEndPointPort=sys.argv[6];
userId = sys.argv[7];

print ""
print "Setting up application server for MDM BPM Integration ..."
print " Cluster name      : "+ClusterName
print " Node name          : "+NodeName
print " Server name         : "+ServerName
print " SI Bus name         : "+SIBusName

```


Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *IBM InfoSphere Information Server Deployment Architectures*, SG24-8028
- ▶ *Creating a BPM Center of Excellence (COE)*, REDP-4898
- ▶ *Scaling BPM Adoption: From Project to Program with IBM Business Process Manager*, SG24-7973
- ▶ *Business Process Management with IBM Business Process Manager*, TIPS0938
- ▶ *IBM InfoSphere Information Server Installation and Configuration Guide*, REDP-4596
- ▶ *Metadata Management with IBM InfoSphere Information Server*, SG24-7939
- ▶ *Smarter Modeling of IBM InfoSphere Master Data Management Solutions*, SG24-7956

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ What's New in IBM InfoSphere Master Data Management V10.1
http://www.ibm.com/software/data/infosphere/mdm/whats_new.html
- ▶ IBM Business Process Manager V8.0 Information Center
<http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0mx/index.jsp>
- ▶ IBM InfoSphere MDM Information Center
<http://pic.dhe.ibm.com/infocenter/mdm/v10r0m0/index.jsp>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Aligning MDM and BPM for Master Data Governance, Stewardship, and Enterprise Processes

See how to enable trusted and accurate information for business processes

Understand how to combine MDM and BPM for enhanced value

Learn how to maximize the value of business solutions

An enterprise can gain differentiating value by aligning its master data management (MDM) and business process management (BPM) projects. This way, organizations can optimize their business performance through agile processes that empower decision makers with the trusted, single version of information.

Many companies deploy MDM strategies as assurances that enterprise master data can be trusted and used in the business processes. IBM InfoSphere Master Data Management creates trusted views of data assets and elevates the effectiveness of an organization's most important business processes and applications.

This IBM Redbooks publication provides an overview of MDM and BPM. It examines how you can align them to enable trusted and accurate information to be used by business processes to optimize business performance and bring more agility to data stewardship. It also provides beginning guidance on these patterns and where cross-training efforts might focus.

This book is written for MDM or BPM architects and MDM and BPM architects. By reading this book, MDM or BPM architects can understand how to scope joint projects or to provide reasonable estimates of the effort. BPM developers (or MDM developers with BPM training) can learn how to design and build MDM creation and consumption use cases by using the MDM Toolkit for BPM. They can also learn how to import data governance samples and extend them to enable collaborative stewardship of master data.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks