TIBCO™

WHITEPAPER

The Evolutionary Steps to Master Data Management

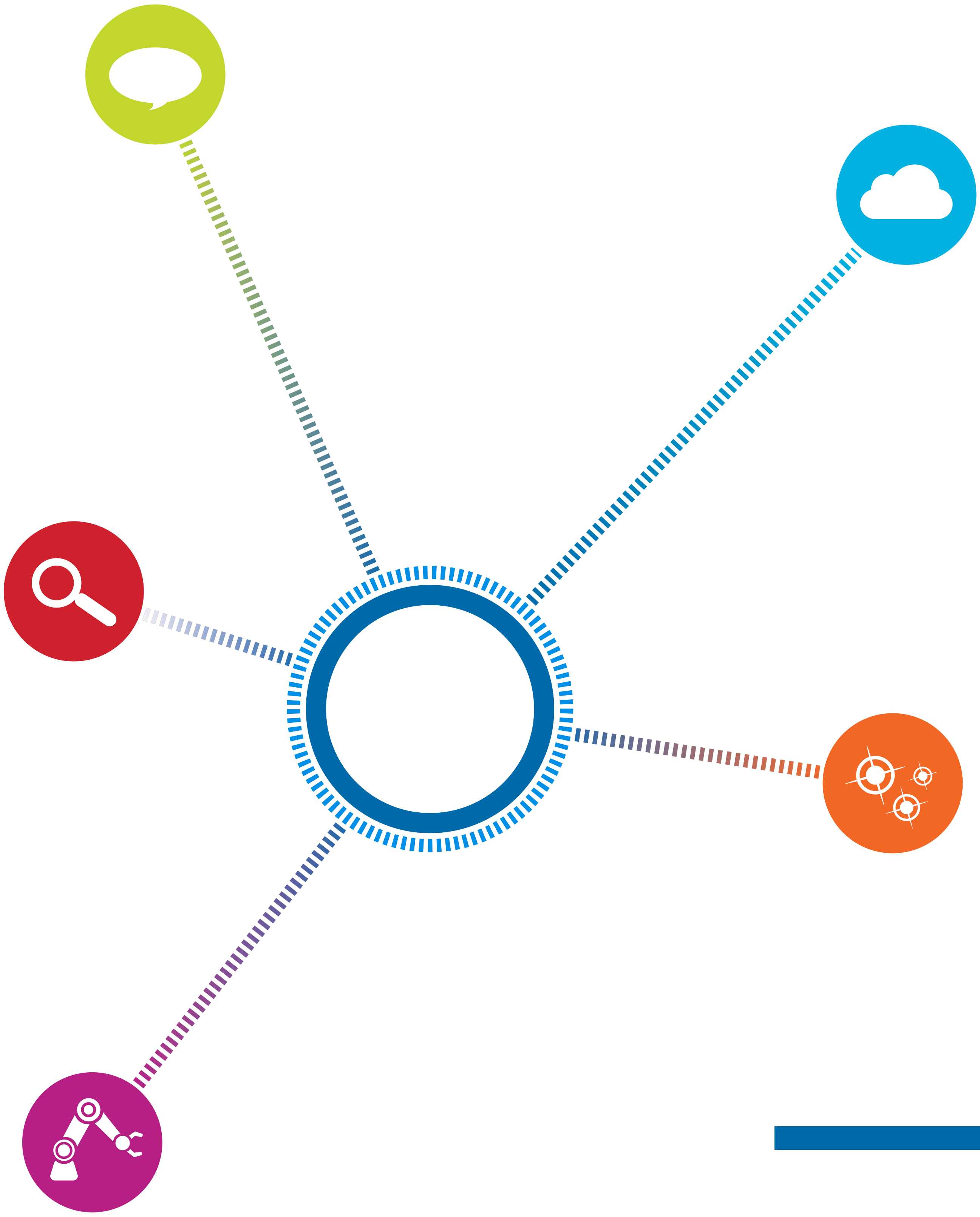


Table of Contents

The Evolutionary Steps to Master Data Management

Introduction

Master data management, or MDM, provides a single source of truth for your most critical data assets. An MDM project involves bringing together information that is isolated in specialized systems and repositories so it can be referenced, correlated, analysed, and consumed. Benefits include accurate customer information, faster information gathering, and actionable insight into operations.

However, every organization is different, and to deliver on MDM's promise, you'll need to implement according to your unique needs. While there's no sure fire way to implement that will work at every organization, this white paper offers one approach that allows IT and business users to adapt to changes in data and business requirements more effectively. It also allows data to be delivered as a service to the many, and not just a select few.

Step 1, Implement a Foundation

To implement MDM, you must establish a service-oriented architecture that provides an integration or service layer, which makes it easier to implement, migrate, and manage MDM data domains.

Step 2, Choose from Three Levels of Implementation

MDM can be implemented in multiple ways that build off of each other.

1. Registry implementation: Only reference IDs are stored within the MDM solution (TIBCO MDM enables other attributes to be stored if required).
2. Coexisting data source implementation: Data is mastered and updated in more than one place, including the MDM environment. The goal is to make the MDM system the go to place for the data domain, so data governance can be applied within the MDM system.
3. Data consolidation implementation: Data is mastered and governed within the MDM solution. Though this approach is ideal, external factors often prevent this implementation style from working across all data domains.

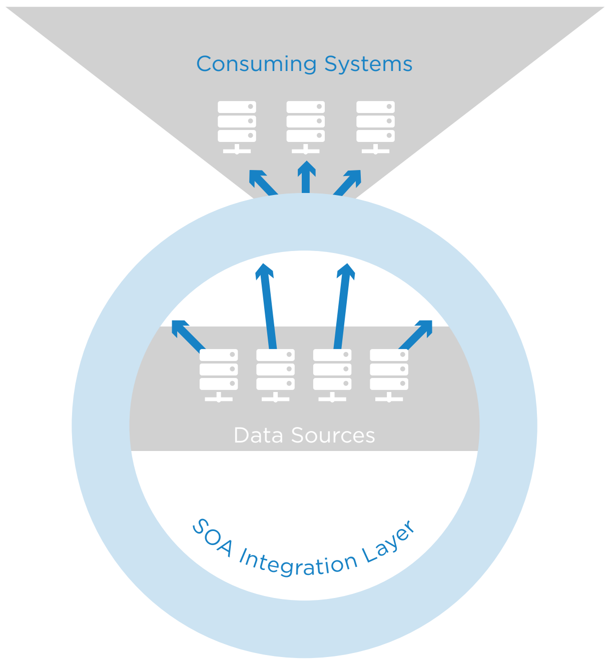
As we describe these implementation styles, assume that one data object will be migrated through each level of implementation. This may not be the case in reality, but it allows for a consistent description of the implementation styles required to achieve a fully MDM managed data domain.

Step 1: Implement a Foundational Service Layer

To build an adaptable framework that can handle future requirements, you'll need the following essentials:

- A real-time enterprise service bus: Communication between systems should ideally be performed via an ESB.
- Implementation of a service layer: A service layer enables and regulates access to data through well-defined APIs that provide common services such as find, read, write, and delete.
- Information-as-a-service capabilities: When data is provided as a service it can be accessed from a request/response interface, or via a subscription interface. Well-defined API's establish a set interface for commonly requested data objects or data hierarchies. These capabilities describe a service-oriented architecture (SOA).
- Batch support: You should be able to upload and export data as a batch file, but eventually you'll want to consider retiring batch processing from your MDM processes.

The proposed architecture defines an ESB as a backbone that will be used to enable real-time communication between all systems. Adding an integration platform, such as TIBCO ActiveMatrix BusinessWorks™, between the ESB and other components, enables data services to be defined and exposed as web services or publish/subscribe services on the ESB or over HTTP. This is a fundamental first step to using the data contained within various systems. The figure below shows the modified architecture.



Implementation of Service Oriented Architecture as the foundation layer

Specific data objects are encapsulated as a service and exposed via an ESB, a web service, or as a batch export/import. Other systems can then access these data objects via web service calls, ESB interface, or an export batch file, depending on the capabilities of the systems being integrated.

Because organizations generally grow organically, the implementation technology used to define the integration layer needs to have the ability to connect to a wide variety of data sources (SAP, Oracle, Saleforce, databases, and others) such as provided by TIBCO BusinessWorks. This integration layer allows various data objects to be presented as services and enables common actions such as find, create, update, and delete. Batch import and export capabilities can also be provided as a drop-in or securely managed file transfer capabilities.

Benefits of service layer implementation

- Enables access to data across the entire enterprise, including the ability to create, read, update, and search.
- Establishes data as enterprise data objects with consistent definitions. For example, "A person object contains first name and last name and both attributes are 60 characters long."
- Forces the identification of data sources across the enterprise. For example, "What is the source of this data? How frequently does it update? And which delivery formats are available for this data domain?"
- Allows you to implement data security at the point of integration, though it may not be flexible enough to cope with the demands of an evolving IT enterprise.
- Makes it possible to manage changes to the API though interface versioning, enabling an evolution of the data source without seriously impacting the target system.

Issues

- Is not yet true master data management because it only establishes common data objects and data access APIs.
- Data validation rules are difficult to implement within an integration layer.
- Data governance (processes, approval, and structure) is either not possible or difficult to manage due to different technologies.
- This should not be done in one go for all data objects. Each system and data object needs to be evaluated, along with the potential impact, based on dependencies and priorities.

Step 2: Choose a style

A) Add Master Data Management

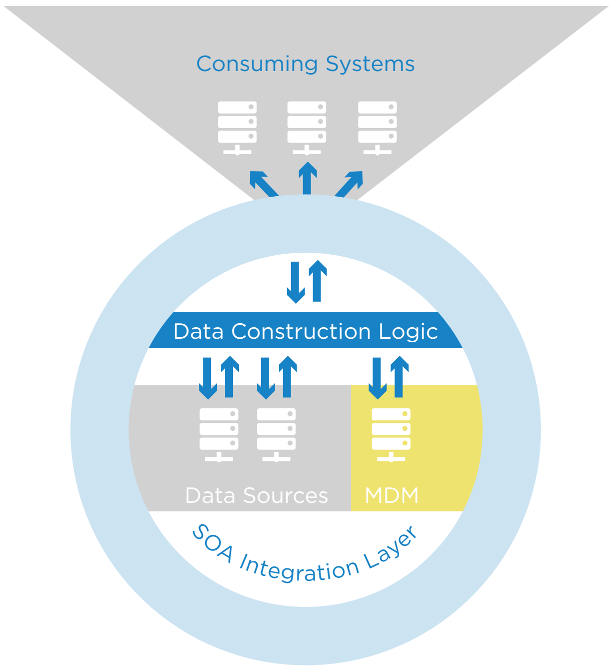
The simplest introduction of TIBCO® MDM may be as a registry implementation. This level of implementation enables existing systems and interfaces (technical and human) to remain unchanged. Direct access to data sources remains in place with no change to the way users interact with source systems.

The introduction of a MDM registry enables:

- Relationships between disparate data sources.
- Audits of changes to data that can be captured and reported on.
- Maintenance of structural integrity by defining an MDM data model that can be changed over time.
- Addition of transactional context to update events that enforces the structure of the data.

A minimum requirement for a registry style implementation is to only store reference IDs within the MDM solution. However, the MDM data model can extend beyond the structure of the data sources to reflect data groupings and relationships between systems not included among the source systems.

For example, a person data object could be built from two or more source systems (CRM and a web database). The entity "person" stored within MDM could have two external reference IDs, one from the online database and one from the CRM system. MDM holds the definition of the relationships between the two data objects and source IDs. Relationships between the entities are defined in MDM and used by TIBCO ActiveMatrix BusinessWorks to expose a composite entry as a service to other systems.



Introduction of MDM into the architecture in registry style

Benefits of MDM implementation

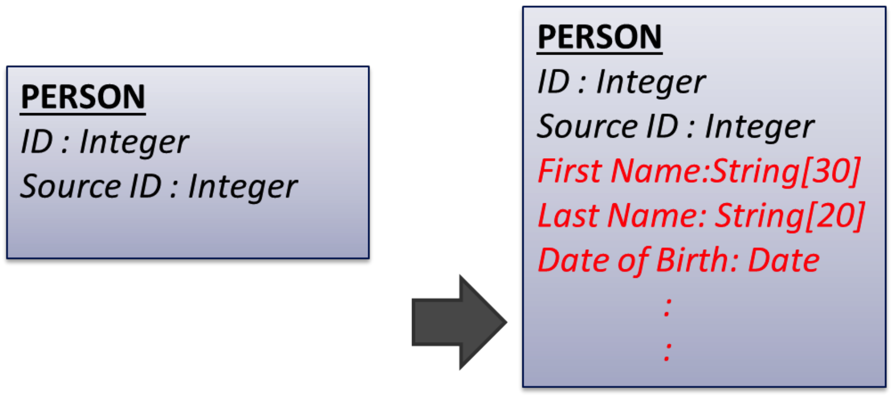
- Enables fast implementation when adequate thought is given to the data object exposed within the integration layer.
- Facilitates relationships between systems that don't currently exist within the local system definition of the data.
- Avoids changes to the way interfaces are used on existing systems.
- Allows composite services to be defined that encapsulate multiple source systems.
- Enforces data structures across systems. For example, an address must include a name.
- Permits events (add, delete, read, write) to be audited at a high level.
- Supports batch and/or real-time updates.

Issues

- Prohibits full data governance across the data objects due to differences between source systems.
- Impedes de-duplication and fuzzy search capabilities across systems.
- Limits insight into event data including history, versioning, and permissions unless each source system's audit logs are integrated.
- Complicates data domains by having multiple access points.

B) Move to Coexistence Implementation

Progressing from a reference implementation, entities within the MDM system can be enriched with more attributes as shown in the diagram below.



Expanding the entity attributes

The data domains will coexist within two or more systems, including MDM. When a record in the source system is created/updated or deleted, a method of propagation is required to update the record in the MDM system. The MDM system can then propagate this update to other systems if required (in real time or via a notification).

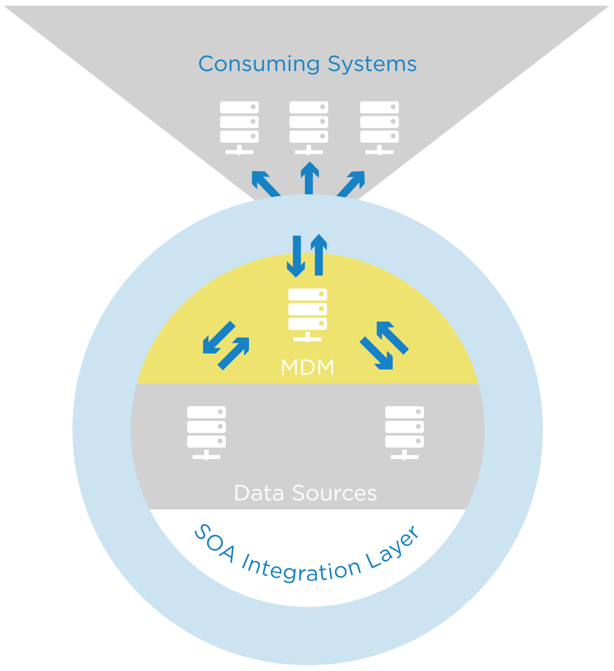
Various configurations can be implemented:

- Updates are only sent to the MDM system from the source system.
- Updates are only sent to the MDM system from the source system, but after data cleansing, can be returned to the source system, enabling source data quality to be improved over a period of time across all systems.
- De-duplication actions are propagated to the source system. This requires more complex integration logic and assumes the source system has the capability to handle a complex update that can request the suppression of records.
- By-directional updates are implemented, enabling updates in the source system and MDM system to propagate between each system.

This architecture uses MDM capabilities such as de-duplication, data cleansing, data enrichment, and searching, along with other data governance capabilities. Ideally, the MDM system eventually becomes the system of record, because integration around data objects can be migrated to the MDM interface using the architecture you established by adding MDM in the first level of implementation.

TIBCO Patterns, a powerful matching engine, is embedded within the TIBCO MDM platform to offer de-duplication and matching capabilities. To ensure master data meets required data quality standards, TIBCO MDM applies both pre-delivered and configurable validation and transformation rules.

Existing edit capabilities can remain in place as any updates to records are propagated to the MDM system via a web service call, JMS message, or batch file.



MDM in coexistence with other data sources

Benefits of coexisting data domain implementation

- Maintains the existing system interface (user interface, technical).
- Enables full audit capabilities. The full record is passed to the MDM system, so it becomes possible to track event and data changes.
- Creates a consistent data quality standard. Data cleansing and de-duplication is done within the MDM system.
- Allows you to move a source system interface in a controlled way by taking advantage of the original MDM environment. Over time, an interface can be migrated away from the source system to the interfaces of your initial implementation.
- Enables real-time updates and the propagation of data objects to other systems once the data becomes the system of record in MDM.
- Maintains the TIBCO MDM architecture, and in general, only requires configuration changes.

Issues

- Two-way communication between MDM and source systems may add complexity and therefore requires careful consideration and design.
  - Issues include race conditions, transaction integrity, and incompatible data standards (fields larger in one system, entities in one system may be represented as multiple entities in other systems).
- Creates synchronization issues between updates due to update latency when using batch integration from source and MDM systems.
- Complicates data governance across data objects. It is possible to create an MDM governance process that requires an approval process in MDM before the data object becomes accessible in MDM, but the source system may already make the same data object generally accessible.

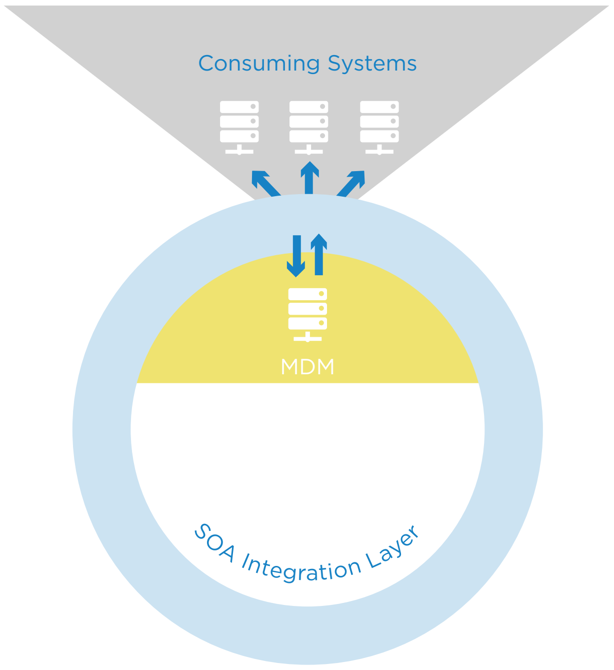
C) Master Data in One Place

This is the ideal implementation of an MDM implementation because it designates that master data domains reside only within MDM, creating a golden copy of the data object. This is the "ideal," but the reality is that not all data objects will or can be mastered within one enterprise-wide MDM system.

Where it is possible to master data in one system, total control over the data can be established. The full life cycle of the data objects can be managed, and security around the data can be defined in a consistent way.

Working in conjunction with all previous levels of implementation, the transition to a fully MDM managed data object should be relatively easy. The interface should, at this point in the evolution of the architecture, be known and well defined, so changing the data source should be a relatively simple reconfiguration exercise at the integration layer.

In the simplest implementation, this approach assumes that work has been done within the other levels of implementation to enable this transition to be performed as a re-wiring at the SOA layer. A more complex MDM implementation would require turning off the edit capabilities in the source system, establishing or adapting processes and procedures around the data objects within the source systems, and handling the migration of data from these systems to the MDM system if required.



Master data resides only in MDM

Benefits of All Master Data in the MDM System implementation

- Reduces IT costs
  - Enables a graceful retirement of source systems. This can be managed over a defined period (months/years.)
  - Does not distribute data over many systems, reducing hardware and management costs.
  - Focuses skills and knowledge on the MDM system.
- Allows you to apply a single data governance standard to the data object(s) and manage it entirely within the MDM system.
- Enables a full realization of security and user level auditing. All edits and validations happen within the context of the MDM system.
- Simplifies change requests. Data objects become adaptable, allowing for the evolution of the data over time.

Issues

- Not all source systems can be retired as they are deeply embedded within the originations enterprise.
- Easy to do as a green field implementation, but takes time if migrating existing systems and data domains.
- To truly take advantage of this implementation, the IT environment should be real-time enabled. Batch integration will not give organizations a competitive advantage.

Summary

MDM becomes more manageable when you define a set of services around key data objects within an SOA integration layer. Flexibility within the TIBCO component stack enables different data entities to be managed in different ways and at different times. Each entity can independently follow different MDM implementation paths within a common framework. This flexible approach enables source systems to be migrated out of an organization in a controlled way.

To prepare for the future of your data and allow for different implementation strategies for different data domains, choose an MDM solution with flexibility at the core of its integration technology and MDM platform.

Global Headquarters
3307 Hillview Avenue
Palo Alto, CA 94304

Tel: +1 650-846-1000
+1 800-420-8450
Fax: +1 650-846-1005

www.tibco.com