09 July 2012 G00234461
Analyst(s): *Marcus Collins*

Data management encompasses several interrelated technologies and techniques. It is also closely integrated with domains such as content management and application development. This reference architecture establishes a high-level context for examining the technologies, techniques, and related domains and is focused on data-management concerns; it is not intended to be exhaustive. For example, content and data integration typically entails integrating database management system (DBMS)-based data with content managed in non-DBMS systems.

What are the components of a master data management (MDM) initiative?

The general form of MDM system architecture provides a complete solution specification for a robust MDM program when it is matched with the data integration program specification explained in " Data Integration: Fantasies and Facts (http://www.gartner.com/document/code/203559?ref=grbody&refval=2076035&latest=true) ." It comprises five sets of functional components:

Development and governance tools and utilities for people to use when building, operating, and governing the MDM system

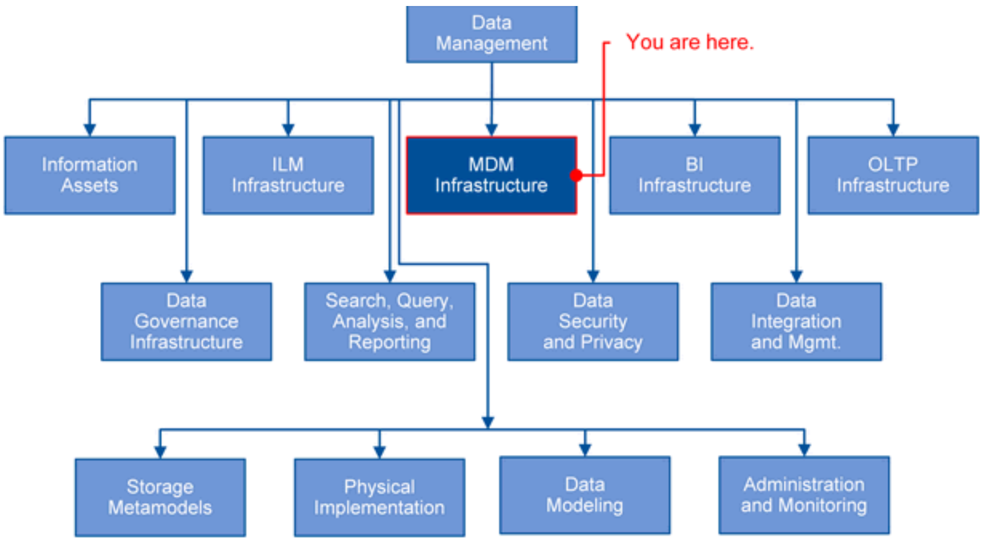
Master data definition registries for transforming chaotic data into master data

Data quality processes for a data quality management (DQM) program

MDM utility services that apply the master data definitions to the master data

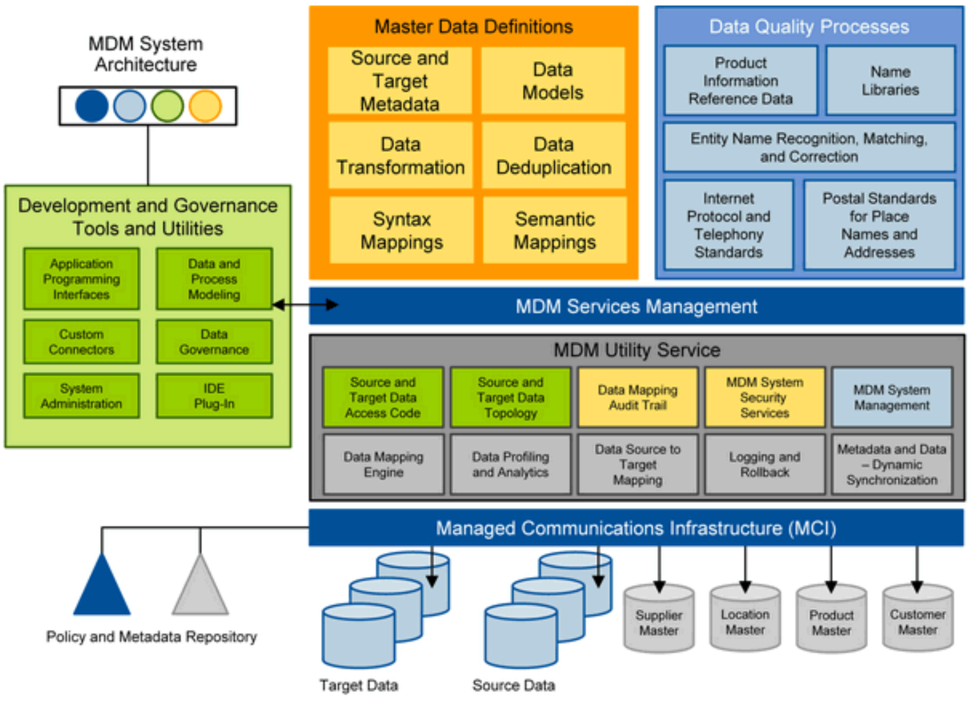
Managed Communications Infrastructure (MCI) and services buses—for the services that compose the MDM system—for connecting to data sources and targets and for processing data into master data

Figure 1. Data Management Reference Architecture



Source: Gartner (June 2012)

Figure 2. Master Data Management Component Diagram


MDM System Architecture
Development and Governance Tools and Utilities
Application Programming Interfaces
Data and Process Modeling
Custom Connectors
Data Governance
System Administration
IDE Plug-In
Master Data Definitions
Source and Target Metadata
Data Models
Data Transformation
Data Deduplication
Syntax Mappings
Semantic Mappings
Data Quality Processes
Product Information Reference Data
Name Libraries
Entity Name Recognition, Matching, and Correction
Internet Protocol and Telephony Standards
Postal Standards for Place Names and Addresses
MDM Services Management
MDM Utility Service
Source and Target Data Access Code
Source and Target Data Topology
Data Mapping Audit Trail
MDM System Security Services
MDM System Management
Data Mapping Engine
Data Profiling and Analytics
Data Source to Target Mapping
Logging and Rollback
Metadata and Data – Dynamic Synchronization
Managed Communications Infrastructure (MCI)
Policy and Metadata Repository
Target Data
Source Data
Supplier Master
Location Master
Product Master
Customer Master

Source: Gartner (June 2012)

Development and Governance Tools and Utilities

Six tools are required to build and maintain the data services platform master data management (DSP-MDM) platform. Development and business personnel use these tools:

Data governance: Data stewards are responsible for defining the business elements that constitute master data. They also have to keep definitions consistent with enterprise requirements. For example, the data steward defines the required components of customer name and customer address. Today, many data stewards perform this activity by writing data definitions in an Excel spreadsheet. An IT developer will receive a copy of the spreadsheet (e.g., by email) and use the data it contains to manually enter master data definitions. IT should provide data stewards with Web-client tools to perform their tasks, either by building or buying the appropriate tools. Several MDM products provide governance support tools.

Application programming interface (API): IT developers must define the APIs for runtime use by other services, applications, and databases in the IT infrastructure. These systems access master data from the DSP-MDM via the APIs. Although DSP platforms and MDM products provide API development tools, the IT development manager should insist that suppliers support tools already in use by the development staff.

Integrated development environment (IDE): MDM requires rigorously well-defined software development, data modeling, and other software development life cycle (SDLC) disciplines. Hence, products used for MDM that only offer proprietary development tools should be shunned in favor of DSP and MDM platforms that support an Eclipse IDE plug-in or that provide connectors to other market-leading IDEs.

Custom connectors: The MDM team must access data in stores that are unsupported by database connectors that are provided with data integration and MDM platforms. The development team must custom build those connectors. Typically, the developer will simply write plain old Java objects (POJOs) or plain old .NET objects (PONOs) to deploy new connectors. The packaged connectors are likely to be easily customizable with a code editor.

Data modeling: The MDM team requires detailed data models to create master data. A number of good data-modeling tools are available for this purpose. Those tools already used by an IT development group should be capable of exporting models to permit automated-models import by the MDM platform. If not, the data and process architects should design a physical data model that is provably consistent with the graphical models.

System administration: The MDM system must be set up, operated, and maintained by systems-engineering and operations personnel.

Master Data Definitions

An MDM system requires registries that contain formal definitions of master metadata, topology of sources and targets, business logic, transformation logic, metadata, models, semantic maps, and syntactical maps:

Source and target metadata: These are specifications of data types, formats, structures, and relationships that are specific to master data source systems and master data target systems. A master data source system is one from which data is obtained to be cleansed and transformed into master data. The master data may be returned to source systems as data updates or source systems may retain data in its original form. Target systems are ones to which master data is transferred. Hence, if a source system is to be updated with master data, it is also a target system. Target systems are usually consumers and not sources of master data. Target systems use master data to accomplish a specific business purpose (e.g., financial reporting). Whenever any data moves from one system to another, the middleware software—in this case the MDM system—uses source and target metadata to obtain data and to provide data in a format that is consistent with those targets and sources. That metadata is kept in the source- and target-metadata registry.

Data models: Master data must be modeled conceptually, logically, and physically. (For more information about data modeling, see " Data Modeling: A Necessary and Rewarding Aspect of Data Management (http://www.gartner.com/document/code/203540?ref=grbody&refval=2076035&latest=true) .") No MDM (or data integration) program proceeds without a serious data-modeling effort. In fact, MDM assumes the existence of accurate, current, and complete data models. These models are stored in the data-models registry.

Data transformation: Raw data from source systems must be transformed to be consistent with the specifications provided by data models. Data cleansing is one example of this. An incorrectly spelled name requires the data-transformation engine to request the logic required to correct the name. This information is stored in the data-transformation registry.

Data deduplication: An MDM system requires special logic to recognize multiple instances of the same master data. Master data is canonical. It cannot abide replication. The data-deduplication registry provides rules for recognizing and reliably removing replication.

Syntax mappings: Any source-to-target data transformation requires rules for automating decisions about retyping, restructuring, and reformatting data in one system into data for another system. The set of all syntax maps is closed, hence it is algorithmically solvable. The syntax-mappings registry contains the formal logic for solving the source-to-target mapping problem for individual data elements without loss of data fidelity.

Semantic mappings: The master data required by one business unit may be dissimilar from master data required by another business unit. A simple example of this is a consumer customer account versus a commercial customer account. The semantic value of "customer" is the same in both cases. The semantic value that satisfies "customer account" information is clearly dissimilar. More subtle differences also exist. For example, a product for one business unit might be a component for another business unit (e.g., one division of an automotive company manufactures engines that another division installs into cars that it assembles and sells to the public). The set of all semantic mappings is unbounded and uncountable in the general case; hence, it is unsolvable using discrete formal methods. The semantic-mappings registry contains the formal logic defined by business and technicians working together to correctly transform semantics of master data from one business context to another.

Data-Quality Processes

A data-quality management (DQM) process is a fundamental requirement for every MDM program. The best DQM program builds on Dr. W. Edwards Deming's "14 Points." [1] (#dv_notes)

A DQM process cleanses master data before storing it in a database management system (i.e., simple, local, and federated database styles), or back at its source (i.e., hub style). DQM uses specialized tools to correct customer names, supplier names, geographical location data, product information, and codes tables. For example, Pitney Bowes' Group 1 Software and Harte-Hanks' Trillium provide customer-and-location DQM software. A complete description of a data-quality program is well beyond the scope of this template. Nonetheless, DQM processing requires these functions and definitions:

Product-information reference data: Information about products is highly specific within an industry, an enterprise, and a business unit. The amount of information required to specify a product varies significantly from one business context to another. For example, it may be adequate for the MDM system to send this description of an electric motor to the financial reporting system: "Motor, electric, 20 amps, part no. X3321." That description of the motor would be woefully inadequate for the product catalog published on the company's website. The catalog requires this description: "Motor, electric. 20 amps, 220 volts, 60 hertz, dual poles, two phase, size 150 x 480 x 600 cm, black, chemically neutral housing, without mounting bracket. Part no. X3321." The product-information reference-data registry contains all product information required by any target system.

Name library: Names are often misspelled. Today, a large business stores names of people from many different cultures and traditions. The name library is a repository of valid names, gender, and rules for printing, transliterating, and appending titles to names.

Internet Protocol (IP) and telephony standards: Telephone numbers, email addresses, IP addresses, and Universal Resource Locators (URLs) must comply with international and national standards to be valid. The IP-and-telephony-standards registry contains this information.

Postal standards for place names and addresses: Government postal authorities meet regularly to revise standards for addresses and names of locations. Addresses can include a postal code in one country or ZIP Code in another country. These data are semantically similar, but not identical. Names of cities, states, regions, counties, countries, and other location information also must follow standards. The postal-standards registry contains this kind of information.

Entity-name recognition, matching, and correction: These are the primary functions of a DQM system. The entity-name processes registry contains the rules that are used by the DQM tools to determine which data elements are names and to which entity type the names refer (e.g., product, person, place, or code). This activity constitutes name recognition and matching (i.e., logical or statistical rules for deciding when two different strings of symbols represent the same entity name). For example, SanFrancisco, the City of San Francisco, and San Fransico all are intended to mean the same place—a hilly place at the end of a peninsula in northern California between San Francisco Bay and the Pacific Ocean. Name-recognition rules must determine that these strings of symbols are names of locations. Rules about misspellings, together with other contextual information, increase the likelihood that these strings of symbols refer to the same place. The rules stored in the registry would indicate the source for the accurate names as well as the transformations required to repair any errors.

MDM Services Management

This is a utility system that organizes MDM functions into an MDM process. IT uses services-bus or application-integration-middleware technology to provide means for one MDM service to call another. For instance, the data-governance utility requires access to the semantic-mapping registry if it is to show the contents to a data steward for review and approval.

MDM Utility Services

MDM utility services comprise operating components for the MDM system. The tools establish physical connections to master data sources and targets. They perform mapping, profiling, analysis, rollback and recovery, and metadata synchronization functions:

Source and target data-access code: The MDM needs query programs (code) written in the language of the source or target system to read and write master data. The code is provided in a suitable form for execution by the data-access code service (e.g., Structured Query Language [SQL] for relational database management systems [RDBMSs]).

Source and target data topology: The data-topology service records the metadata for tracking which master data belongs to which system.

Data-mapping audit trail: Master data is often business-confidential data. The sources, transformations, and uses of confidential data should be tracked from creation to deletion. The audit-trail service provides trace metadata to permit a data steward to review the sources and targets for master data and the uses for which that data has been put in the enterprise.

MDM system security services: The MDM system must be kept secure. The security services provide encryption, decryption, authentication, and access controls for the MDM system.

MDM system management: These services provide utility functions for development, systems-administration, and operations personnel to manage the MDM system.

Data-mapping engine: The data-mapping engine executes rules provided by the semantic and syntactical registries and data-transformation logic (e.g., data-quality error correction). The mapping service transforms raw data into master data per the processing instructions maintained by the MDM system.

Data profiling and analytics: These services are used for DQM. Data profiling identifies the cardinality of a data element (i.e., the count of the number of unique, valid values of a master datum). Analytics provides standard statistical operators for measuring relative frequencies, for establishing trends and variances relative to baseline measurements, and for making probabilistic matches when logical matching fails.

Data source-to-target mappings: This service tracks open associations between sources of master data and the target consumer of master data.

Logging and rollback: These services produce serially ordered records (i.e., log files). The records contain detailed information about create, read, update, and delete (CRUD) operations that any system or service has requested of the MDM registries manager or the master database. The MDM system can also use the log records to "undo" a write operation, complete a read operation, or start and stop any other operation on a registry or the master database. Usually, rollback services are invoked by the system administrator in the event of a system outage.

Dynamic synchronization of metadata and data: This service requests metadata from source and target systems. It automatically updates the MDM registries, topologies, and metadata stores with current information.

MDM Architectural Styles

An MDM system can be built using multiple styles: simple MDM, local MDM, or federated MDM.

Simple MDM

Simple MDM requires a serious and continuing commitment to DQM. It is simple because its design premise assumes only that:

1. Master data is worth managing to high-quality standards
2. Master data needs to be readily and securely accessible to target applications

To meet the latter condition, IT must store only accurate and reliable master data in a modern database—a master database—that is fault tolerant, recoverable, and fast. Data-access tools must have secure and reliable access to connect to the master database, query its contents, and obtain information most business processes will require. Stringent controls must be implemented in the master database to prevent master data corruption or theft.

Local MDM

Local MDM applies simple MDM to master data for one business unit. The benefit of local MDM is that it allows IT to plot a strategy for increasing coverage of an MDM program one business unit at a time. This strategy benefits the business in two ways. First, it drives a proverbial "stake in the ground" for an MDM program. Second, success with one business unit usually engenders internecine jealousies among business-unit leaders, who then demand their own MDM programs. This is the optimal outcome of the program.

Local MDM comes with a risk: that master data, as defined by one business unit, will be insufficiently complete for reuse by another business unit. This can result in multiple instances of the same master data being differently modeled across the enterprise. Those differences must be resolved at the enterprise level. However, if master data models and master metadata remain in a central repository for the MDM team to reuse, then subsequent models will trend toward the enterprise view. If not, the experience gained by developing local MDMs should readily translate into a master meta-model of master data. The master meta-model will help a data architect map master data from multiple business systems into an enterprise view of that data.

Federated MDM

Federated MDM is comparable to a data-integration program. The lessons taught in " Data Integration: Fantasies and Facts (http://www.gartner.com/document/code/203559?ref=grbody&refval=2076035&latest=true) " apply to federated MDM. Any IT organization that deploys federated MDM should follow the program outlined in that overview. There are three styles of federated MDM: hub, federated database, and DSP. (For more information about DSPs, see " Data Services: Data Access in Service-Oriented Architecture (http://www.gartner.com/document/code/203334?ref=grbody&refval=2076035&latest=true) .") The federated style of MDM subsumes the same level of quality and consistency—and the same management processes—as those used for simple and local MDM.

Hub MDM

The hub-MDM architectural style uses data-integration technology to create master data from existing data stores. The MDM hub retrieves data, cleanses it, removes duplicates, and remaps the data to a prescribed model. The resultant master data moves from the hub to target systems, the original source, and other systems that consume master data. Most hub-MDM systems can expose their MDM processes as Web services.

Federated Database MDM

The federated-database-MDM architectural style uses a data-integration hub with an integrated master database (also called a master data registry). The federated database style of MDM functions identically to the hub MDM with an attached database. The federation technology used in this database style can be built by using only a DBMS with federated query capability.

Data Services Platform MDM

The third style of federated MDM is data services platform (DSP). This style can be built onto simple, local, or federated MDM systems. In a DSP architectural style, source and target applications call a DSP API to submit master data for processing, to retrieve master data, and to update master data. The best approach to building the DSP MDM is to use service-oriented architecture (SOA) and DSP to manage all master data processes, including data quality, semantic matching, data transformation, metadata mapping, governance, security compliance, logging, and data storage.

Regardless of which MDM architectural style an organization chooses to employ, and although the organization and operation of a successful data-integration program apply to any MDM program, details concerning data integration are not repeated here (for more information, refer to " Data Integration: Fantasies and Facts (http://www.gartner.com/document/code/203559?ref=grbody&refval=2076035&latest=true) "). However, MDM system architecture can be generalized to include all the architectural styles. IT development should not assume that MDM system architecture represents one or more MDM products, although it does meet system-design requirements for all MDM programs.

[1] Deming, W. Edwards (1986). *Out of the Crisis* . Cambridge, Mass: MIT Press, 2000.

June 2012

Updated content and document links.

May 2012

Revised to include updated figures and template map.

May 2010

This is the initial iteration of this template.

Master Data Management - 09 July 2012 (http://www.gartner.com/document/2076035?ref=ddrec)

Master Data Management - 13 May 2010 (http://www.gartner.com/document/1405748?ref=ddrec)