
# SAS® MDM 4.1
## Administrator's Guide
**Second Edition**

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2014. *SAS® MDM 4.1: Administrator's Guide, Second Edition*. Cary, NC: SAS Institute Inc.

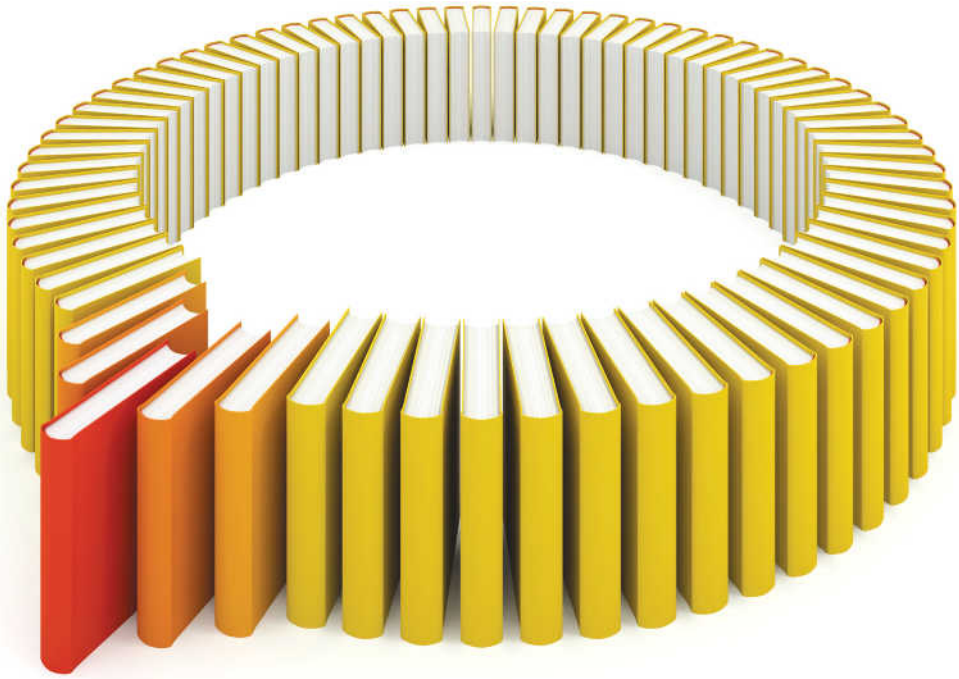
**SAS® MDM 4.1: Administrator's Guide, Second Edition**



SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513-2414.

April 2014

SAS provides a complete selection of books and electronic products to help customers use SAS® software to its fullest potential. For more information about our offerings, visit **support.sas.com/bookstore** or call 1-800-727-3228.



# Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
*for additional books and resources.*

# Contents

# Recommended Reading

- *DataFlux Data Management Studio Installation and Configuration Guide*
- *DataFlux Data Management Studio User's Guide*
- *DataFlux Data Management Server Administrator's Guide*
- *DataFlux Data Management Server Users's Guide*
- *SAS Workflow Studio User's Guide*
- *SAS Intelligence Platform Middle-Tier Administration Guide*
- *SAS Management Console User's Guide*

For a complete list of SAS books, go to support.sas.com/bookstore. If you have questions about which titles you need, please contact a SAS Book Sales Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-3228
Fax: 1-919-677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/bookstore

*Chapter 1*

# Overview of SAS MDM



## About SAS MDM

SAS MDM is a combination of software, templates, documentation, data models, and services that provide the functionality and processes necessary to build and maintain a master entity database. An entity can be a customer, product, patient, site, or any business data object that you define. You can also define attributes for each entity, as well as transformations that support data quality and identity management functionality.

The SAS MDM hub is a set of tables and table relationships that contain entity information and required entity keys and linking information. This hub provides end-users or other applications with a complete view of an entity and its relationships with other entities. In addition, the hub can be used as the single source for creating and maintaining survivor records that can be used in other applications or processes. The original source system IDs, which are important to other enterprise applications or data sources, are also maintained in the hub to facilitate linking activities with other data providers or consumers.

*Note:* SAS MDM is integrated with DataFlux Data Management Platform. Therefore, to use SAS MDM successfully, you must be familiar with DataFlux Data Management Platform.

SAS Data Remediation enables users to manage and correct issues triggered by business rules in SAS MDM batch jobs and real-time processes. These issues can also be triggered by external systems that communicate with data remediation through its REST-based web service application programming interface (API). Data remediation allows user- or role-based access to data exceptions, which are categorized by application and subject area. Once data remediation issues have been reviewed, they can be corrected through the same application, eliminating the need for another user to complete the

correction process. All data remediation issues can also be associated with workflow definitions that route the issues to the correct decision maker for approval or instructions for additional action.

SAS MDM includes the following features:

- batch and real-time modes
- entity matching, de-duplication, and survivor record creation
- entity linking through relationships
- data quality functionality such as address verification, standardization, and parsing
- fielded and advanced search capabilities
- capability of adding, modifying, and retiring SAS MDM entities
- workflow-based data remediation functionality to view and correct SAS MDM data errors
- metadata management for entities, attributes, and transformations
- historical tracking, including changes saved for contributor records and survivor records
- data access, data profiling, data monitoring, data quality, and data enrichment

## How SAS MDM Works

SAS MDM performs the following functions:

- extracts business information from your data sources
- validates and standardizes the data
- captures data errors through user-defined business rules and sequesters the data for review and correction
- consolidates the information into a single view of the information available from all the data sources

The organizational data that you provide can be customer data, product data, services data, or data for other entity types. SAS MDM applies a rigorous methodology to the problem of integrating disparate enterprise data.

## Architecture

### *Overview*

SAS MDM is built on a sophisticated data management platform that provides both development and run-time support for applications.

### *Development*

During development, DataFlux Data Management Studio provides the ability to design and test processes (batch jobs) and real-time services (data jobs), which can then be deployed and run using the following:

- DataFlux Data Management Server
- DataFlux Data Management Studio
- scripting with a command line interface

The DataFlux Data Management Platform relies on the Quality Knowledge Base (QKB) for data quality processing. The QKB provides capabilities such as standardization of data and computation of match codes. The QKB is used both during development and at run time.

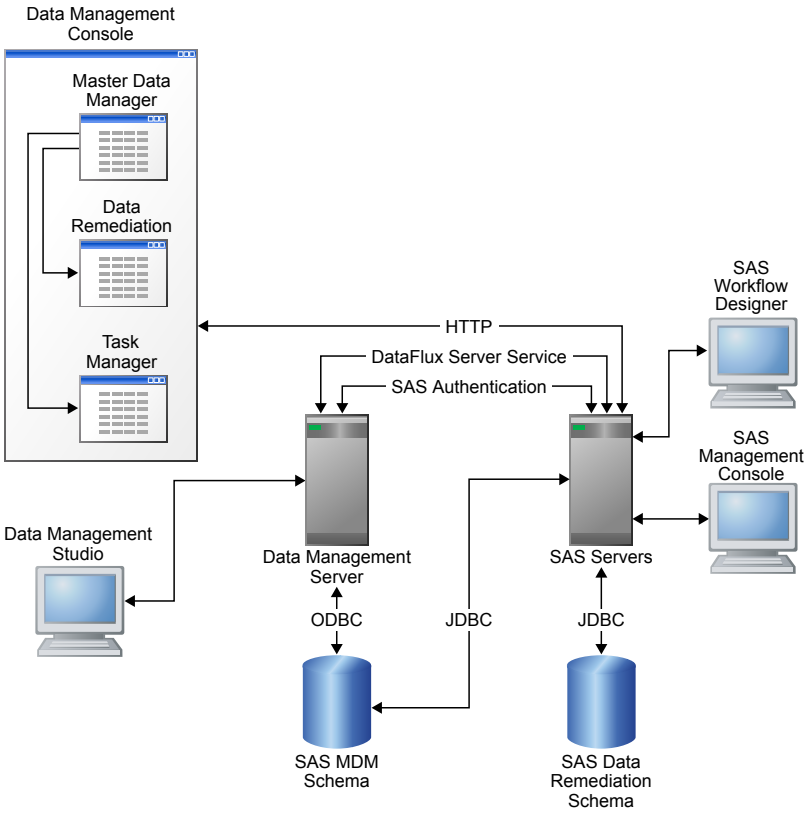
### *Run Time*

At run time, a multi-tier architecture that consists of presentation, business, and data tiers is used.

- The presentation tier hosts the web interface in a Servlet 2.5-compliant container and includes a Java application server and a browser. Batch interfaces might be available for some operations such as bulk loads and updates.
- The business tier runs on a DataFlux Data Management Server and is implemented using DataFlux Data Management Studio processes (batch jobs) and real-time services (data jobs). It implements a Service Oriented Architecture (SOA) that enables any system to use its services through SOAP requests. Common web service standards are used to maintain simplicity when integrating with other systems. Authentication within the business tier runs on a SAS Metadata Server. This server can authorize user permissions for applications, assign roles and security restrictions, and define authentication parameters for accessing other systems, such as the database, for each user or group. The real-time or transactional component of SAS MDM interfaces with other applications through an SOA environment. Specifically, the ability to call business services hosted on DataFlux Data Management Server is accomplished through web services. Process jobs can also be used as business services that accept data on a record-by-record basis. This enables users to use SAS MDM data and services within an enterprise architecture.
- The data tier hosts the supporting data and metadata for the system, which consists of a relational database such as Oracle or Microsoft SQL Server. The data model is documented and accessible to third-party tools and can also be accessed through DataFlux Data Management Studio jobs.

The following display shows how the components fit together:

***Figure 1.1*** *Architecture Diagram*




## SAS MDM Components

### DataFlux Data Management Studio

DataFlux Data Management Studio is a graphical user interface (GUI)-based application that provides access to other tools and services. You can use DataFlux Data Management Studio for job flow management and customization, to manage data source connections, to manage user accounts through SAS Metadata Server, to deploy and execute jobs on the DataFlux Data Management Server, and to run batch jobs and data services within a Windows environment.

### DataFlux Process Jobs and Real-time Business Services

A DataFlux process job is a multi-step data management process that can be run from DataFlux Data Management Studio, from DataFlux Data Management Server, or from a command line interface or script. Real-time business services, also called data jobs, are deployed through DataFlux Data Management Server and are used by the SAS MDM web application to provide data to the application through SOAP calls.

### Quality Knowledge Base

The Quality Knowledge Base (QKB) contains the files, file relationships, and metadata needed to correctly parse, match, standardize, and otherwise process data as required for SAS MDM. The QKB can be customized to meet an organization's needs. The QKB can support different locales to support processing for many different languages.

### DataFlux Data Management Server

The DataFlux Data Management Server enables data management and integration processes to be deployed across an entire IT environment. SAS MDM uses batch jobs and real-time services deployed through DataFlux Data Management Server to manage the master data hub and to provide data through the web interface and reports. DataFlux Data Management Server handles all aspects of database connectivity and service availability through SOAP calls.

*Note:* In many instances, you should have at least two DataFlux Data Management Servers working together through network load balancing or failover hardware or software, to achieve the desired results for performance and meet server uptime goals, particularly for real-time service processing.

### Databases

SAS MDM relies on access to a Database Management System (DBMS), such as Oracle or SQL Server, to act as a repository for metadata and data. The Master Data Management component of SAS MDM requires two database schemas for its operation.

### SAS Data Management Console

The SAS Data Management Console is a central, web-based location for all master data management activities. This environment provides a launch location for all associated SAS MDM applications, as well as other SAS applications.

The SAS MDM applications Master Data Manager, Data Remediation, and Task Manager enable you to perform the following functions:

***Table 1.1*** *SAS MDM Applications*

| Application | Functions |
|---|---|
| Master Data Manager | • Create and manage a hub.<br>• Define new entity types and their attributes.<br>• Search for data within the hub.<br>• Create and update data within the hub. |
| Data Remediation | Monitor and correct errors in data loaded into SAS MDM. |
| Task Manager | Start, stop, view, and interact with active workflow instances. |

### SAS Workflow Designer

This application is a drag-and-drop workflow designer. Users can modify existing MDM workflow templates or create new ones for use with other SAS MDM processes. Workflows can also be uploaded and activated in the SAS server environment and can be downloaded for additional editing.

### SAS Application Server

An application server hosts the SAS Data Management Console and other associated applications. This is installed and configured during the deployment process. No other third-party application servers are supported.

### *SAS Management Console*

This application enables users to manage groups, users, and roles on the SAS Metadata Server for all SAS MDM web applications. It also is the location where various web application configuration parameters are set.

## Deployment Models

The most common type of deployment for SAS MDM is the physical deployment model. This model assumes that all the information necessary to construct a complete survivor record for an entity is stored in the hub. Source system keys are also stored in the hub so that information used to identify and describe entities can be recorded. This supports the ability to link or reference source system data, provides traceability for the contributor records, and describes key relationships with the data as it exists in the hub.

With this approach, uniquely identifying information for entities is stored in the hub, along with other dependent attributes. You can access the survivor record by querying the hub by itself, without the need to refer to the systems of record. This approach simplifies the task of constructing the survivor record, but results in higher storage requirements.

The physical deployment model is the primary implementation, but SAS MDM also works in conjunction with other technologies to provide functionality that supports deployment styles such as hybrid MDM, registry-based MDM, reference-style MDM, and other integration scenarios.

## Integrating Other Applications with SAS MDM

You can integrate other end-user or enterprise applications, as well as other Extraction, Transformation, and Loading (ETL) software with SAS MDM.

Two methods for integration are as follows:

- direct-database or flat-file access, running batch load or batch update processes. In this case, the DataFlux Data Management Server can connect directly to data sources, transform the data, and load it into the hub.
- communicating with the DataFlux Data Management Server using web services. Any application or process that can make a web service call can send data to the hub through DataFlux Data Management Server and receive data from the hub in the same manner.

Both methods are typically used to interact with SAS MDM. In some cases it might make sense to communicate directly with DataFlux Data Management Server from an application integration layer of an enterprise application such as Siebel or SAP. In other cases, using an enterprise messaging mechanism such as TIBCO or webMethods might be more practical if that infrastructure has already been deployed within your organization.

You also have the option of making web service calls directly from a .NET or Java component or application that can use web services. This might be common in circumstances where internally designed and built operational or reporting processes need to interact with the DataFlux Data Management Server to access customer or service data inside the hub.

*Chapter 2*

# Installing SAS MDM



## Installing the Software for the First Time

### *Overview*

SAS MDM is available through SAS delivery channels. For information about installing this product, see your SAS Software Order E-mail (SOE).

The following table lists default software locations:

| Description | Default Path |
|---|---|
| Windows installation | `<SAS Home Directory>\<product_instance_name>\<version>` |
| UNIX installation | `<SAS Home Directory>\<product_instance_name>\<version>` |
| SAS MDM support files and the associated data management repository on client machines | `C:\ProgramData\SAS\SASMDM\<version>` |
| SAS mid-tier servers | `\SAS\<configuration_name>` |

SAS MDM installation and configuration is a multi-step process and, depending on your target host systems, might need to be repeated on multiple computers if your deployment plan requires this.

## Pre-Installation

Your SAS SOE directs you to the complete pre-installation checklist. Here is the basic pre-installation information:

- Your plan file and sid file must be in a directory accessible to the SAS Deployment Wizard (SDW). Accept the SDW default configuration settings unless you are otherwise instructed.
- You must install a 32-bit JRE on the DataFlux Data Management Studio client machine and a 64-bit JRE on the DataFlux Data Management Server to support the data remediation Java application used by SAS MDM. If you install the JRE after deploying your SAS software, you must update the JAVA information in your app.cfg file for both applications.

  *Note:* Both DataFlux Data Management Studio and DataFlux Data Management Server must have Read and Write access to the file system where the data remediation records are stored.
- JDBC drivers must be downloaded from the website of the database vendor to a directory before beginning the installation process.

  *Note:* Only JDBC drivers, and no other files, should be placed in this directory.

  The SDW prompts you for the directory where the JDBC drivers are stored. The SAS MDM web application needs these files to connect to the SAS MDM target database.

During the installation and configuration process, you are asked for information about your target SAS MDM database. The target database is the Oracle or SQL Server database into which your master data is loaded. DataFlux Data Management Studio and DataFlux Data Management Server must also connect to the SAS MDM target database. Have the following information available before you start the installation process:

***Table 2.1*** *Database Parameters*

| Parameter | Default Values |
|---|---|
| SAS MDM Database vendor | Oracle or SQL Server |
| SAS MDM Database DSN | SASMDM41 |
| SAS MDM Database name | mdm41 |
| SAS MDM Database schema | MDM41 |
| SAS MDM Database port | Oracle: 1521 or SQL Server: 1433 |
| SAS MDM Database Oracle SID | mdm41 |
| SAS MDM Database password | SASMDM41 |
| SAS MDM Database host | Host where the database is installed |

For more information about how to define users, schemas, tablespaces, and other key features required for SAS MDM to deploy correctly, see "Configuring Databases" on page 11. At a minimum, you must supply database and user configuration information, or the application will not start successfully.

### Post-Installation

For post-installation configuration instructions, see Chapter 4, "Configuring SAS MDM," on page 17.

## SAS MDM Update Process

### Overview

If you already have SAS MDM installed, the installation and configuration process takes steps to upgrade your existing environment. This involves an MDM database schema evolution process, updates to web applications, and the delivery of updated or new MDM jobs in the DataFlux Data Management Platform environment. Please review the following information to prepare for this process:

1. Your SAS MDM target database schema is updated to the latest version. Back up this database before the upgrade process begins.
2. Your SAS Data Remediation database is updated to the latest version. Back up this database before the upgrade process begins.

   *Note:* Both DataFlux Data Management Studio and DataFlux Data Management Server must have Read and Write access to the file system where the data remediation records are stored.
3. Consider backing up your Data Management repository file directories for both DataFlux Data Management Studio and DataFlux Data Management Server.

### Migration

Migration from DataFlux qMDM Solution to SAS MDM is not supported. For help with migration issues, contact SAS Information Management Consulting at:

**http://www.sas.com/software/information-management/consulting.html** and **http://www.sas.com/reg/standard/corp/gendetail**.

### Post-Update

For post-update configuration instructions, see "Post-Update Tasks for SAS MDM Updates" on page 21.

## System Requirements

You can review system requirements for SAS products at the following location:

**http://support.sas.com/documentation/installcenter/index.html**

Select the release. A search window for that install center documentation appears. Search for your product name. A results page appears with links to the system requirements for your software.

## SAS MDM Content

The SAS MDM installation process copies several directories and many files onto your systems. On DataFlux Data Management Studio, a repository named SASMDM is created for you. This repository contains all the files associated with SAS MDM, including data jobs that can be run on the client or on DataFlux Data Management Server, database scripts, sample workflow templates, and other files.

On DataFlux Data Management Server, this information is copied into SASMDM folders either in the **var** directory or in the **share** directory.

On Windows, the directory structure looks like this:

```
\ProgramData\SAS\SASMDM\4.1\Repository\SASMDM
    \DataStorage                          [Studio repository metadata]
    \FileStorage\batch_jobs\sasmdm        [All batch data jobs for SAS MDM]
                \data_services\sasmdm     [All real-time data service jobs for SAS MDM]
                \process_services\sasmdm  [All process server jobs for SAS MDM]
                \sasmdm\samples\data      [Source data files for sample entities]
                               \scripts   [Script files associated with sample entities]
                \scripts                  [Database creation scripts and other metadata support files]
                \workflows                [Example workflow templates]

C:\ProgramData\SAS\SASMDM\4.1\sasmdm
    \archive           [Location where artifacts from the data archiving process reside]
    \data              [Location for temporary data files]
    \logs              [Location for log files generated by certain SAS MDM processes like bulk loading]
    \remediation_data  [Location for  data files generated by data remediation processing]

C:\Program Files\SASHome\DataFluxDataManagementServer\2.5\var\sasmdm
    \archive           [Location where artifacts from the data archiving process reside]
    \data              [Location for temporary data files]
    \logs              [Location for log files generated by certain SAS MDM processes like bulk loading]
    \remediation_data  [Location for  data files generated by data remediation processing]
```

*Note:* The **\remediation_data** directory can contain files that might have identifiable information in them, depending on how you configure your data remediation application to accept data from SAS MDM processes. Ensure protection of the contents of this directory.

*Chapter 3*

# Configuring Databases



## Metadata and Instance Data

SAS MDM builds a repository of metadata and instance data. This requires, at a minimum, either Oracle 11gR2, SQL Server 2008, or SQL Server 2012. Ensure that the character set for the database supports your data requirements. For example, not all accented characters (such as the cedilla, umlaut, or circumflex characters) or special-use characters (such as trademark, copyright, or Euro symbols) can be stored in all character sets. SAS recommends using a UTF character set to ensure maximum compatibility between the data and your database. SAS also recommends that databases be configured to ignore accents when performing character comparisons.

## Building the SAS MDM Repository

### *Overview*

This section explains the process of creating a new SAS MDM repository. You must have access to a relational database system and have read, write, and table creation permissions. This database system can be either local or on a network.

Create a user or schema named MDM41 defining common privileges (for example, table reads) as well as the following:

- create procedure
- create sequence

- create table
- create view
- create trigger

You might need to grant extra tablespace to this schema, depending on your database configuration. You might also choose to create a temporary user or schema for development activities that can add, modify, and drop database objects as needed in that schema.

*Note:* For Oracle installations, a separate schema must be created to manage encryption. The user name mdm_secure and table name mdm_secure_info must not be changed. Database limitations for encryption also apply.

## Creating Databases and Tablespaces

### Overview

The creation of databases should use the standards and best practices of your organization. The commands that follow are examples and can be modified to meet your requirements.

### Oracle

Create tablespaces to hold data and indexes, modifying values as needed for your system. Note that the file mdm_oracle_ddl.sql is hardcoded to use QMDM_DATA and QMDM_INDICES as tablespace names. If you are not using the same tablespace names, then the file mdm_oracle_ddl.sql must be modified to match the tablespace names that you use. For more information, see your database documentation.

As sysdba, execute the following SQL statements after modifying them as required for your installation:

```
-- create new tablespace
create tablespace qmdm_data datafile
'C:\oracle\product\10.2.0\oradata\sas\qmdm_data.dbf' size
10M autoextend
on extent management local autoallocate;
create tablespace qI mdm_indices datafile
'C:\oracle\product\10.2.0\oradata\sas\qmdm_indices.dbf' size
10M autoextend
on extent management local autoallocate;
-- create mdm user and grant privileges
create user MDM41 identified by SASMDM41 default tablespace
qmdm_data quota unlimited on qmdm_data quota unlimited on
qmdm_indices;
grant connect, resource, create table, create view, create
procedure, create trigger, create session, create sequence
to MDM41;
```

### SQL Server

For SQL Server installations, create a separate database for SAS MDM. Do not install it in the master database. For more information, see your database documentation.

As the sa user, execute the following SQL statements after modifying them as required for your installation:

```
create database mdm41
```

```
create login mdm41 with password = 'SASMDM41';
go
-- Create user
use mdm41;
create user mdm41 for login mdm41 with default_schema=mdm41;
grant connect, create table, create view, create procedure,
create schema, showplan to mdm41;
grant execute on schema::dbo to mdm41;
go
```

As the mdm41 user, execute the following SQL statements after modifying them as required for your installation:

```
-- Login as mdm41
use mdm41;
go
create schema mdm41;
go
```

## *Setting Databases for Encryption*

### *Overview*

SAS MDM provides the ability to encrypt data being stored in the hub. This feature requires additional settings and grants for Oracle and SQL Server. Even if you do not use encryption, you must set your SAS MDM database environment to support it.

### *Oracle*

For Oracle installations, you must create a separate user to manage encryption. The user name mdm_secure and table name mdm_secure_info must not be changed. Database limitations for encryption also apply.

```
-- create mdm_secure schema
create user mdm_secure identified by SASMDM41 default tablespace
qmdm_data;
grant connect, resource, create table to mdm_secure;
grant execute on UTL_RAW to mdm_secure;
-- create secure info table
create table mdm_secure.MDM_SECURE_INFO
(
MDM_ENCRYPTION_KEY RAW(2000)
)
tablespace QMDM_DATA;
-- Insert key into table
insert into mdm_secure.mdm_secure_info
select utl_raw.cast_to_raw('sasmdm') from dual;
-- Grant privileges to MDM41 user
grant execute on DBMS_CRYPTO to MDM41;
grant execute on UTL_RAW to MDM41;
grant select on mdm_secure.mdm_secure_info to MDM41;
```

### *SQL Server*

For SQL Server installations, you must create an encryption key and certificate. The key name qMDMKey and certificate name qMDMCer must not be changed. Database limitations for encryption also apply. As sa user, execute the following:

```
use mdm41;
-- Create encryption certificate and key
create master key encryption by password='SASMDM41';
create certificate qMDMCer with subject='sasmdm';
create symmetric key qMDMKey with algorithm=triple_des encryption
by certificate qMDMCer;
go
grant control on certificate :: [qMDMCer] to [mdm41];
grant view definition on symmetric key :: [qMDMKey] to [mdm41];
go
```

## Creating a Data Source

SAS MDM accesses DataFlux services on DataFlux Data Management Server and those services access the relational database that hosts the SAS MDM database. To access a database with DataFlux Data Management Studio, an Open Database Connectivity (ODBC) driver for the specified DBMS must be installed, and the database must be configured as an ODBC data source. When this has been completed successfully, the database name appears in the Data Connections folder on the DataFlux Data Management Studio riser.

The SAS MDM installation and configuration process creates the required data source names (DSNs) for you. DSNs are created using the DataFlux ODBC Wire Protocol drivers for either Oracle or SQL Server with the correct driver options set. These two drivers are the only ones supported for communicating with the SAS MDM target database from the DataFlux Data Management Platform. If you need to modify the default DSNs, use the following instructions.

*Note:* The ODBC Reference documentation that is available in every DataFlux Data Management Studio installation can help you with ODBC data source configuration.

## Special Database Driver Considerations

Because SAS MDM uses stored procedures in the target database, you must enable these procedures using ODBC Connection Manager when configuring your database connections. These settings are applied for you during the SAS MDM installation and configuration process and are shown here for reference. For example, if you want to use the Oracle Wire Protocol Driver, you must enable the following settings:

***Table 3.1*** *Driver Settings*

| Advanced Settings | Performance Settings |
|---|---|
| Application Using Threads | Use Current Schema for SQL Procedures |
| Enable Procedure Returns Results | Catalog Functions Include Synonyms |
| Enable SQLDescribeParam | Enable Scrollable Cursors |
| Describe at Prepare | Wire Protocol Mode: 2 |
| Enable N-Char Support | |

*Note:*

- The DSN must be configured in the mdm_macros.cfg file. The SAS MDM installer default DSN is SASMDM41, but the DSN can be edited as needed. If you edit the DSN, ensure that the mdm_macros.cfg file is updated with the new name (including the copy on the DataFlux Data Management Server).
- Additional driver settings can be configured for optimum performance. For more information, see your database driver documentation.

## DNS Resolution of Host Names

You might encounter slow response time with Domain Name System (DNS) resolution of host names for data sources. Consider using the IP Address in the ODBC DSN configuration to avoid DNS lookups. You can also modify the hosts file for your server to avoid DNS lookups. For Windows platforms, this file is typically **`C:\Windows\System32\drivers\etc\hosts`**. For UNIX platforms, the file is **`etc/hosts`**.

## Using Database Utilities

Some SAS MDM batch jobs give you the option of using database utilities to improve software interactions with the database. For example, bulk load jobs can be set to use database utilities that might perform the operations more efficiently. Some of these job templates use calls to command line applications that are specific to either Oracle or SQL Server. SAS MDM requires access to these applications.

Here are the database-specific calls:

- SQL*Plus
- SQL Loader
- sqlcmd
- bcp

You are not normally required to use the database utilities. However, for bulk-loading operations, you might see large performance gains when loading your hub. See your database documentation for more information about the specific calls.

To use Oracle database utilities properly:

1. Do one of the following:
   - On Windows, set the path to the Oracle bin directory in your system PATH variable.
   - On UNIX, set the system variable ORACLE_HOME. Then export $ORACLE_HOME to the PATH.
2. Enter the database location description in the Oracle tnsnames.ora file. The following is a typical format:

```
ORCL =
    (DESCRIPTION =
        (ADDRESS_LIST =
                  (ADDRESS = (PROTOCOL = TCP)(HOST =
 your_server_name)(PORT = 1521))
    )
```

```
    (CONNECT_DATA =
      (SERVICE_NAME = orcl)
    )
  )
```

*Chapter 4*

# Configuring SAS MDM



## Post-Installation Tasks for New Installations

After the SAS Deployment Wizard has finished the installation, you can complete post-installation tasks.

*Note:* On Windows systems, you might need to modify file directory security options to display some of the locations where SAS MDM content is stored.

Here are the post-installation tasks:

1. Set up your database user, schema, tablespace, and other required database parameters, if you have not already done so. See "Configuring Databases" on page 11 for information about these parameters.
2. Verify that DSNs have been created for the SAS MDM target database. By default, these are set to SASMDM41.
   - For DataFlux Data Management Studio, open DataFlux Data Management Studio and verify through the data riser that you have access to your database. Your data source should be listed under **Data Connections**.
   - For DataFlux Data Management Server:

1. Add a server under the **Data Management Servers** riser.
2. Open the server and locate the **Data Connections** tab.
3. Open the **Manage ODBC Connections** dialog box. Your data source should be listed.

3. Ensure that the connections in DataFlux Data Management Studio are saved for both DataFlux Data Management Studio and for DataFlux Data Management Server.
4. Verify that the mdm_macros.cfg file for DataFlux Data Management Studio and for DataFlux Data Management Server contains the correct information for the following macros: MDM_HUB_DB, MDM_DSN, MDM_SCHEMA, and MDM_PACKAGE. See "Setting SAS MDM Macros" on page 22 for details.
5. Download and install the QKB from **`http://support.sas.com/qkbdownload`**. Verify that the mdm_macros.cfg file for DataFlux Data Management Studio and for DataFlux Data Management Server contains the correct QKB location information for the MDM_QKB macro.
6. (Optional) If you want to use Data Remediation functionality for DataFlux Data Management Studio and for DataFlux Data Management Server, you must update the mdm_macros.cfg files. Enter the correct remediation URL, user name, and password information in the following macros: MDM_REMEDIATION_URL, MDM_REMEDIATION_USER, and MDM_REMEDIATION_PASSWORD.

   *Note:* The password in MDM_REMEDIATION_PASSWORD must be base-64 encoded.

   See "Macro Descriptions" on page 23 for more information about macros.
7. (Optional) Download and install address verification data from **`http://support.sas.com/demosdownloads/setupcat.jsp?cat=DataFlux+Data+Updates`**.
8. Start the DataFlux Data Management Server service or process from the machine on which it was installed.
9. Add an administrative user to the SAS Administrators group and add users to the default Data Management groups in the SAS Management Console. For setup and configuration, ensure that the user identity for administrative and configuration tasks is a member of the SAS Administrators and Data Management Administrators groups. See "Security Considerations for SAS MDM" on page 25 for complete information.
10. Log on to DataFlux Data Management Server from DataFlux Data Management Studio as a member of the SAS Administrators group. Assign access privileges to other groups that need to access the Master Data Management application. See "Security Considerations for SAS MDM" on page 25 for complete information.
11. (Optional) Configure Workflow as follows:

    a. As a system administrator, add the Data Management Administrators group to the ROLE_WORKFLOW_ADMINISTRATOR role using the SAS Web Administration Console. The default URL is **`http://<host>:<port>/SASAdmin`**. Port 80 is the default port for Windows systems. Port 7980 is the default port for UNIX systems.

    b. Open SAS Workflow Studio and connect to the default SAS environment.

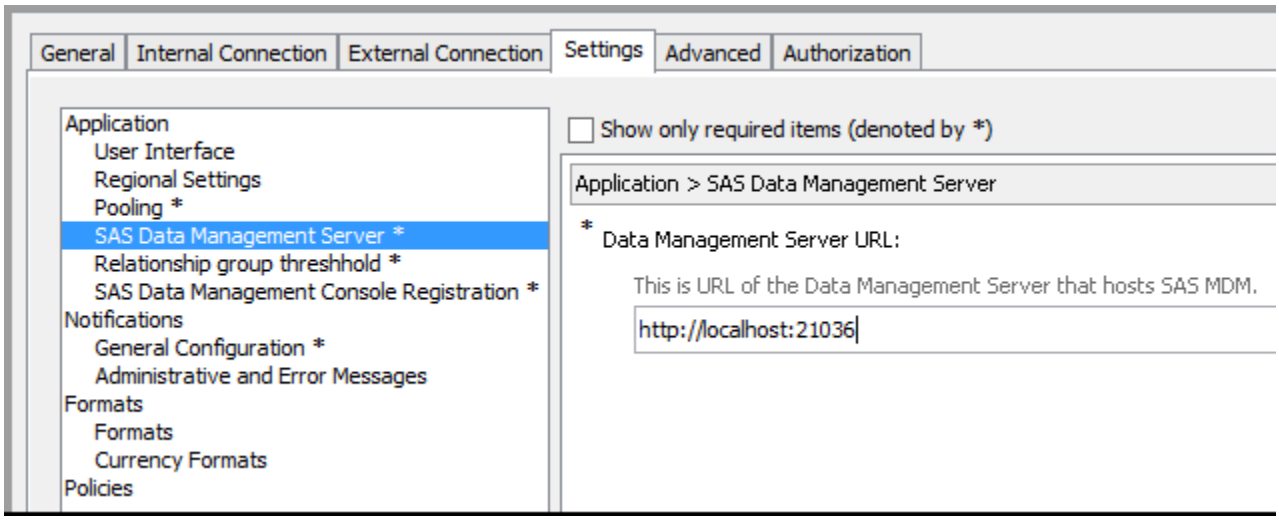
    c. Log on to the server as a member of the Data Management Administrators group and upload MDM Tag and MDM Lifecycle workflows to the SAS Workflow server. Ensure that you activate the workflows. The workflow template files can

be found in the repository (**<SAS MDM Install Directory>/workflows**) created during installation.

d. Enable e-mail notifications through Preferences found on the top right side in the SAS Web Administration Console.

12. If the SAS Deployment Wizard was not able to determine the location of the DataFlux Data Management Server instance that supports SAS MDM, you must set it in SAS Management Console. Navigate to **Application Management** ⇨ **Configuration Manager** ⇨ **SAS Application Infrastructure** and right-click **MDM 4.1**. Select **Properties** ⇨ **Settings** ⇨ **Application** ⇨ **DataFlux Data Management Server** and enter the correct information.

***Figure 4.1*** *SAS Management Console*




*Note:* You might need to restart your application server for this change to take effect. If you have trouble getting the SAS MDM web application to connect to DataFlux Data Management Server, verify that DataFlux Data Management Server is accessible through an IPV4 network address. DataFlux Data Management Server does not accept requests from an IPV6 network address.

13. You must verify and possibly modify the default connection information in the DataFlux Data Management Studio install_schema.djf job. Open this job with DataFlux Data Management Studio. The **Inputs** tab for the Echo node contains the defaults. Follow the instructions in install_schema.djf to make the necessary changes for Oracle and SQL Server. Other values, such as the database DSN and schema information, are taken from the appropriate, previously defined macro variables. See "Setting SAS MDM Macros" on page 22 for more information. After verifying the information, execute install_schema.djf to install the schema.

*Note:* Be aware of the following:

- If the scripts do not run, you might need to edit the command line used to call the database. Open install_schema.djf in DataFlux Data Management Studio and follow the instructions in the Expression nodes for your specific database.
- If your DataFlux Data Management Studio installation does not have access to SQL*Plus or sqlcmd, you might want to install the schema without using DataFlux Data Management Studio. You can use a database management tool such as PL/SQL Developer or SQL Developer. For example, if you are using an Oracle database, you can use SQL*Plus to execute the DDL for your database.

- If you do not have the correct database command-line access from your environment, an administrator using the correct user name or schema defined for SAS MDM must run the following scripts:

***Table 4.1*** *Scripts*

| Database | Scripts |
|---|---|
| Oracle | mdm_oracle_ddl.sql<br>mdm_oracle_procs.sql |
| SQL Server | mdm_sql_server_procs.sql<br>mdm_sql_server_ddl.sql |

By default, the scripts are located in **<SAS MDM Install Directory>\sasmdm\scripts**.

14. Run the add_default_meta.djf job to install the metadata. This job is located in the **<SAS MDM Install Directory>\batch_jobs\sasmdm** directory.
15. (Optional) Run the add_ui_language_meta_<locale>.djf job to install the translatable metadata. For example, to load the English metadata, use add_ui_language_meta_enusa.djf. These jobs are located in the **<SAS MDM Install Directory>\batch_jobs\sasmdm** directory.
16. (Optional) If you want to load sample entities and data into your SAS MDM database, run the add_sample_meta.djf job. These jobs are located in the **<SAS MDM Install Directory>\batch_jobs\sasmdm** directory.
17. (Optional) To load metadata for the sample entity types, run the add_ui_language_sample_<locale>.djf job. For example, to load the English sample metadata, use add_ui_language_sample_enusa.djf. These jobs are located in the **<SAS MDM Install Directory>\batch_jobs\sasmdm** directory.
18. To access SAS MDM and associated applications, open the SAS Data Management Console at **http://<host>:<port>/SASDataManagement**. Port 80 is the default port for Windows systems. Port 7980 is the default port for UNIX systems.
19. (Optional) To generate jobs that you can use to load and interact with sample data, perform the following tasks:

    *Note:* See the *SAS MDM User's Guide* for instructions about using the SAS MDM user interface.

    a. Open SAS Data Management Console.

    b. Click **Data Model**.

    c. On the **Data Model** tab, select the sample entity types into which you want to load data.

    d. Generate the associated jobs and services. Omit the generation of the **_stnd** jobs and mdm_extract_individual.ddf as they exist already and contain pre-defined data quality transformations necessary to support the sample entities.

    e. Download the add_update_<entity_type> jobs found in **batch_jobs/sasmdm** to the same directory in to DataFlux Data Management Studio. Download the mdm_survive_<entity_type>, mdm_extract_<entity_type>, and mdm_cluster_<entity_type> data services from the **data_services/sasmdm** directory to the same directory in DataFlux Data Management Studio.

f. Open each add_update job and modify the sample data location (EXTRACT_PARAMETER) in the initial Echo 1 step of the process job and save it. This must be only a filename and must include the extension. The path to sample data is represented in the MDM_SAMPLE_DATA macro. Sample data can be found in the **`<SAS MDM Repository>/samples/data directory`**.

g. Run the process job from DataFlux Data Management Studio or upload it and run it from DataFlux Data Management Server.

20. (Optional) To use the SAS Data Remediation component with SAS MDM, see the *SAS MDM User's Guide* for additional configuration options.

# Post-Update Tasks for SAS MDM Updates

To update from a previous version of SAS MDM, perform the following:

1. The **Priority** field in the data remediation API has been changed to **Importance**, and its values have changed from integers to the strings very low, low, medium, high, and critical. Because of this change, you must regenerate (or update) your entity-specific jobs:

   - mdm_extract_<entity>
   - mdm_survive_<entity>
   - mrm_add__<entity>

   You must manually move modifications that you have made to the standard generated output to the newly generated jobs. If the Remediation node is active in other SAS MDM jobs because of customizations, those jobs must be manually modified as well. If the Remediation node is used, you must modify these jobs through the Advanced Properties settings of the Remediation node.

2. Minor modifications have been made to the workflow templates used by SAS MDM to support the pass-through of URL data fromSAS MDM. You must manually upload the modified workflow templates to the workflow server using SAS Workflow Studio. See post-installation Step 11 on page 18 for more information about how to do this.

3. Minor modifications have been made to the job templates used by SAS MDM to support the provisioning of the MDM_VALID_TO_DTTM field to the mdm_survive_<entity_type> data services. If you require this support, regenerate the add_update_<entity_type>.djf batch job and the mdm_cluster_<entity_type>, mdm_survive_<entity_type>, mrm_add_<entity_type>, mrm_move_<entity_type>, and mrm_revert_move_<entity_type> data services.

4. Run the evolve_schema_41.djf batch job to schema-evolve the previous SAS MDM 4.1 schema. This job appears in the **`batch_jobs/sasmdm`** folder. Modify the echo node at the top of the job according to the instructions in the job.

5. New hierarchies and tool support are not automatically installed as part of the upgrade. You can either enter this sample data into the system using the web application or selectively run nodes from the add_sample_meta.djf job.

*Note:* After the installation is upgraded, all users of the Data Management Console must clear their browser caches.

# IPV6 and DataFlux Data Management Server

DataFlux Data Management Server does not support requests from IPV6 network addresses. If your environment resolves server network addresses (including localhost) to IPV6 format and does not default to IPV4 format upon failing to connect with IPV6, you must find the IPV4 network address for DataFlux Data Management Server.

Use the IPV4 address when you are specifying the host in client applications such as DataFlux Data Management Studio or in the configuration options of SAS MDM web applications in SAS Management Console. Use **`ipconfig`** on Windows or **`nslookup`** on UNIX to find the IPV4 address of the computer hosting DataFlux Data Management Server and use that value for connecting to the host from the other applications.

# Setting SAS MDM Macros

## *Overview*

SAS MDM jobs and services require certain values in the mdm_macros.cfg file on all clients and on the server. These values are automatically added during the SAS MDM installation. If you need to make changes to your mdm_macros.cfg file after the SAS MDM installation and configuration process, you can edit the mdm_macros.cfg file on your client computer using DataFlux Data Management Studio. Some of the paths used as values for the keys in this file must be modified for UNIX.

If you change macro values through the DataFlux Data Management Studio interface, an mdm_macros.cfg file is also created in **`\Users\<Your_User_ID>\AppData\Roaming\<Studio Install Directory>\<version>\etc\macros`**. You must combine those values with the values in the mdm_macros.cfg file in**`\Program Files (x86)\<Studio Install Directory>\<version>\etc\macros`** without duplicating key names. After making these changes, you must copy the combined file to the macros directory on DataFlux Data Management Server. If your DataFlux Data Management Server system runs on another platform, ensure that the path delimiters are appropriate for the platform.

If you run SAS MDM services on a DataFlux Data Management Server as web services, you must ensure that DataFlux Data Management Server has access to the mdm_macros.cfg file.

For more information about the configured macros and their use, see the mdm_macros.cfg file.

## *Database Values*

The values that you use for your database must be reflected in the mdm_macros.cfg file or files for your installation. For Oracle, the schema name must be in upper case letters.

SQL Server does not use packages for stored procedures, nor does it necessarily separate logon information from schemas. Modify the mdm_macros.cfg file or files with the following values:

```
MDM_HUB_DB=SQL SERVER
MDM_SCHEMA=mdm41
```

```
MDM_PACKAGE=
```

### *Macro Descriptions*

The following table lists the macros along with their default values and descriptions. When editing the mdm_macros.cfg file, do not set the macros MDM_CLUSTER_MEM, MDM_BRANCH_MEM, MDM_SORT_MEM, and MDM_SURVIVE_MEM to values lower than 4 MB or 4,194,304 bytes. In some cases, doing so can cause the various processes to stop and return memory-related errors.

For the macro MDM_REMEDIATION_PASSWORD, the value must first be encoded using a Base64 encoder before being added to the macro file. Many of these encoders are available on the Internet, but the encoder must be one that supports UTF-8 encoding.

To encode the password:

1. Enter your plain text password and then encode the password phrase using the Base64 encoder.
2. Take the resulting string of characters and paste it into the mdm_macros.cfg file. For example, if you enter *Bob Blake* into the encoder, **`Qm9iIEJsYWtl`** is returned.
3. Copy the encoded value into the MDM_REMEDIATION_PASSWORD macro.
4. Remediation files generated by both DataFlux Data Management Studio and DataFlux Data Management Server must be accessible to the web application. The MDM_REMEDIATION_FILE_DIR must be set to the same value for both DataFlux Data Management Studio and DataFlux Data Management Server. Both DataFlux Data Management Studio and DataFlux Data Management Server must have Read and Write access to the directory where these files are stored.

Here are the macro default values and descriptions:

***Table 4.2*** *Macros*

| Macro with Default | Description |
|---|---|
| MDM_DEF_TIMESTAMP_FORMAT<br>MM/DD/YYYY hh:mm:ss | Default timestamp format for the SAS MDM repository. |
| MDM_DEF_DATE_FORMAT<br>MM/DD/YYYY | Default date format for the SAS MDM repository. |
| MDM_BRANCH_MEM<br>4194304 | Branch memory limit. |
| MDM_SORT_MEM<br>16777216 | Sort memory limit. |
| MDM_CLUSTER_MEM<br>16777216 | Cluster memory limit. |
| MDM_SURVIVE_MEM<br>16 | Survivorship memory limit (MB). |

| Macro with Default | Description |
|---|---|
| MDM_MAX_ROWS<br>1000 | File input limit (primarily bulk jobs). |
| MDM_SORT_THREADS<br>4 | Sort thread limit. |
| MDM_MAX_CLUSTERS<br>1000 | Maximum number of entries returned by entity search. |
| MDM_QKB | The fully qualified path to the QKB. |
| MDM_ARCHIVE<br>C:\Program Files\SASHome<br>\DataFluxDataManagementServer\2.5\var\sasmdm<br>\archive\ | Output path from the archival job. |
| MDM_DATA<br>C:\Program Files\SASHome<br>\DataFluxDataManagementServer\2.5\var\sasmdm\data\ | Output path for bulk jobs. |
| MDM_LOGS<br>C:\Program Files\SASHome<br>\DataFluxDataManagementServer\2.5\var\sasmdm\logs\ | Output path for log files. |
| MDM_SCRIPTS<br>C:\Program Files\SASHome<br>\DataFluxDataManagementServer\2.5\share\sasmdm<br>\scripts\ | Path to the scripts and metadata directory. |
| MDM_SAMPLE_DATA<br>C:\Program Files\SASHome<br>\DataFluxDataManagementServer\2.5\share\sasmdm<br>\samples\data\ | The sample data directory. |
| MDM_SAMPLE_SCRIPTS<br>C:\Program Files\SASHome<br>\DataFluxDataManagementServer\2.5\share\sasmdm<br>\samples\scripts\ | The sample scripts directory. |
| MDM_HUB_DB<br>ORACLE | The database type: ORACLE or SQL SERVER. |
| MDM_DSN<br>DSN=SASMDM41 | The name of the DSN for the SAS MDM repository. |
| MDM_SCHEMA<br>MDM41 | The schema to use within the SAS MDM repository. This value should be all uppercase for Oracle. |

| Macro with Default | Description |
|---|---|
| MDM_PACKAGE<br>MD. | The package for the stored procedure. Leave this blank for SQL Server. |
| MDM_OS_DELIMITER<br>; | OS-specific delimiter. Use **;** for windows and **:** for UNIX platforms. |
| MDM_REMEDIATION_LIBS<br>C:\Program Files\SASHome \DataFluxDataManagementServer\2.5\lib\mdm\ | The location of the data remediation Java application. |
| MDM_REMEDIATION_FILE_DIR<br>C:\Program Files\SASHome \DataFluxDataManagementServer\2.5\var\sasmdm \remediation_data\ | The location where temporary data remediation data files are created. |
| MDM_REMEDIATION_USER | The user name that is used to access the SAS Data Remediation web service application. |
| MDM_REMEDIATION_PASSWORD | The base-64 encoded password that is used to access the SAS Data Remediation web service application. |
| MDM_REMEDIATION_URL<br>http://localhost:80/SASDataRemediation/rest/groups | The location of the SAS Data Remediation REST web service application. |

# Security Considerations for SAS MDM

## *Overview*

SAS MDM requires that security be enabled on DataFlux Data Management Server. The SAS MDM installation creates the following system settings for you:

```
dmserver/secure = yes
dmserver/secure/grp_admin = SASAdministrators
```

The installation process also defines several Data Management user groups for you:

- Data Management Administrators
- Data Management Stewards
- Data Management Business Users
- Data Management Business Approvers
- Data Management Power Users
- Data Management Executives

Use SAS Management Console to add users and groups to these groups or to define new ones for use with SAS MDM. Because SAS Administrators is set as the default group for grp_admin, you must assign at least one of your users to SAS Administrators so that

the assigned user can modify Access Control Lists and other permissions for your SAS MDM users.

Permissions consist of Group and User permissions and Access Control Lists (ACLs). Group and User permissions determine the actions that users are allowed to take on the server. ACLs control which users are allowed to access jobs on the servers. If Group permissions and ACLs are not configured, DataFlux Data Management Server provides defaults. DataFlux Data Management Studio is used to create permissions and ACLs for the groups, jobs, and services that are deployed on the DataFlux Data Management Server. The permissions that are used for the jobs depend on the roles and the groups that you have defined and on the needs of your business.

*Note:* Security checks are made by DataFlux Data Management Server when a SOAP request is received. When a job calls another job directly, DataFlux Data Management Server is not involved. When a job sends DataFlux Data Management Server a SOAP request to run another job using the real-time service node, the security check is made only at the top-level service

For more information about setting up security parameters for users and groups, see the *DataFlux Data Management Server Administrator's Guide*.

## Setting Permissions

### Group Permissions

On the **Data Management Servers** riser in DataFlux Data Management Studio, select the server to be configured. Use the **Security** tab to add groups to the server configuration, and modify the permissions appropriately.

The following are initial recommendations and can be changed as required for your installation:

***Table 4.3*** *Group Permissions*

| Group | Permissions |
|---|---|
| Data Management Administrators | All |
| Data Management Stewards | All |
| Data Management Business Users | All execute and list |
| Data Management Business Approvers | All execute and list |
| Data Management Power Users | All execute and list |
| Data Management Executives | All execute and list |

### ACLs

ACLs can be set up for each job or service to give access rights to individuals or groups. For example, assume that only a certain individual, or group of individuals in an organization, is allowed to change records within SAS MDM. The user name or associated group can be given special access rights through an ACL. For sensitive services, an administrator might want to deny everyone access using the ACL and then explicitly allow some users or a group to access or run the job or service. For more

information about setting up permission levels, see the *DataFlux Data Management Server Administrator's Guide*.

### Batch Job Permissions

On the **Data Management Servers** riser in DataFlux Data Management Studio, select the server to be configured. Expand the Batch Jobs folder and establish the following ACLs in the SAS MDM folder.

***Table 4.4*** *Batch Job Permissions*

| Group | Permissions |
|---|---|
| Data Management Administrators | All |
| Data Management Stewards | All |
| Data Management Business Users | None |
| Data Management Business Approvers | None |
| Data Management Power Users | None |
| Data Management Executives | None |

### Data Services Permissions

On the **Data Management Servers** riser in DataFlux Data Management Studio, select the server to be configured. Expand the Real-Time Data Services folder and establish the following ACLs in the SAS MDM folder.

***Table 4.5*** *Data Services Permissions*

| Group | Permissions |
|---|---|
| Data Management Administrators | All |
| Data Management Stewards | All |
| Data Management Business Users | All |
| Data Management Business Approvers | All |
| Data Management Power Users | All |
| Data Management Executives | All |

*Note:* You might want to make Data Management Stewards the owner of all SAS MDM jobs and services to facilitate maintenance activities.

### Enabling IP Address-Based Security

SAS MDM must have access to DataFlux Data Management Server in order to function. Requests from users using SAS MDM to run jobs appear to originate from the application server, not from the computers of the users. Configuring DataFlux Data Management Server IP address-based security does not secure your system against web users. Instead, you can use request filters to block IP addresses from accessing your application server. For more information, see the *DataFlux Data Management Server Administrator's Guide* and your application server documentation.

## SAS MDM Logging

### Installation Log

A log of the modifications made to your computer during installation is written to **\<SAS Home Directory>\InstallMisc\InstallLogs**.

### Database Script Logs

The database scripts that install or modify the schema generate log files in **<SAS MDM Install Directory>\sasmdm\logs**, as defined by the MDM_LOGS macro. These log files can be useful in determining problems during the initial configuration of the repository.

### Data Management Server and Studio Application Logs

Log messages that document DataFlux Data Management Server activity or job statistics while running jobs and services locally through DataFlux Data Management Studio or dfwfproc are recorded in log files specific to the software that you are running.

For more information about DataFlux Data Management Server and DataFlux Data Management Studio logging, see the *DataFlux Data Management Studio User's Guide* and the *DataFlux Data Management Server Administrator's Guide*.

### SAS MDM Web Application Logs

Logs for all SAS MDM web application components can be found on the application server that hosts them. They are typically found in a location similar to the following:

SAS MDM
**..\Config\Lev1\Web\Logs\SASServer13_1\sasmdm4.1.log**

SAS Task Manager
**..\Config\Lev1\Web\Logs\SASServer13_1\SASTaskManager2.1.log**

SAS Data Remediation
**..\Config\Lev1\Web\Logs**
**\SASServer13_1\SASDataRemediation2.1.log**

SAS Data Management Console
**..\Config\Lev1\Web\Logs**
**\SASServer13_1\SASDataManagement9.4.log**

Other logs that might be useful for troubleshooting purposes might not be on the same system as the SAS MDM web application components:

SAS Workflow

**..\Config\Lev1\Web\Logs**
**\SASServer1_1\SASWorkflowServices9.4.log**

SAS Metadata Server

**..\Config\Lev1\SASMeta\MetadataServer\Logs**

*Chapter 5*

# Localization Configuration



## Loading Localized Metadata

Localized metadata can be loaded into the hub so that entity names, attribute names, and other items are displayed based on the locale used to log on to SAS MDM. Two sample jobs are included that load localized metadata for French and German into **`<Repository>/batch_jobs/sasmdm`**.

## Enabling QKB Support for Other Locales

Each of the following services contains an embedded reference to the standardization job:

- mrm_add_<entity_type>.ddf
- add_update_<entity_type>.ddf
- mdm_query_survivor_<entity_type>.ddf
- mdm_query_relationship_<entity_type>.ddf

By default, the job supports ENUSA. To change to a single locale (for example, DEDEU), replace the references to the ENUSA locale with DEDEU. To support multiple locales simultaneously, add branches to the job for each locale. Ensure that the locales that you need are supported by the QKB and that the respective QKB licenses are correct. For more information, see the “Enabling Data Quality” section of the *SAS MDM User's Guide*.

# Bulk Loading Data for Other Locales

If you load data for other locales into your hub using database bulk-load utilities, ensure that the encoding in the add_update_<entity_type>.ddf jobs matches the codepage for your database. If these values do not match, some of the data might be translated into the database's codepage incorrectly.

The add_update_<entity_type>.ddf jobs interact with the DBMS bulk-load utilities by writing a data file representing your entities, and then invoking the bulk-load utilities with that data file as input. To change the encoding of the data file, open the **Properties** dialog box for the Write to Staging File node at the bottom of the job, and set the Encoding to a value that matches your database.

In addition to setting the output file encoding, you might find it necessary to change the control files used by the database utilities, which guide the interpretation of the file contents.

***Table 5.1*** *Database Control Files*

| Database | Relevant Files |
|---|---|
| Oracle | Look for the **`characterset`** parameter in .ctl files. |
| SQL Server | bcp files have a column collation field. |

For more information, see your database documentation.

*Chapter 6*

# Administrative Considerations



## Managing Database Growth

As records are created and updated within SAS MDM, the database continues to grow. Much of the growth is due to retired records, information that has been retained in the hub for historical purposes only. Depending on the needs of your organization, the retired records can be removed by transferring the data to offline storage or deleted entirely.

SAS MDM provides a sample job named mdm_archive_<entity_type>.djf that is suitable for customization. This job is not accessible through SAS MDM. You must run it manually. It is used to generate archive files for a specific entity type. The job finds expired records (including audit and entity records), generates files with those records, and removes them from the hub.

*Note:* Archiving data can result in extensive changes in the hub and might cause performance issues for other users. If possible, restrict archival activity to periods when the hub can be taken offline for maintenance. Alternatively, run archiving frequently enough that the overall operation is quick, or modify the archival jobs to process subsets of the hub.

# Server Connectivity

SAS MDM provides a set of jobs that can be run from SAS MDM, from DataFlux Data Management Studio, or from a command line using dfwfproc. Regardless of how you run these jobs, SAS MDM requires a DataFlux Data Management Server installation to run any real-time service nodes included in the jobs. The SAS MDM interface runs all of its jobs through DataFlux Data Management Server. When a job is run in DataFlux Data Management Studio or from a command line using dfwfproc, both the client and the server might be involved in the execution. This makes it very important that all jobs, macros, and database environments are synchronized on the client side of the DataFlux Data Management Studio environment and the server side of the DataFlux Data Management Server environment.

*Note:* It is important to remember the distributed nature of SAS MDM when diagnosing problems. Jobs that work locally but fail when run on the server, or jobs that fail to update data as expected, might indicate that jobs, macros, or database connections are not synchronized.

After you modify a job or service template, macro, metadata input file, or QKB element on a Windows computer, you must manually copy these files to your DataFlux Data Management Server environment.

# Oracle Considerations

### *Specifying MDM_SCHEMA*

For Oracle databases, the value of the MDM_SCHEMA macro must be specified with uppercase characters. Failure to specify MDM_SCHEMA in uppercase can cause some data to be inaccessible through DataFlux Data Management Studio, DataFlux Data Management Server, and SAS MDM.

### *Performance*

With Oracle 11g databases, you might experience poor performance during bulk loads, which can be improved by altering the operation of the optimizer.

To alter optimizer operation:

1. Log on as sys on your database.
2. Run: `ALTER SYSTEM SET OPTIMIZER_FEATURES_ENABLE = '10.2.0.1';`
3. Restart the database.

# SQL Server Considerations

By default, SQL Server searching is not case-sensitive. If you need to enable case-sensitive search capabilities, ensure that your database is initialized accordingly. For more details, see your database documentation.

# Diagnosing Problems

## Overview

The following sections provide answers to common questions that arise concerning SAS MDM installations.

## Server Connection

***Table 6.1*** *Server Connection Issues*

| Symptom | Resolution |
|---|---|
| On a 64-bit Windows system, the DSN has been set up, and the connection has been saved. The connection is accessible in DataFlux Data Management Studio, but not in DataFlux Data Management Server. | The 64-bit credentials must be saved explicitly for 64-bit systems. The connection must be replicated in a 64-bit ODBC connection, either by using the ODBC Administrative Tool in the Control Panel or by using the odbcad32.exe executable in `%systemdrive%\Windows\System32`. |

## Bulk Loads

***Table 6.2*** *Bulk Load Issues*

| Symptom | Resolution |
|---|---|
| A bulk load using a database utility (for example, sqlldr or bcp) does not work. | Check the log files for the run in **<SAS MDM Install Directory>\data**. Look at the timestamps to determine which logs might be relevant.<br>If the logs do not appear, or if they indicate errors running the database utility, try the following:<br>• Ensure that the database utility is installed and can be run. Try running it from the command line.<br>• Look at the entity_stage_bulk_loader.ddf job. If necessary, use alternative syntaxes to reference the database.<br>• Check the Echo node at the top of the add_update_<entity_type>.djf job, and ensure that the values are correct. |

## SAS MDM Logon

***Table 6.3*** *Logon Issues*

| Symptom | Resolution |
|---|---|
| Logging on to SAS MDM generates a message about the wrong number of types or arguments in a call to a stored procedure. | Ensure that your ODBC settings are configured correctly. The Procedure Return Results setting might not have been set properly. If you are running a 64-bit system, ensure that both the 32-bit driver used by DataFlux Data Management Studio, and the 64-bit driver used by DataFlux Data Management Server have the same settings. |
| Logging on to SAS MDM generates a message about not being able to load the model. | Check your application server and ensure that the data source has been configured correctly. For tcServer, check the configuration in the context.xml file and the availability of the JDBC drivers in the application server classpath. The application server uses context.xml to configure the resources required by the web application. The resource is a data source that provides connection information to the database for the workflow engine, which uses a JDBC connection pool instead of an ODBC DSN. |
| Not all areas of the SAS Data Management Console web application are visible. | The suggested minimum browser screen size for SAS MDM web components is 1024 x 768, but larger sizes are preferred. Smaller sizes might obscure parts of the application. |

| Symptom | Resolution |
| --- | --- |
| You cannot log on and the error log states that **`context initialization failed`** or that it is **`unable to find [jdbc]`**. | If the source directory used to specify the JDBC driver classpath during the SAS MDM installation contained any additional JAR files, you might have introduced extra JAR files into your web server environment. To correct the problem, remove the incorrect JAR files from the web application server **`lib`** directory (for example, ..\SAS \mdmadv_1machine\Lev1\Web\WebAppServer \SASServer13_1\lib) and restart the web server. When setting the JDBC driver classpath, the only file in the source directory should be the JDBC driver JAR file that matches your SAS MDM target database system. |

*Appendix 1*

# Data Model

# Database Tables and Columns

## *Overview*

This section includes a listing of database tables and a description of the columns for each table.

*Note:* The following data types are for an Oracle hub. Equivalent data types are used for a SQL Server hub.

## *Database Tables*

***Table A1.1*** *Database Tables*

| Table | Description |
|---|---|
| MDM_ATTRS | Contains a complete list of all attribute metadata for each entity type. |
| MDM_ATTR_DATATYPES | Specifies the constraining table for the available attribute data types. |
| MDM_ATTR_ACCESS | Specifies the access control table for attributes. |
| MDM_ATTR_DISPLAY | Contains display parameters for combinations of entity type and attribute type. |
| MDM_ATTR_PROFILES | Contains a list of localized attribute profile names. |
| MDM_CLUSTER_CONDITIONS | Maintains a list of cluster conditions for each entity type. |
| MDM_CLUSTER_CRITERIA | Maintains a list of attributes for each cluster condition. |
| MDM_CLUSTER_MAPPING | Maintains cluster collapse information. |
| MDM_CLUSTER_MEMBER_HISTORY | Maintains a history of cluster members for move and collapse scenarios. |
| MDM_ENTITY_CLUSTERS | Identifies clusters uniquely and the names by which they are known. |
| MDM_ENTITY_REL | Maintains authored relationships between clusters. |
| MDM_ENTITY_REL_CONDITIONS | Contains a list of conditions for entity type relationships. |

| Table | Description |
|---|---|
| MDM_ENTITY_REL_TYPES | Contains a list of relationships between entity types. |
| MDM_ENTITY_TYPES | Specifies the constraining table for the entity types that are covered in the hub, such as COMPANY or PART. |
| MDM_ENTITY_TYPE_ACCESS | Specifies the access control table for entity types. |
| MDM_ENTITY_TYPE_INDICES | Contains list of indices to be maintained on entity type specific tables such as MDM_<entity_type>_TT, MDM_<entity_type>_ST, MDM_<entity_type>_CC. |
| MDM_ENTITY_TYPE_TOOLS | Contains list of tools by entity type. |
| MDM_ERROR_LOG | Contains data about errors that occur within the stored procedures used by Master Data Management. |
| MDM_FORCED_CLUSTERS | Maintains a list of records that are moved from one cluster to another. |
| MDM_HIERARCHY_LEVELS | Contains hierarchy level details for hierarchy types. |
| MDM_HIERARCHY_TYPES | Contains hierarchy type associations with entity types. |
| MDM_NAMED_HIERARCHIES | Contains list of named hierarchies added by user. |
| MDM_LANGUAGES | Specifies the constraining table for the list of configured languages. |
| MDM_PROCESS_CONTROL | Specifies the control table to track load-update processes in the hub. |
| MDM_SRC_SYS | Specifies the registry table for all source systems that feed the hub. |
| MDM_STAGING | Stages non-user attribute data before making it live in the hub. |
| MDM_TOOL_ACCESS | Specifies access control for entity type tools. |

### *MDM_ATTRS Columns*

The MDM_ATTRS table contains a complete list of all attribute metadata for each entity type.

The information in the attribute definition is used to construct a column of the same name, with the data type, length (if any), and constraints as indicated by the metadata. In order for the entity type to be published successfully, the attribute name must be compatible with any restrictions imposed by the underlying database.

If you loaded localized metadata, the MDM_ATTRS table includes additional columns for the labels that appear in SAS MDM (MDM_DEF_ATTR_LABEL and MDM_ATTR_DESC). These column names are postfixed with the locale name. For example, if you loaded the French metadata, the table includes MDM_DEF_ATTR_LABEL_FRFRA and MDM_ATTR_DESC_FRFRA columns.

***Table A1.2*** *MDM_ATTRS Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ATTR_ID | INTEGER | NOT NULL | Unique attribute identifier. Automatically generated by SAS MDM. |
| MDM_OPSEQ | INTEGER | NULL | Sequence for optimistic locking. |
| MDM_ENTITY_TYPE_ID | INTEGER | NOT NULL | Attribute entity type identifier. Constrained by MDM_ENTITY_TYPES table. |
| MDM_ATTR_NAME | VARCHAR2(30) | NOT NULL | Descriptive attribute name. |
| MDM_ATTR_PROFILE_ID | INTEGER | NULL | Unique profile identifier. Constrained by MDM_ATTR_PROFILES table. |
| MDM_ATTR_DATATYPE_ID | INTEGER | NOT NULL | Attribute data type identifier. Constrained by MDM_ATTR_PROFILES table. |
| MDM_ATTR_LENGTH | INTEGER | NULL | Maximum size of the attribute. A nonzero value is required for string and list data types. |
| MDM_ATTR_CONSTRAINT | VARCHAR2(200) | NULL | Lists or regex strings used for validation of Master Data Management input fields used to populate the attribute. Can be blank. |
| MDM_REQUIRED | CHAR(1) | NOT NULL | Indicates whether the attribute is required for the data type with which it is associated. Enter **Y** to specify that the attribute is required. |
| MDM_VALID_FROM_DTTM | TIMESTAMP WITH LOCAL TIME ZONE | NOT NULL | Date and time at which the attribute is published for the first time. Automatically generated by SAS MDM. |

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_VALID_TO_DTTM | TIMESTAMP WITH LOCAL TIME ZONE | NOT NULL | Date and time at which the attribute is retired. Automatically generated by Master Data Management. |
| MDM_READONLY_FLAG | CHAR(1) | NOT NULL | Indicates whether the attribute can be modified from Master Data Management. Enter `Y` to specify that the attribute cannot be modified. |
| MDM_PARENT_ATTR_ID | INTEGER | NULL | Indicates parent ID for attribute relationships. |
| MDM_ENCRYPT_FLAG | CHAR(1) | NOT NULL | Indicates that the attribute should be encrypted in the transpose table. |

## *MDM_ATTR_ACCESS Columns*

The MDM_ATTR_ACCESS table is an access control table for attributes.

***Table A1.3*** *MDM_ATTR_ACCESS Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ATTR_GROUP_ID | INTEGER | NOT NULL | Unique identifier for attribute group. |
| MDM_ENTITY_TYPE_ID | INTEGER | NOT NULL | Unique entity type identifier. Constrained by MDM_ENTITY_TYPES table. |
| MDM_ATTR_ID | INTEGER | NOT NULL | Unique attribute identifier. Constrained by MDM_ATTRS table. |
| MDM_GROUP | NVARCHAR | NULL | Name of the access group. |
| MDM_ACTIVE | CHAR(1) | NOT NULL | |

## *MDM_ATTR_DATATYPES Columns*

The MDM_ATTR_DATATYPES table is a constraining table for the available attribute data types.

***Table A1.4*** *MDM_ATTR_DATATYPES Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ATTR_DATATYPE_ID | INTEGER | NOT NULL | Unique attribute data type identifier. |
| MDM_ATTR_DATATYPE | VARCHAR2(50) | NOT NULL | Name of the data type. |
| MDM_ATTR_DATATYPE_LABEL | NVARCHAR2(50) | NULL | Human-readable name of the data type. |
| MDM_DB_DATATYPE | VARCHAR2(50) | NOT NULL | DBMS-specific data type for the data type. For example, string attributes are created as VARCHAR columns in SQL Server, but are created as VARCHAR2 columns in Oracle. |

## *MDM_ATTR_DISPLAY Columns*

The MDM_ATTR_DISPLAY table contains display parameters for combinations of entity type and attribute type.

***Table A1.5*** *MDM_ATTR_DISPLAY Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ENTITY_TYPE_ID | INTEGER | NOT NULL | Attribute entity type identifier. Constrained by MDM_ENTITY_TYPES table. |
| MDM_ATTR_ID | INTEGER | NOT NULL | Attribute identifier. Constrained by MDM_ATTRS table. |
| MDM_SEARCH_DISPLAY_ORDER | INTEGER | NULL | Display order of the attribute in the search form on the **Master Data Management** tab. Enter 0 (zero) to hide the attribute in the **Master Data Management** tab. |
| MDM_TABLE_DISPLAY_ORDER | INTEGER | NULL | Attribute order for displaying search results. |
| MDM_CLUSTER_DISPLAY_ORDER | INTEGER | NULL | Display order of the attribute on the **Cluster Members** tab of the Master Data Management entity editor. |

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_FORM_DISPLAY_ORDER | INTEGER | NULL | Display order of the attribute in the Master Data Management entity editor. |
| MDM_PREVIEW_DISPLAY_ORDER | INTEGER | NULL | Display order of the attribute in the Master Data Management Details panels. |

### MDM_ATTR_PROFILES Columns

The MDM_ATTR_PROFILES table contains a list of localized attribute profile names.

If you have loaded localized metadata, the MDM_ATTR_PROFILES table includes additional columns for the labels that appear in SAS MDM (MDM_ATTR_PROFILE_NAME). These column names are postfixed with the locale name. For example, if you have loaded the French metadata, the table includes an MDM_ATTR_PROFILE_NAME_FRFRA column.

***Table A1.6*** *MDM_ATTR_PROFILES Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ATTR_PROFILE_ID | INTEGER | NOT NULL | Unique profile identifier. Automatically generated by SAS MDM. |

### MDM_CLUSTER_CONDITIONS Columns

The MDM_CLUSTER_CONDITIONS table maintains a list of cluster conditions for each entity type.

***Table A1.7*** *MDM_CLUSTER_CONDITIONS Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_CLUSTER_CONDITION_ID | INTEGER | NOT NULL | Unique cluster condition identifier. |
| MDM_ENTITY_TYPE_ID | INTEGER | NOT NULL | Cluster condition entity type identifier. Constrained by MDM_ENTITY_TYPES table. |
| MDM_SORT_ORDER | INTEGER | NOT NULL | Sort order of cluster condition for an entity type. |

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_VALID_FROM_DTTM | TIMESTAMP WITH LOCAL TIME ZONE | NULL | Start of validity for cluster condition. |
| MDM_VALID_TO_DTTM | TIMESTAMP WITH LOCAL TIME ZONE | NULL | End of validity for cluster condition. |

### *MDM_CLUSTER_CRITERIA Columns*

The MDM_CLUSTER_CRITERIA table maintains a list of attributes for each cluster condition.

***Table A1.8*** *MDM_CLUSTER_CRITERIA Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_CLUSTER_CONDITION_ID | INTEGER | NOT NULL | Unique cluster condition identifier.<br>Constrained by MDM_CLUSTER_CONDITIONS table. |
| MDM_ATTR_ID | INTEGER | NOT NULL | Cluster condition attribute identifier. Constrained by MDM_ATTRS table. |

### *MDM_CLUSTER_MAPPING Columns*

The MDM_CLUSTER_MAPPING table maintains cluster collapse information.

***Table A1.9*** *MDM_CLUSTER_MAPPING Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| ORIGINAL_CLUSTER_ID | INTEGER | NOT NULL | Unique cluster identifier that is retired in the process of collapse. |
| CURRENT_CLUSTER_ID | INTEGER | NOT NULL | Unique cluster identifier into which the records are collapsed. |

### *MDM_CLUSTER_MEMBER_HISTORY Columns*

The MDM_CLUSTER_MEMBER_HISTORY table maintains a history of cluster members for move and collapse scenarios.

***Table A1.10*** *MDM_CLUSTER_MEMBER_HISTORY Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ENTITY_CLUSTER_ID | INTEGER | NOT NULL | Unique cluster identifier to which the entity ID belonged. |
| MDM_ENTITY_ID | INTEGER | NOT NULL | Unique entity identifier. |
| MDM_COMMENT | NVARCHAR2(200) | NULL | Comment on the history of the record. |
| MDM_REASON | CHAR(1) | NOT NULL | Reason code: C for cluster collapse, or F for moved records. |
| MDM_MODIFIED_DTTM | TIMESTAMP WITH LOCAL TIME ZONE | NOT NULL | Date and time of modification for the record. |
| MDM_VALID_TO_DTTM | TIMESTAMP WITH LOCAL TIME ZONE | NULL | End of validity. |

## *MDM_ENTITY_CLUSTERS Columns*

The MDM_ENTITY_CLUSTERS table uniquely identifies clusters and the names by which they are known.

***Table A1.11*** *MDM_ENTITY_CLUSTERS Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ENTITY_CLUSTER_ID | INTEGER | NOT NULL | Unique entity cluster identifier. |
| MDM_ENTITY_CLUSTER_NAME | NVARCHAR2(100) | NOT NULL | Entity cluster name. |
| MDM_CLUSTER_TYPE | CHAR(1) | NOT NULL | Cluster type: natural or forced. Natural clusters are constructed solely by the clustering rules expressed in the jobs. Forced clusters are generated by users moving records into the cluster. |

## *MDM_ENTITY_REL Columns*

The MDM_ENTITY_REL table maintains authored relationships between clusters.

*Table A1.12 MDM_ENTITY_REL Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ENTITY_REL_TYPE_ID | INTEGER | NOT NULL | Entity type relationship identifier. Constrained by MDM_ENTITY_REL_TYPES table. |
| MDM_FROM_CLUSTER_ID | INTEGER | NOT NULL | Unique cluster identifier. Constrained by MDM_ENTITY_CLUSTERS table. |
| MDM_TO_CLUSTER_ID | INTEGER | NOT NULL | Unique cluster identifier. Constrained by MDM_ENTITY_CLUSTERS table. |
| MDM_VALID_FROM_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NOT NULL | Start of validity for relationship. |
| MDM_VALID_TO_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NOT NULL | End of validity for relationship. |

## *MDM_ENTITY_REL_CONDITIONS Columns*

The MDM_ENTITY_REL_CONDITIONS table contains a list of conditions for entity type relationships.

*Table A1.13 MDM_ENTITY_REL_CONDITIONS Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ENTITY_REL_TYPE_ID | INTEGER | NOT NULL | Entity type relationship identifier. Constrained by MDM_ENTITY_REL_TYPES table. |
| MDM_ENTITY_REL_CONDITION_ID | INTEGER | NOT NULL | Unique entity relationship condition identifier. |
| MDM_FROM_ATTR_ID | INTEGER | NOT NULL | Unique attribute identifier. Constrained by MDM_ATTRS table. |
| MDM_TO_ATTR_ID | INTEGER | NOT NULL | Unique attribute identifier. Constrained by MDM_ATTRS table. |

## *MDM_ENTITY_REL_TYPES Columns*

The MDM_ENTITY_REL_TYPES table contains a list of relationships between entity types.

If you have loaded localized metadata (see "Localization Configuration" on page 31), the MDM_ENTITY_REL_TYPES table includes additional columns for the labels that appear in SAS MDM (MDM_REL_TYPE_DESC, MDM_REL_TYPE_LABEL, and MDM_INV_REL_TYPE_LABEL). These column names are postfixed with the locale name of the localized metadata. For example, if you have loaded the French metadata, the table includes MDM_REL_TYPE_DESC_FRFRA, MDM_REL_TYPE_LABEL_FRFRA, and MDM_INV_RELTYPE_LABEL_FRFRA columns.

***Table A1.14*** *MDM_ENTITY_REL_TYPES Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ REL_TYPE_ID | INTEGER | NOT NULL | Entity type relationship identifier. |
| MDM_REL_TYPE | VARCHAR2(50) | NOT NULL | Entity type relationship name. |
| MDM_SORT_ORDER | INTEGER | NULL | Sort order of relationship type. |
| MDM_INVERSE_SORT_ORDER | INTEGER | NULL | Inverse sort order of relationship type. |
| MDM_FROM_ENTITY_TYPE_ID | INTEGER | NOT NULL | Unique entity type identifier. Constrained by MDM_ENTITY_TYPES table. |
| MDM_TO_ENTITY_TYPE_ID | INTEGER | NOT NULL | Unique entity type identifier. Constrained by MDM_ENTITY_TYPES table. |
| MDM_VALID_FROM_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | Start of validity for relationship type. |
| MDM_VALID_TO_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | End of validity for relationship type. |

## *MDM_ENTITY_TYPES Columns*

The MDM_ENTITY_TYPES table is a constraining table for the entity types that are covered in the hub, such as COMPANY or PART.

If you have loaded localized metadata (see "Localization Configuration" on page 31), the MDM_ENTITY_TYPES table includes additional columns for the labels that appear in SAS MDM (MDM_ENTITY_TYPE_LABEL and MDM_ENTITY_TYPE_DESC). These column names are postfixed with the locale name. For example, if you have loaded the French metadata, the table includes MDM_ENTITY_TYPE_LABEL_FRFRA and MDM_ENTITY_TYPE_DESC_FRFRA columns.

***Table A1.15*** *MDM_ENTITY_TYPES Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ENTITY_TYPE_ID | INTEGER | NOT NULL | Unique entity type identifier. Automatically generated by Master Data Management. |
| MDM_QPSEQ | INTEGER | NULL | Sequence for optimistic locking. |
| MDM_ENTITY_TYPE | VARCHAR2(23) | NOT NULL | Entity type name (for example, COMPANY or PART). |
| MDM_ENTITY_TYPE_SORT_ORDER | INTEGER | NOT NULL | Controls the entity type's sort order in Master Data Management selection lists. Enter zero (0) to hide the entity type in the Master Data Management. |
| MDM_LAST_MOD_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | Date and time at which the entity type was last modified. Automatically generated by Master Data Management. Use this attribute in combination with MDM_GOLIVE_DTTM to determine whether the metadata has changes that have not been applied to the hub. |
| MDM_GOLIVE_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | Date and time at which entity type is made live. Automatically generated by Master Data Management. |
| MDM_VALID_FROM_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | Date and time at which entity type is first published. Automatically generated by Master Data Management. |
| MDM_VALID_TO_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | Date and time at which entity type is retired. Automatically generated by Master Data Management. |
| MDM_PARENT_ID | INTEGER | NULL | Unique tree identifier of the parent. Constrained by MDM_ENTITY_TYPES table. |
| MDM_ABSTRACT | CHAR(1) | NOT NULL | Flag indicating if the entity type is abstract or not. |

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_LABEL_ATTR_ID | NUMBER | No value | Identifier of the label attribute for this entity type. |

### MDM_ENTITY_TYPE_ACCESS Columns

The MDM_ENTITY_TYPE_ACCESS table contains access control list for entity types.

***Table A1.16*** *MDM_ENTITY_TYPE_ACCESS*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ENTITY_GROUP_ID | INTEGER | NOT NULL | Entity type group identifier. |
| MDM_ENTITY_TYPE_ID | INTEGER | NOT NULL | Entity type identifier. Constrained by MDM_ENTITY_TYPES table. |
| MDM_GROUP | NVARCHAR | NULL | Group name. |
| MDM_ACTIVE | INTEGER | NOT NULL | Entity type access active indicator. |

### MDM_ENTITY_TYPE_INDICES Columns

The MDM_ENTITY_TYPE_INDICES table contains the list of custom indices created on ST, TT, and CC tables.

***Table A1.17*** *MDM_ENTITY_TYPE_INDICES*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ENTITY_TYPE | VARCHAR2(23) | NOT NULL | Entity type name. |
| MDM_ATTR_NAME | VARCHAR2(30) | NOT NULL | Attribute name. |
| MDM_INDEX_ID | INTEGER | NOT NULL | Index identifier for entity type. |
| MDM_INDEX_COLUMN_ORDER | INTEGER | NOT NULL | Order of column in the index. |

### MDM_ENTITY_TYPE_TOOLS Columns

The MDM_ENTITY_TYPE_TOOLS table contains the list of tools for each entity type.

***Table A1.18*** *MDM_ENTITY_TYPE_TOOLS*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| TOOL_ID | INTEGER | NOT NULL | Tool identifier. |
| MDM_ENTITY_TYPE_ID | INTEGER | NOT NULL | Entity type identifier. |
| TOOL | VARCHAR2(260) | NOT NULL | Tool name. |
| TOOL_SORT_ORDER | INTEGER | NOT NULL | Sort order for tool. |
| TOOLTYPE | CHAR(1) | NOT NULL | Type identifier for the tool. **D** for data job, **P** for process job, and **B** for batch job. |
| CARDINALITY | CHAR(1) | NOT NULL | Expected input cardinality (S-M-L). |
| RESULTTYPE | CHAR(1) | NOT NULL | Type of result being returned. Valid result types are Status (S), Batch status (B), URL (U), Table (T), and Attribute value (A). |
| CONTACT | NVARCHAR2(513) | NULL | Contact for the tool. |
| MDM_VALID_FROM_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | Start of validity for the tool. |
| MDM_VALID_TO_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | End of validity for the tool. |

## *MDM_ERROR_LOG Columns*

The MDM_ERROR_LOG table contains data about errors that occur within the stored procedures used by Master Data Management.

***Table A1.19*** *MDM_ERROR_LOG Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_ERROR_LOG_ID | INTEGER | NOT NULL | Unique error log record identifier. |
| TABLE_NAME | VARCHAR2(50) | NULL | Name of table in which error occurred. |
| SERVICE_NAME | VARCHAR2(50) | NULL | Name of service in which error occurred. |
| RECORD_ID | INTEGER | NOT NULL | Identifier of record in which error occurred. |

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| RECORD_ID_FROM | VARCHAR2(30) | NULL | Identifier of record from which error was generated. |
| ERROR_SOURCE | VARCHAR2(100) | NOT NULL | Name of source (service, stored procedure, window) where error originated. |
| ERROR_CODE | VARCHAR(10) | NOT NULL | Database-generated error code. |
| ERROR_MESSAGE | VARCHAR2(1000) | NULL | Database-generated error message. |
| ERROR_DATE | TIMESTAMP WITH LOCAL TIMEZONE | NOT NULL | Date when error log record was added. |

### *MDM_FORCED_CLUSTERS Columns*

The MDM_FORCED_CLUSTERS table maintains list of records that are moved from one cluster to another.

***Table A1.20*** *MDM_FORCED_CLUSTERS Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| ORIGINAL_CLUSTER_ID | INTEGER | NOT NULL | Unique cluster identifier. Constrained by MDM_ENTITY_CLUSTERS table. |
| SRC_SYS_ID | INTEGER | NOT NULL | Unique source system identifier. |
| SRC_SYS_REC_ID | VARCHAR2(200) | NOT NULL | Source system record identifier. |
| ENTITY_TYPE_ID | INTEGER | NOT NULL | Unique entity type identifier. Constrained by MDM_ENTITY_TYPES table. |
| CURRENT_CLUSTER_ID | INTEGER | NOT NULL | Unique cluster identifier. Constrained by MDM_ENTITY_CLUSTERS table. |

### *MDM_HIERARCHY_LEVELS Columns*

The MDM_HIERARCHY_LEVELS table contains the list of levels for each hierarchy type.

***Table A1.21*** *MDM_HIERARCHY_LEVELS Columns*

| Name | Data type | Null Option | Description |
|---|---|---|---|
| MDM_HIERARCHY_LEVEL_ID | INTEGER | NOT NULL | Hierarchy level identifier. |

| Name | Data type | Null Option | Description |
|---|---|---|---|
| MDM_HIERARCHY_TYPE_ID | INTEGER | NOT NULL | Hierarchy type identifier. Constrained by MDM_HIERARCHY_TYPES table. |
| MDM_REL_TYPE_ID | INTEGER | NOT NULL | Relationship type identifier. Constrained by MDM_ENTITY_REL_TYPES table. |
| MDM_REVERSE_REL_DIRECTION | CHAR(1) | NULL | Flag to reverse relationship direction. |
| MDM_PARENT_LEVEL_ID | INTEGER | NULL | Parent hierarchy level identifier. |

## *MDM_HIERARCHY_TYPES Columns*

The MDM_HIERARCHY_TYPES table is a constraining table denoting specific hierarchy types.

If you have loaded localized metadata (see "Localization Configuration" on page 31), the MDM_HIERARCHY_TYPES table includes additional columns for the labels that appear in SAS MDM (MDM_HIER_TYPE_LABEL and MDM_HIER_TYPE_DESC). These column names are postfixed with the locale name of the localized metadata. For example, if you have loaded the French metadata, the table includes MDM_HIER_TYPE_LABEL_FRFRA, and MDM_HIER_TYPE_DESC_FRFRA columns.

***Table A1.22*** *MDM_HIERARCHY_TYPES Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_HIERARCHY_TYPE_ID | INTEGER | NOT NULL | Unique hierarchy type identifier. |
| MDM_OPSEQ | INTEGER | NULL | Sequence for optimistic locking. |
| MDM_HIERARCHY_TYPE | VARCHAR2(50) | NOT NULL | Hierarchy type designation. |
| MDM_VALID_FROM_DTTM | DATE | NULL | Start of validity for hierarchy type. |
| MDM_VALID_TO_DTTM | DATE | NULL | End of validity for hierarchy type. |

## *MDM_NAMED_HIERARCHIES Columns*

The MDM_NAMED_HIERARCHIES table contains the list of named hierarchies added by users.

**Table A1.23** MDM_NAMED_HIERARCHIES Columns

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_HIERARCHY_TYPE_ID | INTEGER | NOT NULL | Hierarchy type identifier. Constrained by MDM_HIERARCHY_TYPES table. |
| MDM_ENTITY_CLUSTER_ID | INTEGER | NULL | Cluster identifier. Constrained by MDM_ENTITY_CLUSTERS table. |
| MDM_HIERARCHY_NAME | NVARCHAR2(50) | NOT NULL | Hierarchy name. |
| MDM_HIERARCHY_DESC | NVARCHAR2(50) | NULL | Hierarchy name description. |

## MDM_LANGUAGES Columns

The MDM_LANGUAGES table is a constraining table for the list of configured languages.

**Table A1.24** MDM_LANGUAGES Columns

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_LANGUAGE_ID | INTEGER | NOT NULL | Unique country code identifier. |
| MDM_OPSEQ | INTEGER | NULL | Sequence for optimistic locking. |
| MDM_LANGUAGE_NAME | NVARCHAR2(50) | NULL | Language name. |
| MDM_QKB_LOCALE | VARCHAR2(5) | NOT NULL | QKB locale associated with language. |
| MDM_ISO_LOCALE | VARCHAR2(5) | NOT NULL | ISO locale associated with language. |
| MDM_IS_DEFAULT | CHAR(1) | NOT NULL | Flag to indicate default language. |
| MDM_VALID_FROM_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | Start of validity for country code. |
| MDM_VALID_TO_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | End of validity for country code. |

## MDM_PROCESS_CONTROL Columns

The MDM_PROCESS_CONTROL table is a control table to track load-update processes in the hub.

***Table A1.25*** *MDM_PROCESS_CONTROL Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| PROCESS_ID | INTEGER | NOT NULL | Unique process identifier. |
| MDM_STAGING_ID | INTEGER | NULL | Unique staging identifier. |
| PROCESS_NAME | VARCHAR2(50) | NOT NULL | Process name. |
| MDM_ENTITY_TYPE_ID | INTEGER | NOT NULL | Unique entity type identifier. Constrained by MDM_ENTITY_TYPES table. |
| PROCESS_RUNNING | CHAR(1) | NOT NULL | Flag to indicate whether process is currently running. |
| PROCESS_SUCCESS | CHAR(1) | NULL | Flag to indicate success or failure of process. |
| PROCESS_USER | NVARCHAR2(50) | NULL | User who initiated process. |
| PROCESS_START_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NOT NULL | Start date and time of process. |
| PROCESS_END_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | End date and time of process. |

## *MDM_SRC_SYS Columns*

The MDM_SRC_SYS table is a registry table for all source systems that feed the hub.

***Table A1.26*** *MDM_SRC_SYS Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_SRC_SYS_ID | INTEGER | NOT NULL | Unique identifier of source system. |
| MDM_OPSEQ | INTEGER | NULL | Sequence for optimistic locking. |
| MDM_SRC_SYS_NAME | NVARCHAR2(50) | NOT NULL | Name of source system. |
| MDM_SRC_SYS_DESC | NVARCHAR2(100) | NULL | Description of source system. |
| MDM_VALID_FROM_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | Start of validity for source system. |
| MDM_VALID_TO_DTTM | TIMESTAMP WITH LOCAL TIMEZONE | NULL | End of validity for source system. |

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_SRC_SYS_PATH | VARCHAR(500) | NULL | Identifies the original source system in a hub. For source systems that were not imported from a hub, this value is not populated. |

### *MDM_STAGING Columns*

The MDM_STAGING table stages non-user attribute data before making it live in the hub.

***Table A1.27*** *MDM_STAGING Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_STAGING_ID | INTEGER | NOT NULL | Unique staging identifier. |
| MDM_ENTITY_ID | INTEGER | NOT NULL | Unique record identifier. |
| MDM_ENTITY_CLUSTER_ID | INTEGER | NULL | Unique cluster identifier. |
| MDM_ENTITY_CLUSTER_NAME | NVARCHAR2(100) | NULL | Cluster name. |
| MATCH_TYPE | INTEGER | NULL | Match type of the incoming record. |
| OLD_CLUSTER_ID | INTEGER | NULL | Old CLUSTER ID if the record is migrating to a new cluster. |
| ROW_PROCESSED | CHAR(1) | NOT NULL | Flag indicating whether the record is processed. |

## Entity Tables

### *Overview*

The following tables are generated from the metadata for the specific entity type. As attributes for an entity type are defined, columns are added to the tables. The tables that are created are not automatically removed as entity types are retired.

## Sample Entity Tables

***Table A1.28*** *Sample Entity Tables*

| Name | Naming Convention | Description |
|---|---|---|
| Staging Table | MDM_<entity_type>_ST | Table used to prepare data before updating the hub. |
| Transpose Table | MDM_<entity_type>_TT | Table used to maintain record data for particular entity types. |
| Cluster Conditions Table | MDM_<entity_type>_CC | Table used to pre-load cluster condition data columns during a data load. |
| Audit Log Table | MDM_<entity_type>_AL | Table to maintain audit trail of changes to hub data. |

The following columns are common to all the staging tables and transpose tables. These tables are further populated with columns representing the attributes defined for the entity type.

## MDM_<Entity_Type>_ST Columns

The staging tables are schema-evolved to include columns for the attributes defined by the related entity type.

***Table A1.29*** *MDM_<Entity_Type>_ST Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_STAGING_ID | INTEGER | NOT NULL | Unique staging identifier. |
| ROW_ID | INTEGER | NOT NULL | Identifier for row within each staging ID. |
| MDM_ENTITY_CLUSTER_ID | INTEGER | NULL | Unique cluster identifier. |
| MDM_ENTITY_CLUSTER_NAME | NVARCHAR2(100) | NOT NULL | Cluster name. |
| MDM_SURVIVOR | CHAR(1) | NOT NULL | Flag indicating if the record is a survivor record. |
| MATCH_TYPE | INTEGER | NOT NULL | Match type of the incoming record. |
| CID | INTEGER | NOT NULL | Intermittent cluster ID. |

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| ROW_PROCESSED | CHAR(1) | NOT NULL | Flag indicating if the record is processed or not. |
| MDM_SRC_SYS_ID | INTEGER | NOT NULL | Unique source system identifier. |
| MDM__SRC_SYS_REC_ID | NVARCHAR2(200) | NULL | Source system record identifier. |
| MDM_LANGUAGE_ID | INTEGER | NOT NULL | Unique language identifier. |

### *MDM_<Entity_Type>_TT Columns*

The transpose tables are schema-evolved to include columns for the attributes defined by the related entity type.

***Table A1.30*** *MDM_<Entity_Type>_TT Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_STAGING_ID | INTEGER | NOT NULL | Unique staging identifier. |
| MDM_ENTITY_ID | INTEGER | NOT NULL | Unique entity identifier. |
| MDM_SRC_SYS_ID | INTEGER | NOT NULL | Unique source system identifier. |
| MDM_SRC_SYS_REC_ID | NVARCHAR2(200) | NULL | Source system record identifier. |
| MDM_LANGUAGE_ID | INTEGER | NOT NULL | Unique language identifier. |
| MDM_ENTITY_CLUSTER_ID | INTEGER | NOT NULL | Unique cluster identifier. |
| MDM_SURVIVOR | CHAR(1) | NOT NULL | Flag indicating if the record is a survivor record. |
| MDM_VALID_FROM_DTTM | TIMESTAMP WITH LOCAL TIME ZONE | NULL | Start of validity for entity. |
| MDM_VALID_TO_DTTM | TIMESTAMP WITH LOCAL TIME ZONE | NULL | End of validity for entity. |

### MDM_<Entity_Type>_CC Columns

The cluster conditions tables are schema-evolved to include columns for the attributes defined by the related entity type.

***Table A1.31*** *MDM_<Entity_Type>_CC Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_STAGING_ID | INTEGER | NOT NULL | Unique staging identifier. |
| ROW_ID | INTEGER | NOT NULL | Unique row identifier. |

### MDM_<Entity_Type>_AL Columns

***Table A1.32*** *MDM_<Entity_Type>_AL Columns*

| Name | Data Type | Null Option | Description |
|---|---|---|---|
| MDM_AUDIT_LOG_ID | INTEGER | NOT NULL | Unique audit log identifier. |
| TABLE_NAME | VARCHAR2(50) | NOT NULL | Name of the table in which the auditable action was made. |
| RECORD_ID | INTEGER | NOT NULL | Identifier of the record for which the auditable action was made. |
| USER_ACTION | VARCHAR2(10) | NOT NULL | Type of action. |
| MDM_MODIFIED_BY | NVARCHAR2(513) | NOT NULL | ID of the user who performed the auditable action. |
| MODIFIED_DTTM | TIMESTAMP WITH LOCAL TIME ZONE | NOT NULL | Date and time at which the auditable action occurred. |

# Bulk Load Tables

## *Overview*

The following tables are generated while running bulk load jobs to stage data during the load. They are generated from metadata for the specific entity type. As attributes for an entity type are defined, columns are added to the tables. The tables are automatically deleted at the end of a successful load. If the load fails for any reason, the tables are not deleted and can be used for debugging purposes.

## *Sample Bulk Load Tables*

***Table A1.33*** *Sample Bulk Load Tables*

| Name | Naming Convention | Description |
|---|---|---|
| Cluster Conditions Table | MDM_BULK_CC_<staging_id> | Table used to pre-load cluster condition data columns during a data load. |
| Entity Staging Table | MDM_BULK_ST_<staging_id> | Table used to stage data before updating the hub. |
| Staging Table | MDM_STAGING_<staging_id> | Table used to stage data before updating the hub. |

The following columns are common to all the staging tables and transpose tables. These tables are further populated with columns representing the attributes defined for the entity type.

## *MDM_BULK_CC_<Staging_ID> Columns*

These tables have the same structure as the MDM_<entity_type>_CC table for the entity type associated with staging ID of the particular load.

## *MDM_BULK_ST_<Staging_ID> Columns*

These tables have the same structure as the MDM_<entity_type>_ST table for the entity type associated with staging ID of the particular load.

## *MDM_STAGING_<Staging_ID> Columns*

These tables have the same structure as the MDM_STAGING table.

# Sample Views

The following tables contain available SAS MDM views.

***Table A1.34*** *Sample Bulk Load Tables*

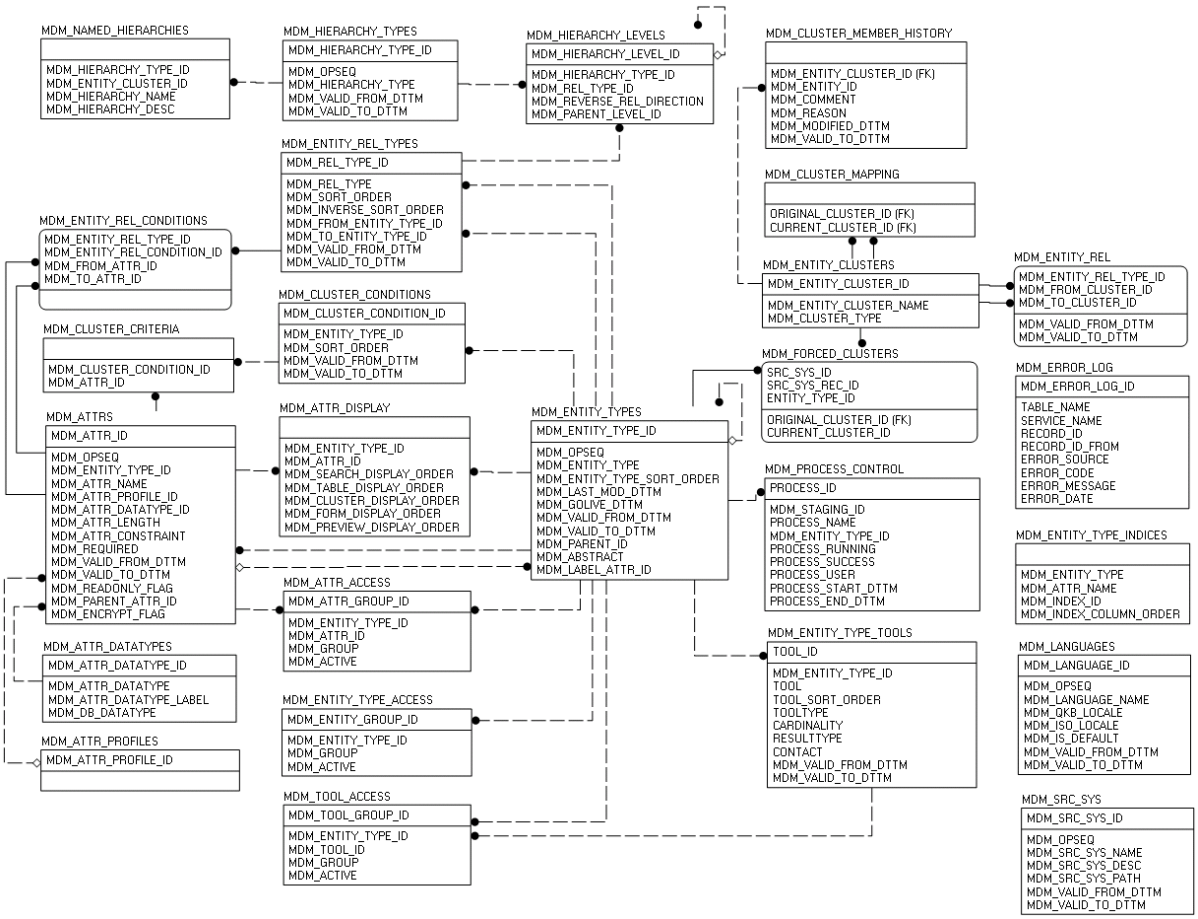
| View | Naming Convention | Description |
|---|---|---|
| Match View | MDM_<entity_type>_MA | Transpose view with only contributor records. Primarily used by clustering jobs to find matched entity rows or records prior to clustering. |
| Cluster View | MDM_<entity_type>_CV | View used in clustering operations. |
| Search View | MDM_<entity_type>_SV | Search view for a given entity type. Primarily used to incorporate entity type inheritance into searches. |

# Data Model Diagrams

## *Core Data Model Diagram*

The following diagram illustrates the core tables in the SAS MDM data model. These tables are not dependent on the entity types defined in the hub.

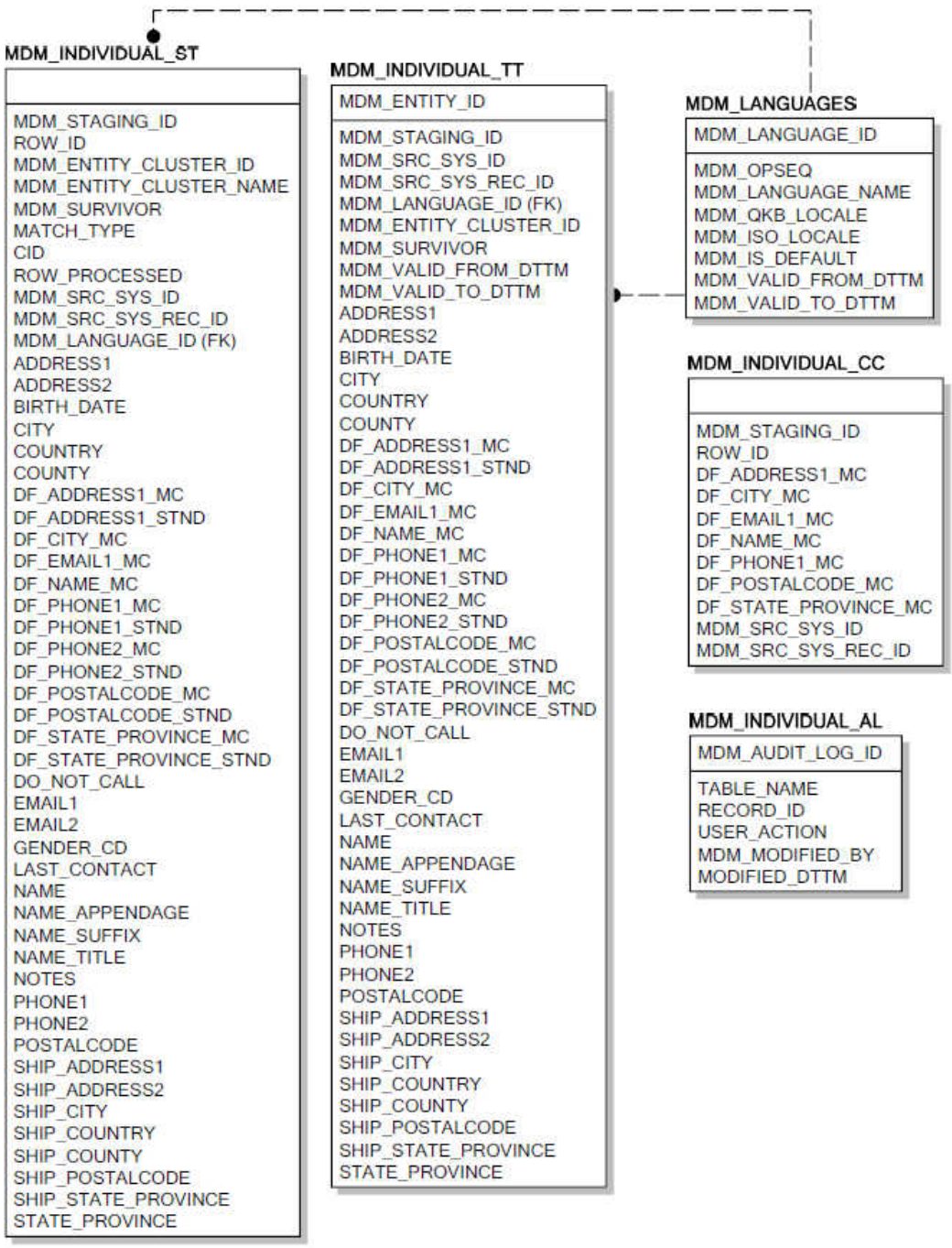
***Figure A1.1*** *Core Data Model*




## *Entity Domain Data Model Diagram*

The following diagram shows the tables generated for the Individual sample entity type. Similar tables are created for the other samples or any other entity type that is defined in the hub. Most of the columns in these tables are represented by the attributes defined for the entity type.

***Figure A1.2*** *Entity Domain Model*



*Appendix 2*

# Service Descriptions



## Required Services

### *Overview*

The services listed in this section are used for internal functions and are not meant to be modified. Any modification might lead to problems in other associated or integrated areas of SAS MDM.

### *Master Data Management Services*

The following services pertain exclusively to the Master Data Management.

***Table A2.1*** *Master Data Management Services*

| Name | Description |
|---|---|
| mrm_retire_entityType.ddf | Retires an entity type. |
| mrm_query_edit.ddf | Loads records from the server for Master Data Management. Because attributes for the entity are returned as name-value pairs, the service can work with any entity type. |
| mrm_get_clusterhistory.ddf | Gets cluster history from the hub. |

| Name | Description |
|---|---|
| mrm_goLive_entityType.ddf | Makes an entity type go live from staging to production. |

## Core SAS MDM Services

The following core SAS MDM services handle more complex tasks, such as adding entities.

***Table A2.2*** *Core SAS MDM Services*

| Name | Description |
|---|---|
| mdm_add_relationships.ddf | Adds entity relationships. |
| mdm_get_archival_count.ddf | Gets an archival count. Called from mdm_archive_<entity_type>.djf. |
| mdm_archive.ddf | Archives rows for a given entity type. Called from mdm_archive_<entity_type>.djf. |
| mdm_start_process_control.ddf | Starts the bulk job load process. |
| mdm_prep_forced_cluster_processing.ddf | Prepares add-update rows for forced cluster processing. Called from add_update_<entity_type>.ddf and mrm_add_<entity_type>.ddf. |
| mdm_match_type_processing.ddf | Applies match type logic to add-update rows for forced cluster processing. Called from add_update_<entity_type>.ddf and mrm_add_<entity_type>.ddf. |
| mdm_forced_cluster_processing.ddf | Applies forced cluster logic to add-update rows. Called from add_update_<entity_type>.ddf and mrm_add_<entity_type>.ddf. |
| mdm_end_process_control.ddf | Ends a running process. |
| mdm_retire_entity.ddf | Retires entities (called by other services). |
| mdm_check_process_completion.ddf | Checks whether the bulk job load process is complete. |
| mdm_make_staged_records_live.ddf | Makes staged records live in the hub. |
| entity_stage_bulk_loader.ddf | Sets the max bulk load and runs the max bulk-loading process until complete. Embedded job used with all the add_update_<entity_type>.djf bulk jobs. |
| mdm_retire_relationships.ddf | Retires entity relationships. |
| mdm_query_hierarchies.ddf | Retrieves a list of hierarchies for the current entity. |
| mdm_rename_hierarchies.ddf | Adds or modifies name and description fields for a hierarchy. |

### *Core Metadata Services*

The following core metadata services handle language metadata, schema evolution, data migration, and default metadata.

***Table A2.3*** *Core Metadata Services*

| Name | Description |
|---|---|
| add_default_meta.djf | Adds default metadata. |
| add_ui_language_meta_de.djf | Adds UI language metadata for German. |
| add_ui_language_meta_en_us.djf | Adds UI language metadata for English. |
| add_ui_language_meta_fr.djf | Adds UI language metadata for French. |
| entitytype_golive.ddf | Takes the name of the entity type and publish it. |
| install_schema.djf | Installs the SAS MDM schema. |
| migrate_mdf_meta.djf | Imports the source systems and language defined in a Master Data Management Foundation hub. |
| add_attributetyperelationships.ddf | Adds the relationship between flux attributes and their base attribute. |
| add_attributetypes.ddf | Adds attribute types to the database. |
| add_clusterconditions.ddf | Adds cluster conditions to the database. |
| add_entitytypeindices.ddf | Adds entity type indices to the database. |
| add_entitytypes.ddf | Adds entity types to the database. |
| add_groupassociations.ddf | Adds group associations to the database. |
| add_i18n_attributeprofiles.ddf | Adds attribute profile translations to the database. |
| add_i18n_attributetypes.ddf | Adds attribute type translations to the database. |
| add_i18n_entitytypes.ddf | Adds entity type translations to the database. |
| add_i18n_profilerelationships.ddf | Adds profile relationships to attribute types. |
| add_i18n_relationshiptypes.ddf | Adds relationship type translations to the database. |
| add_languages.ddf | Adds languages to the database. |
| add_relationshiptypematches.ddf | Adds match conditions to a relationship type in the database. |
| add_relationshiptypes.ddf | Adds relationship types to the database. |

| Name | Description |
|---|---|
| add_sourcesystems.ddf | Adds source systems to the database. |
| load_met.djf | Loads metadata (in the form of a .met file) into the hub. |
| update_entitytypes.ddf | Updates label attributes for entity types. |
| add_entitytypetools.ddf | Adds tool definitions to entity types. |
| add_hierarchylevels.ddf | Adds levels to a hierarchy type definition. |
| add_hierarchytypes.ddf | Adds hierarchy type definitions. |

## Dashboard Services

The following services support the SAS MDM Master Data and Workflow Dashboard reports.

***Table A2.4*** *Dashboard Services*

| Name | Description |
|---|---|
| dashboard_batch_loads.ddf | Returns data about batch loads for the SAS MDM Dashboard report. |
| dashboard_cluster_data.ddf | Returns data about cluster sizes for the SAS MDM Dashboard report. |
| dashboard_contrib_surv.ddf | Returns counts for the number of active and retired contributor and survivor records, by source and entity, for the SAS MDM Dashboard report. |
| dashboard_data_source_snapshot.ddf | Returns information for the Data Source Snapshot section of the SAS MDM Dashboard report. Returns information for the Data Source Snapshot section of the SAS MDM Dashboard report. |
| dashboard_monthly_data.ddf | Returns the number of active and retired contributors and survivors by month, for SAS MDM. |
| dashboard_overall_health.ddf | Returns the numbers of active and retired contributors and survivors currently in the hub, for the Overall Health chart in the SAS MDM Dashboard. |
| dashboard_sparse_data.ddf | Returns counts of fields containing data or no data, by source, entity, and attribute name, for the SAS MDM Dashboard report. |
| dashboard_sparse_data_source.ddf | Returns data on the number of blank and non-blank attributes, by data source and attribute name, for the SAS MDM Dashboard report. |
| dashboard_sparse_data_surv.ddf | Returns counts of fields with blank and non-blank values only for active survivor records, for the SAS MDM Dashboard report. |

| Name | Description |
|---|---|
| dashboard_stats.djf | Analyzes the hub and creates reporting tables for the SAS MDM Dashboard report. |
| dashboard_timeslice_data.ddf | Returns counts of records added to or retired from the hub in the past day, week, month, quarter, and year, for the SAS MDM Dashboard report. |

### Sample Tool Services

The following services support the SAS MDM sample tools.

***Table A2.5*** *Sample Tools Services*

| Name | Description |
|---|---|
| tool_entity_errors.ddf | Show entries from the SAS MDM error log. |
| tool_party_map.ddf | Show the party address in Google Maps. |
| tool_party_us_addver.ddf | Run address verification for party records. |
| tool_company_email_process.djf | Creates and sends e-mail messages. |
| tool_company_search.ddf | Search Google based on the company name. |
| tool_individual_multi_update.ddf | Adds multiple individual records to hub. |
| tool_individual_query_gender.ddf | Search the hub for individuals with Gender='U'. |
| tool_part_email_attachment.djf | E-mails the list of parts from hub. |

## Sample Services

### Overview

These jobs and services are specific to sample entity types provided with SAS MDM and must be modified if you are not using the standard definition of entity types provided with SAS MDM. They can be customized for other entity types by following the notes inside each job.

### Sample MDM Services

The following service provides support for the sample entity types. For a list of the generated jobs for an entity type, see the *SAS MDM User's Guide*.

*Table A2.6 Sample MDM Services*

| Name | Description |
|---|---|
| mrm_stnd_<entity_type>.ddf | Provides computation of match codes and standardized values for the sample entity types. |
| mdm_extract_household.ddf | Demonstrates changes required to be made to import MDF instance data into the hub. |
| mdm_extract_individual.ddf | Demonstrates changes required to be made to prepare incoming rows for remediation and workflow. The **`Check for required fields node`** in this job includes use cases. |

### *Sample Metadata Services*

The following services load metadata into the SAS MDM hub.

*Table A2.7 Sample Metadata Services*

| Name | Description |
|---|---|
| add_indices.djf | Loads entity type indices. |
| add_sample_meta.djf | Loads sample entity types. |
| add_ui_language_sample_de.djf | Loads German sample UI metadata. |
| add_ui_language_sample_enusa.djf | Loads English sample UI metadata. |
| add_ui_language_sample_fr.djf | Loads French sample UI metadata. |

## Schema Evolution Service

Schema evolution is a service for running schema evolution scripts.

*Table A2.8 Schema Evolution Service*

| Name | Description |
|---|---|
| evolve_schema_41.djf | Runs the schema evolution scripts to upgrade the database from the SAS MDM 4.1 service level to the first maintenance release for SAS MDM 4.1 service level. |

# Index