REDPOINT MASTER DATA MANAGEMENT

REDPOINT GLOBAL
| 36 WASHINGTON STREET
WELLESLEY HILLS, MA 02481
+1 781 725 0250 | WWW.REDPOINTGLOBAL.COM

CONTENTS

REDPOINT MDM OVERVIEW

Redpoint Redpoint Master Data Management (MDM) is a data stewardship system for managing the *Golden Record*: a single, accurate, and always up-to-date view of your most critical data. The MDM system includes a variety of capabilities for controlling your data:

Data model management

Database administrators (*DBAs*) can create and manage data models, including:

- Tables (including versioned, reference, and time-series types)
- Schemas
- Linkages
- Indexes
- Personally Identifiable Information (PII) designations
- Masking rules

Data models may evolve over time so long as changes do not invalidate existing data.

You can find more information on the MDM data models here.

Data versioning

All create, read, update, and delete (CRUD) changes to versioned tables are retained in history, and time-series tables are append-only. This approach ensures that information is never lost (unless purged for privacy compliance reasons), and historical search and comparison is always available. Queries may reference current data or historical data at any point in time, facilitating historical comparison and trending reports. Users of MDM can drill down into individual record history.

Changesets

Data and metadata changes are made via a *changeset* (explicit commit step). This ensures that all changes occur in a coordinated and linear fashion.

Conflict resolution

If simultaneous changes to the same data are attempted, the conflict resolution system will merge compatible changes or flag incompatible changes for review.

Matching

The MDM system supports a rich matching model:

- Users can create Redpoint Data Management projects to perform fuzzy matching on MDM records, either during load or after load.
- Multiple match levels (for example, individual/household) are supported.
- Automatic matches can be reviewed by data stewards using the workflow system.
- If a data steward overrides automatic match logic, these decisions are retained to prevent automatching from making the same mistake twice.

Workflow

MDM supports a cooperative workflow model in which users can coordinate and review data changes. For example:

- Data changes are submitted by business users but reviewed by data stewards before commit.
- Borderline automatic match results are reviewed by data stewards and accepted or rejected.
- A manual match override can be suggested by one user and reviewed by another.

ETL connectors

Redpoint Data Management is a powerful extract, transform, and load (ETL) tool that connects directly to the MDM system for the purpose of loading, querying, extracting, and matching. These connectors work in web services to support real-time transactions.

Privacy compliance

MDM supports compliance with privacy rules such as the EU's General Data Protection Regulation (GDPR), California Consumer Privacy Act (CCPA), and Brazil's Lei Geral de Proteção de Dados (LGDP) by allowing personally identifiable information (PII) to be masked or purged from the system. Such actions reach back into history and remove all traces of PII, so successful erasure can be verified.

Computed views and subscriptions

MDM supports a computed-view model that combines table data, joins, and aggregation rules. This is ideal for:

- Golden record creation
- Marketing and statistical variables
- Lifetime-value calculations

A view may be queried directly, and such queries may reference historical or current data. A view may also be *subscribed* to, in which case changes to underlying table data trigger real-time updates of the view results, and incremental delivery of the changes. In addition, view aggregation rules may contain date-sensitive filters (for example, transactions in the last 30 days), the results of which are automatically updated nightly.

Web service APIs

For integration with other systems, all of MDM's features are accessible via its web service APIs.

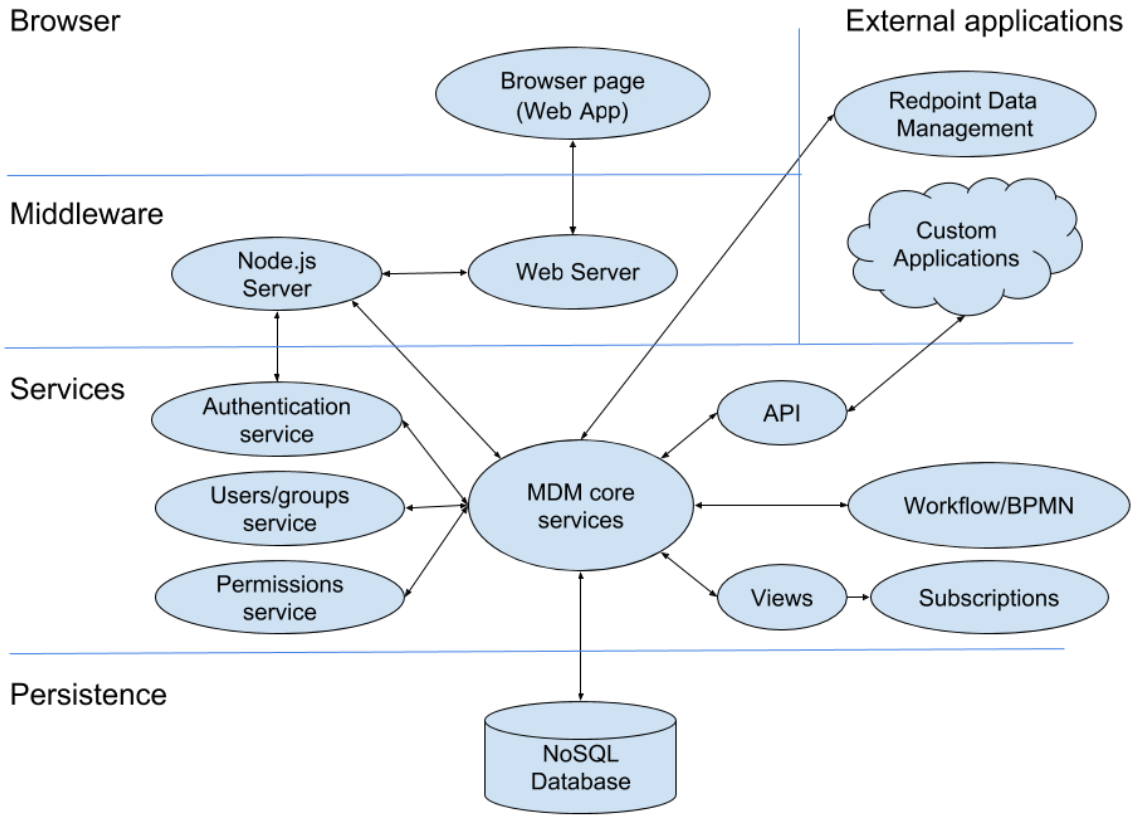
Scripting

A scripting tool is provided to support typical provisioning steps:

- Creation of data models
- Import of reference data
- Setting up users, groups, and permissions

1.1. HOW MDM WORKS

MDM consists of these components:



MDM architecture

Browser

The MDM Web App is the Node.JS-based client application for MDM.

External applications

Redpoint Data Management ingests, cleanses, transforms, and integrates data. A Redpoint Data Management ETL process automatically identifies potential duplicate records and compares them in both "loose" and "tight" match passes. If these produce different results, the process initiates a MDM review workflow so that a steward can choose the preferred match.

Middleware

- The web server handles requests from the Web App by sending requests to and accepting data from the Node.js server.
- The Node.js server runs server-side JavaScript to produce dynamic web page content before the page is sent to the web app. It also sends requests to and accepts data from the Services layer.

Services

MDM core services are the heart of MDM. They accept and process requests from the Web App, query the computational subservices, and access a NoSQL database.

- The **Authentication** subservice authenticates MDM users. A user who successfully logs into MDM is issued a security token, which must be submitted along with any request. This service also authenticates a user's security token at every system request.

- The **Users/groups** subservice keeps track of MDM users and the access groups to which they belong.
- The **Permissions** subservice keeps track of user permissions.
- **Workflows** are constructed from Business Process Model and Notation (BPMN) models. This module contains the logic for processing workflow steps.
- A **View** is a directed graph of data transformations (or *projections*) that supports complex queries including joins and aggregations. You can also create **Subscriptions** to a view, which notifies you when the data in a view changes (that is, when records are added, deleted or modified). When views and subscriptions change, these services push the changes out to the subscribers via the chosen method (currently only web-service-call delivery is supported).
- All services are directly accessible via the MDM API.

Persistence

MDM is backed by a NoSQL document database.

1.2. DISPLAY ONLINE HELP

MDM provides online documentation to assist you in your work.

To display online help

1. At the bottom of the Navigation tree of the MDM Web App, select **Help**.
2. Select **Documentation** from the menu that displays.

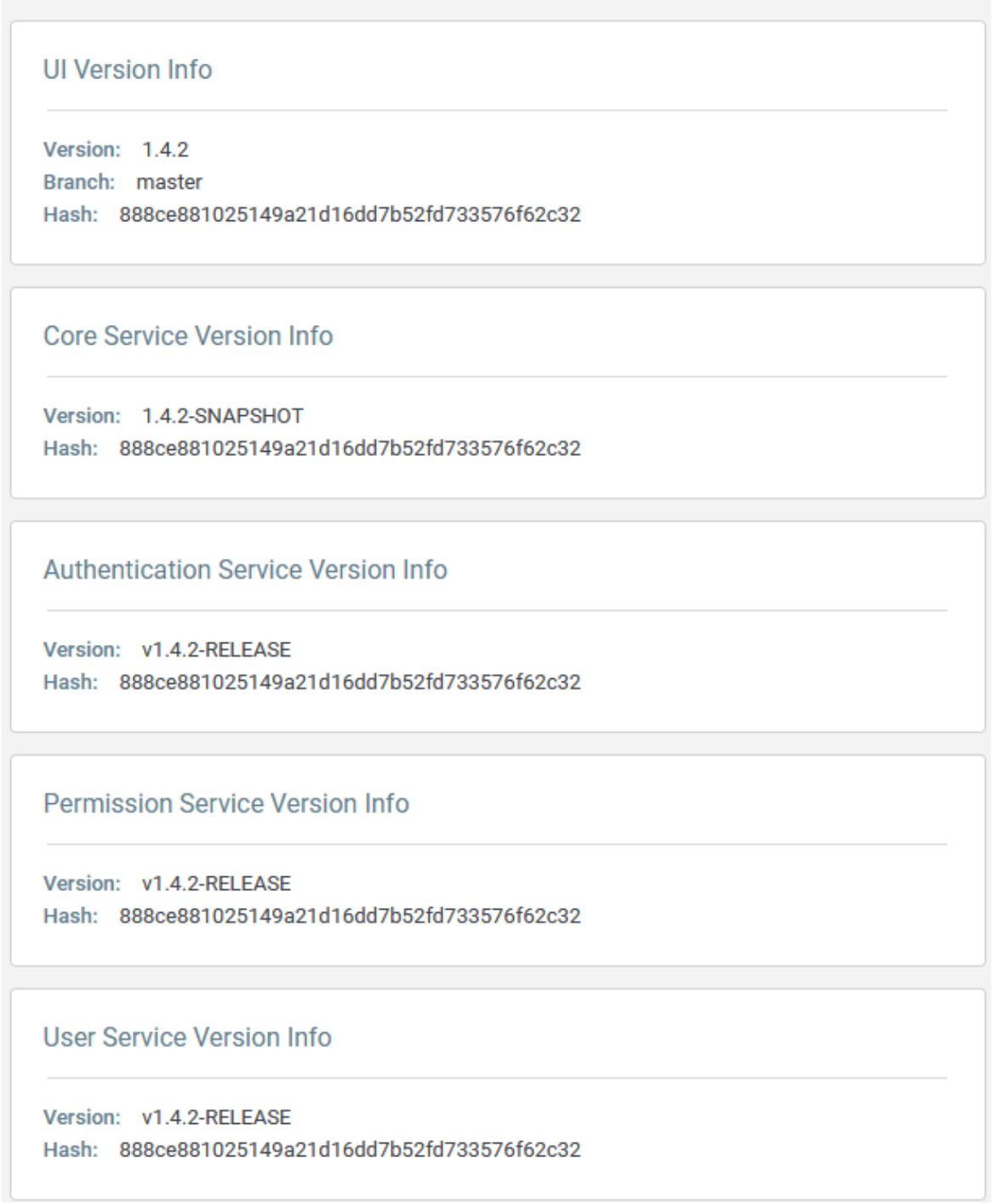
1.3. VIEW VERSION INFORMATION

You can easily view version information for MDM. The UI Version, Core Service Version, Authentication Service Version, Permission Service Version, and User Service Version info display.

To view version information

1. At the bottom of the Navigation tree of the MDM Web App, select **Help**.
2. Select **About MDM** from the menu that displays.

   The Version information and Hash information display for each entity. Additionally, for the UI Version, the Branch displays.



Version Information

1.4. OBTAIN SUPPORT

We want to make your experience with MDM as pleasant and productive as possible. Please email or call us when you need help. See Troubleshooting for a list of information we'll need in order to help you.

Contact us at:

Redpoint Global Inc.
888 Worcester Street, Suite 200
Wellesley Hills, MA 02481

Phone: +1 781 725 0250

Fax: +1 781 235 3739

Email: support@redpointglobal.com

You can also obtain support information by doing the following:

1. At the bottom of the Navigation tree of the MDM Web App, select **Help**.
2. Select **Redpoint Support** from the menu that displays.

1.5. TROUBLESHOOTING

When you contact us about a problem, please include the following information:

- Product serial number and version number (view this by clicking on the **Help** link at the bottom of the Web App navigation menu and selecting **About MDM**).
- Your name, company name, telephone number, and email address.
- Operating system and version number.
- Exact wording of any error messages that appear.
- Exactly what you were doing when the problem or error occurred.
- How you tried to resolve the problem.

We may also ask you for:

- A sample of the input data.
- Your system configuration.

2. THE WEB APP

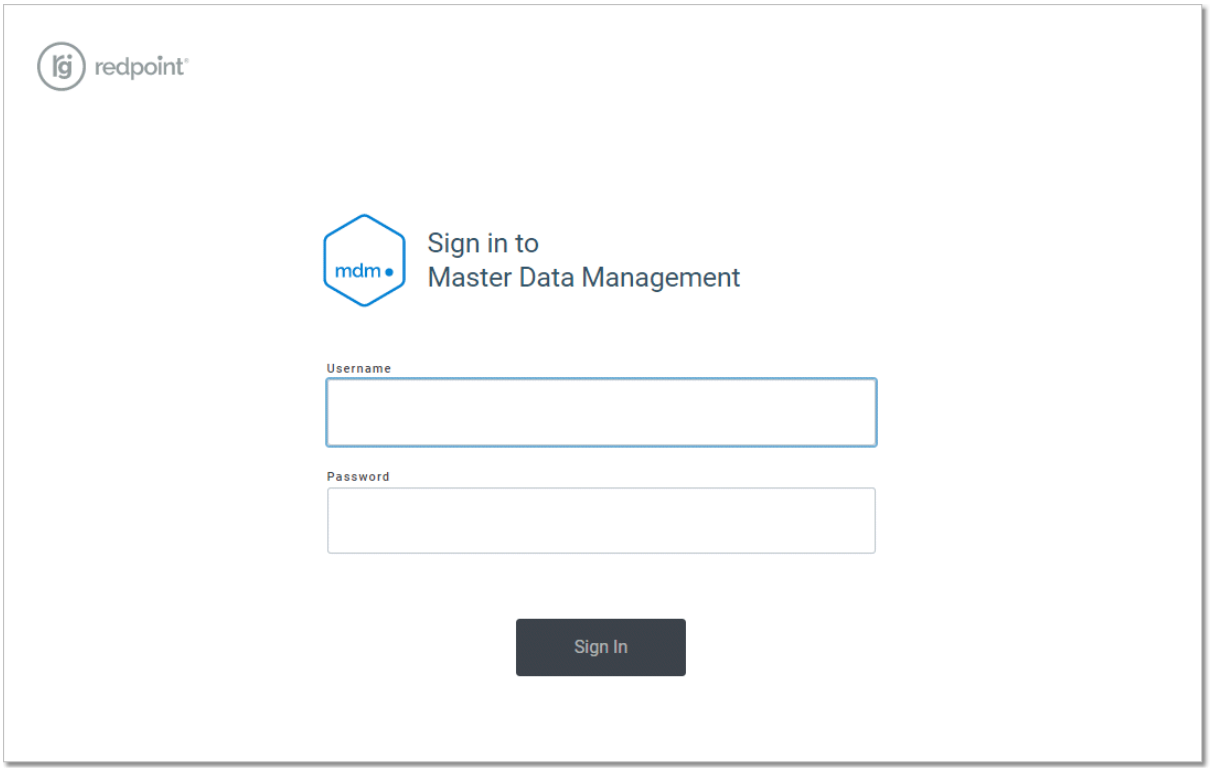
The Web App (or MDM client) is the stewardship interface for MDM. While all MDM functionality is available through the MDM API, the MDM app is the primary means of:

- Viewing records at any point in the database history
- Adding, modifying, and deleting records (for data stewards and DBAs—for business users, the primary mechanism will be a workflow)
- Creating, modifying, and deleting databases and tables
- Reviewing records
- Masking and erasing records
- Confirming that specific records have been erased

2.1. LOG ON TO REDPOINT MDM

To log on to the MDM Web App

1. In a browser, enter the URL for your MDM Web App. (By default, the URL is *server*/mdm:9090, where *server* will vary depending on the machine/network.)

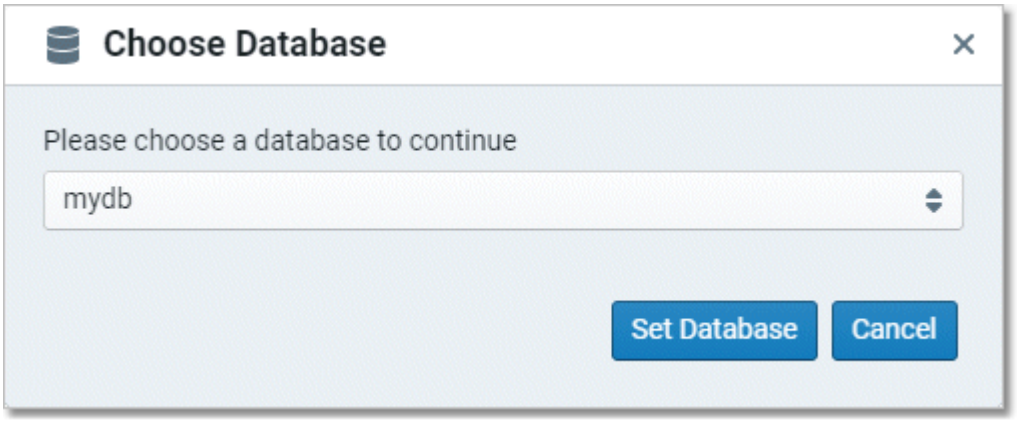


MDM logon page

2. Enter your **Username** and **Password**, and click **Sign In**.

   Note that during the initial configuration of MDM, you will log on as **system**.

   When you log on to MDM for the first time, the **Choose Database** dialog displays. Select the database and click **Set Database**.



Choose database dialog

2.2. DASHBOARD

The Dashboard is the highest element in the Navigation menu. At the top, the Dashboard displays different workitem statistics depending on your role: the number of workitems assigned to you, the number of workflows started by you, and the number of active workitems.

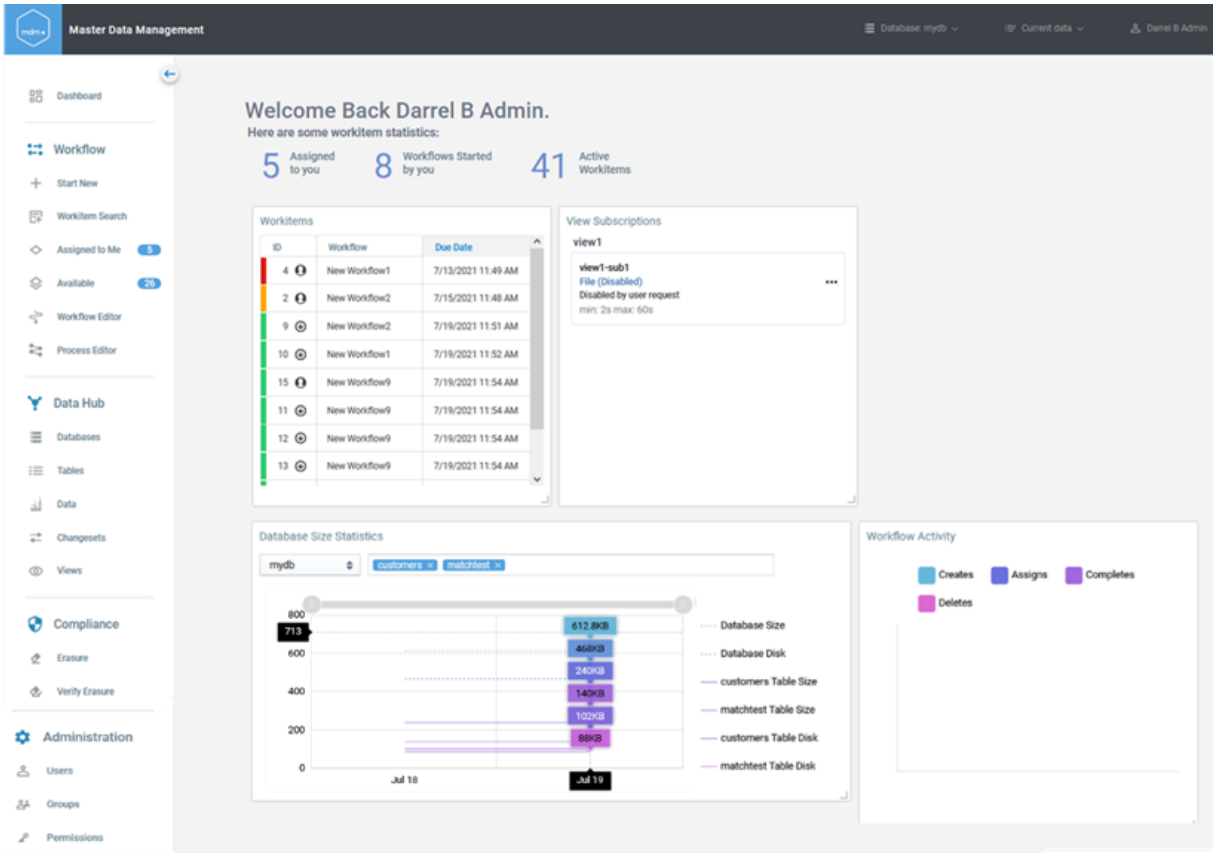
The Dashboard contains panels with the following information:

- Workitems - Displays a list of workitems assigned or available to you with the ID, Workflow name, and Due Date. A vertical bar displays to the left of the workitem ID reflecting the due date:
  - Due date more than 48 hours in the future - green
  - Due date within 48 hours - yellow
  - Due date in the past - red

You can Claim (down-facing arrow icon) or Continue (person icon) a workitem by clicking the icon to the right of the workitem ID.

- View Subscriptions - Displays a list of configured subscriptions, such as customer transactions over a set time period. You can click a subscription to display the details. For details on subscriptions, see Create a subscription.
- Database Size Statistics - Displays statistics for the selected databases. The size displays on the vertical axis and the date displays on the horizontal axis. This view provides diagnostic tools that allow you to view the growth of tables and the amount of disk space in use over time.
- Workflow Activity - Contains the legend for the Database Size Statistics panel.

You can resize each of the panels and move them to different positions. The layout will remain between logons. To reset the layout to the default layout, see Settings.



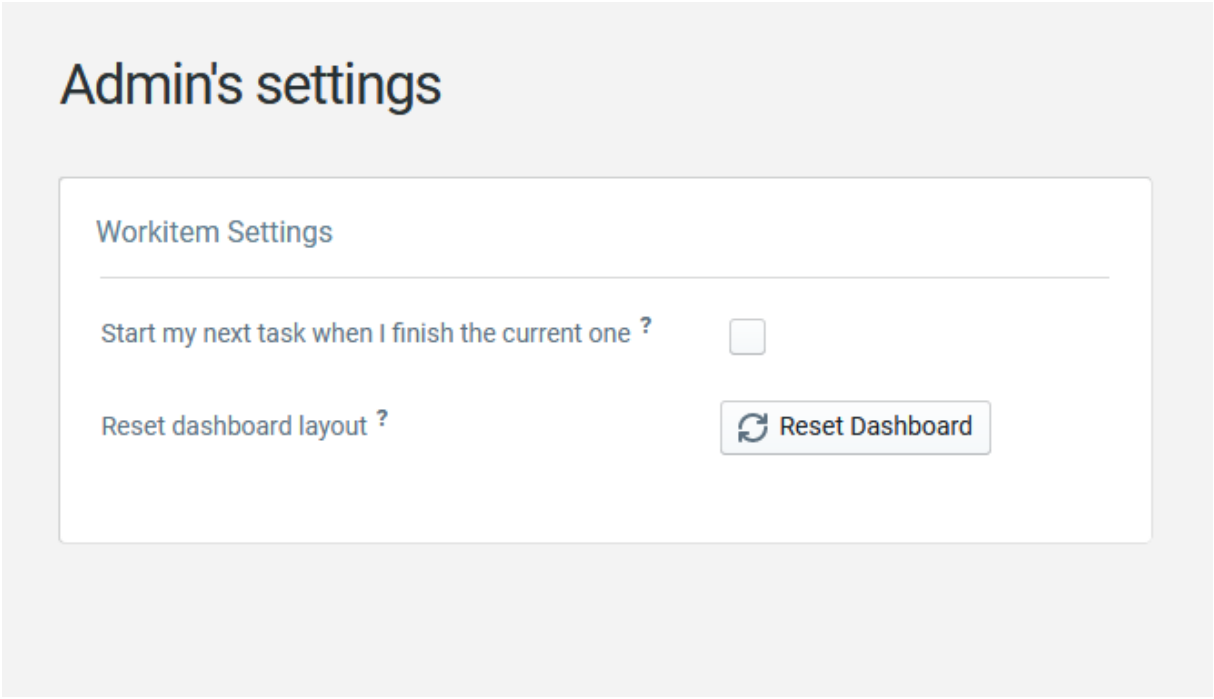
MDM Dashboard

2.3. WORKITEM SETTINGS

Workitem settings consist of two options: whether tasks are automatically started when the previous task is completed and whether the dashboard layout is reset.

To modify workitem settings

1. At the upper-right corner of any Web App page, click your user name.
2. Select Settings from the menu that displays.



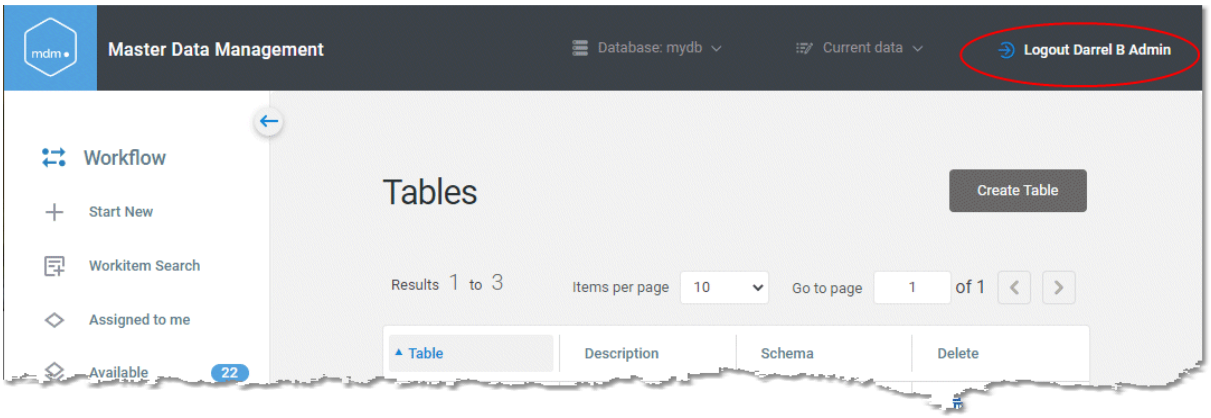
Settings dialog

3. To automatically start the next assigned task when you complete the previous one, select the **Submit my next task when I finish the current one** check box.
3. To reset the dashboard layout to the default layout, click the **Reset dashboard layout** button.

2.4. LOG OFF REDPOINT MDM

To log off the MDM Web App

1. At the upper-right corner of any Web App page, click your user name.



MDM Console logout link

2. Select **Logout** from the menu that displays.

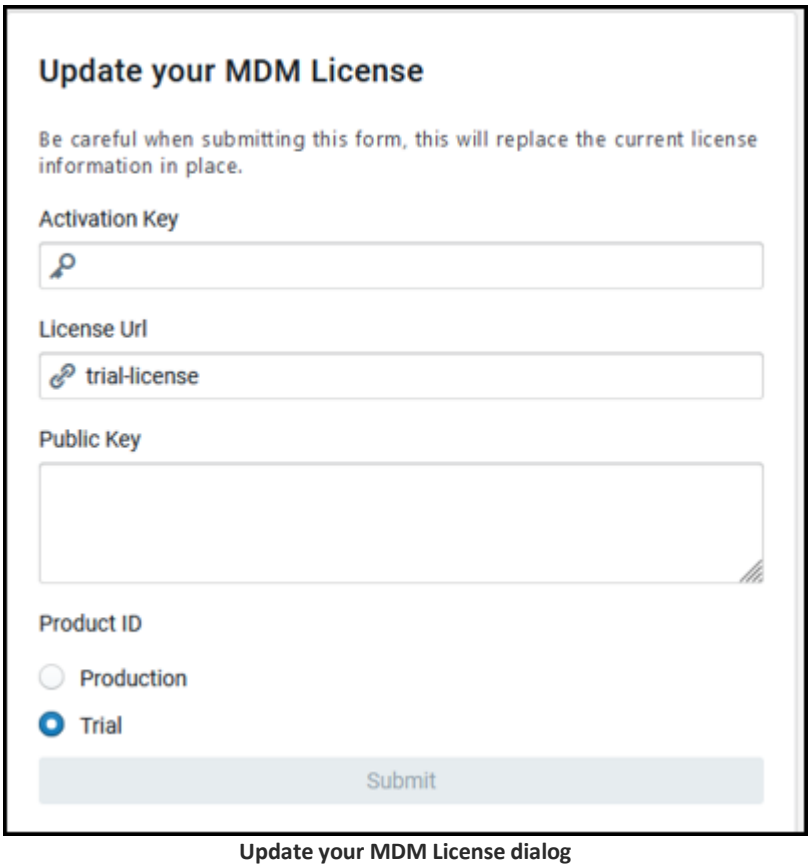
2.5. UPDATING YOUR MDM LICENSE

If your database is destroyed, is new, or your license is no longer valid, you can update your MDM license information.

To update your MDM license

1. In a browser, enter the URL for your MDM Web App followed by **/license**.

   The Update your MDM License dialog box appears.



Update your MDM License dialog

2. Enter the **Activation Key**.

3. Enter the **License URL**.

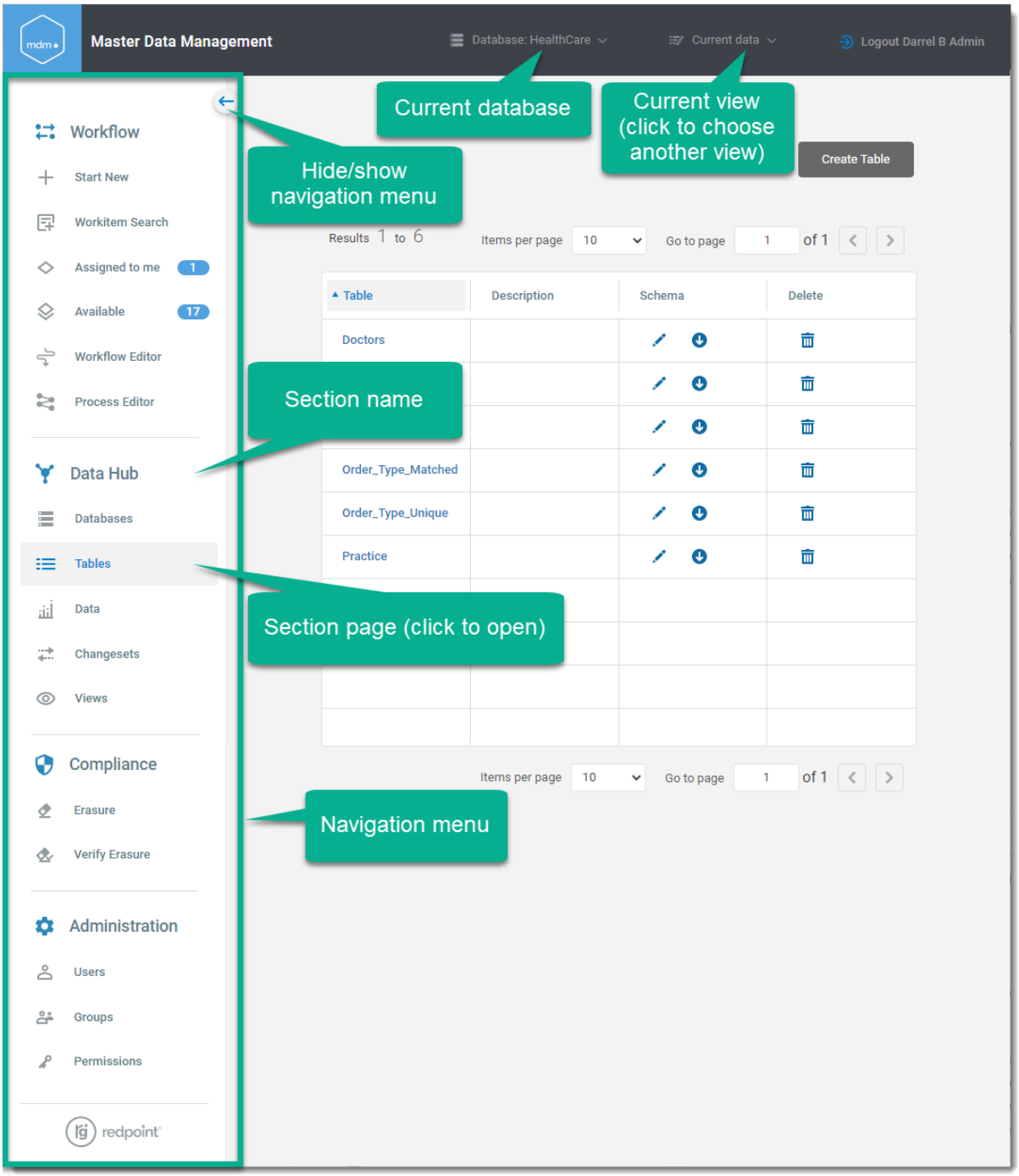
4. Enter the **Public Key**.

5. Select the Product ID: **Production** or **Trial**.

6. Click **Submit**.

2.6. WEB APP OVERVIEW

This following screen shot shows the components of the MDM Web App:



MDM Console

2.6.1. NAVIGATION MENU

The main content areas of the MDM Web App are:

Dashboard

The Dashboard section is at the top of the Navigation menu and provides a summary of your workitems, computed views, and database size statistics.

Workflow

The concept of *workflow* is central to the MDM process. A workflow is a process for performing a series of tasks on the master data, often requiring actions by multiple users. Workflows are bound to databases. The available actions in a given workflow are determined by the permissions of the group of which the user is a member. For example, when a user who is a member of the **Bizuser** group clicks **Start New** in the **Workflow** section, she may choose to create either a **New Customer** or an **Edit Customer** workflow item (or *workitem*). When that workitem is submitted, it becomes available for review by a user who is a member of the **Steward** group. There are different kinds of workflows such as creating a record, updating a record, reviewing proposed data changes, and reviewing potential record matches. Users in the **system** or **dbas** groups have permissions to define Business Process Model and Notation (BPMN) processes and create, edit, and delete workflows that implement these processes.

You can find more information on the MDM workflow here.

Data Hub

The Data Hub section allows interactive access to databases and tables. It allows you to:

- Select/change the current database
- Search for and query tables
- View a record's history
- Change metadata, including:
  - Create or delete a database
  - Create, delete, or modify tables
  - View, edit, and export table schemas

You can find more information on the Data Hub section here.

Compliance

This section contains Masking and Erasure functions as well as a Verify Erasure function that lets you confirm whether a record was masked or purged. These features are only available for tables that have been configured with one or more PII columns and masking hashes.

You can find more information on the Compliance section under GDPR and privacy support.

Administration

This section lets you view and configure:

- Who can access MDM
- What users can access in the Web App

- What each user can do with each MDM entity

You can find more information on the Administration under Security.

Help

This section displays online documentation and indicates the version of the software.

3. THE DATA HUB

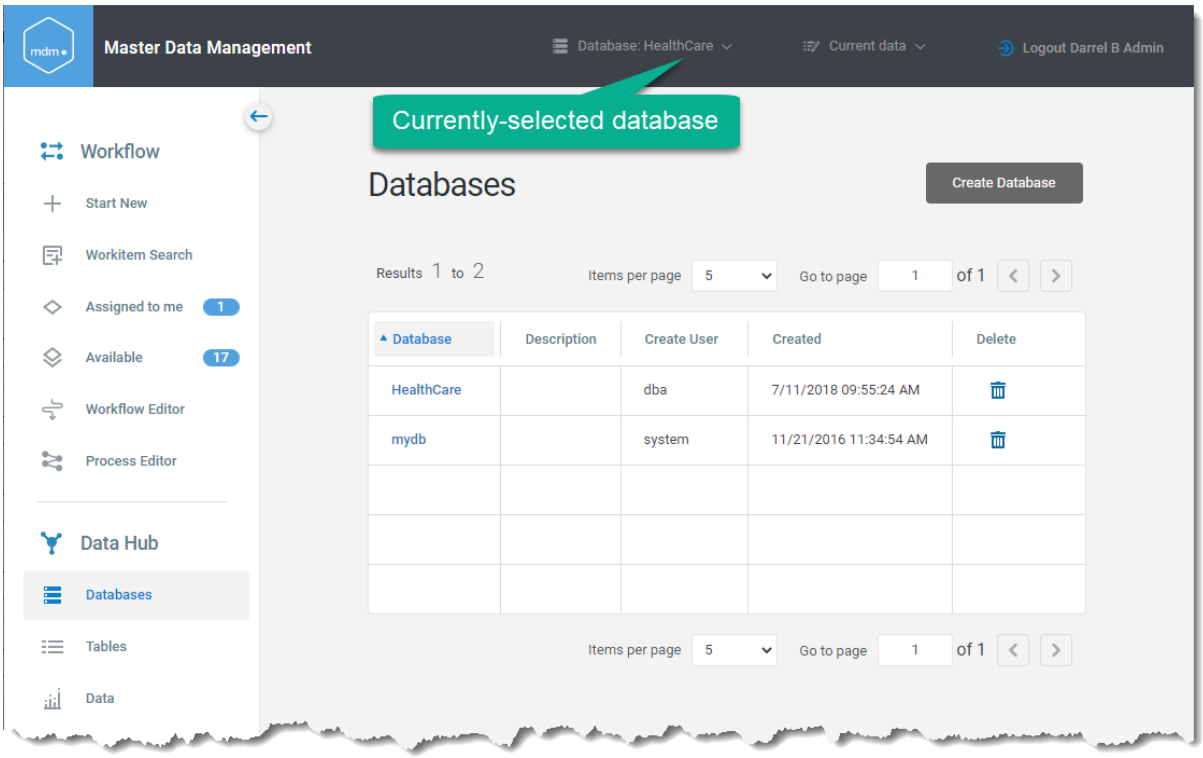
The **Data Hub** section allows interactive access to databases and tables. Depending on your permissions, you may:

- Select/change the current database
- Search for and query records in tables
- View a record's history and activity
- Change metadata, including:
  - Create or delete a database
  - Create, delete, or modify tables
  - View, edit, and export table schemas

3.1. SELECT/CHANGE THE CURRENT DATABASE

To select or change the current database

1. In the **Data Hub** section of the MDM Web App, click **Databases**. All the databases for the current MDM instance are displayed. **Note**: In many organizations, users will work within a single database.



Select or change the current database

If there are more databases than can be listed on the page:

- You can control the number of databases listed per page by changing the **Items per page** option.
- You can use the **Go to page** option or previous [<] and next [>] buttons to move between table pages.

2. Click on the database name.

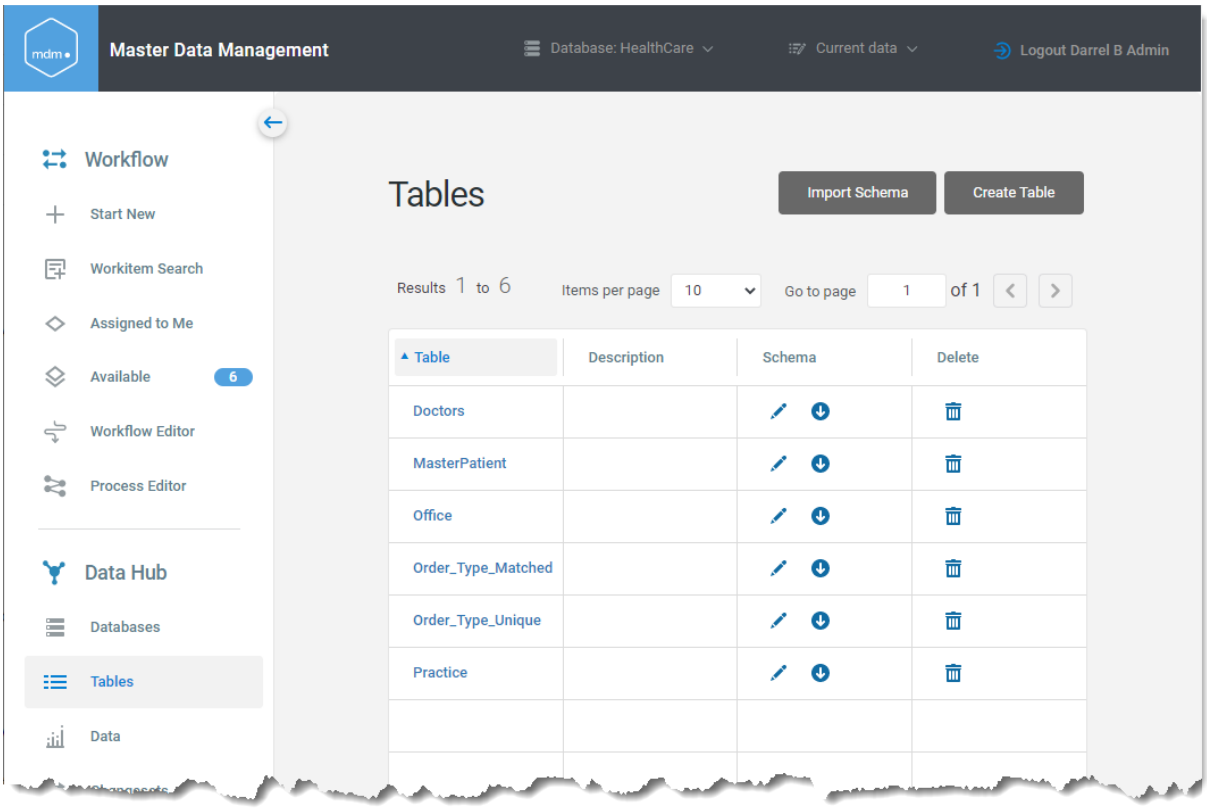
3.2. SELECT A TABLE

Before you select a table, make sure to select the correct database.

3.2.1. FROM THE TABLES PAGE

To select a table from the **Tables** page

1. In the **Data Hub** section of the MDM Web App, click **Tables**. The tables for the currently selected database display.



Selecting a table from the Tables page

If there are more tables than can be listed on the page:

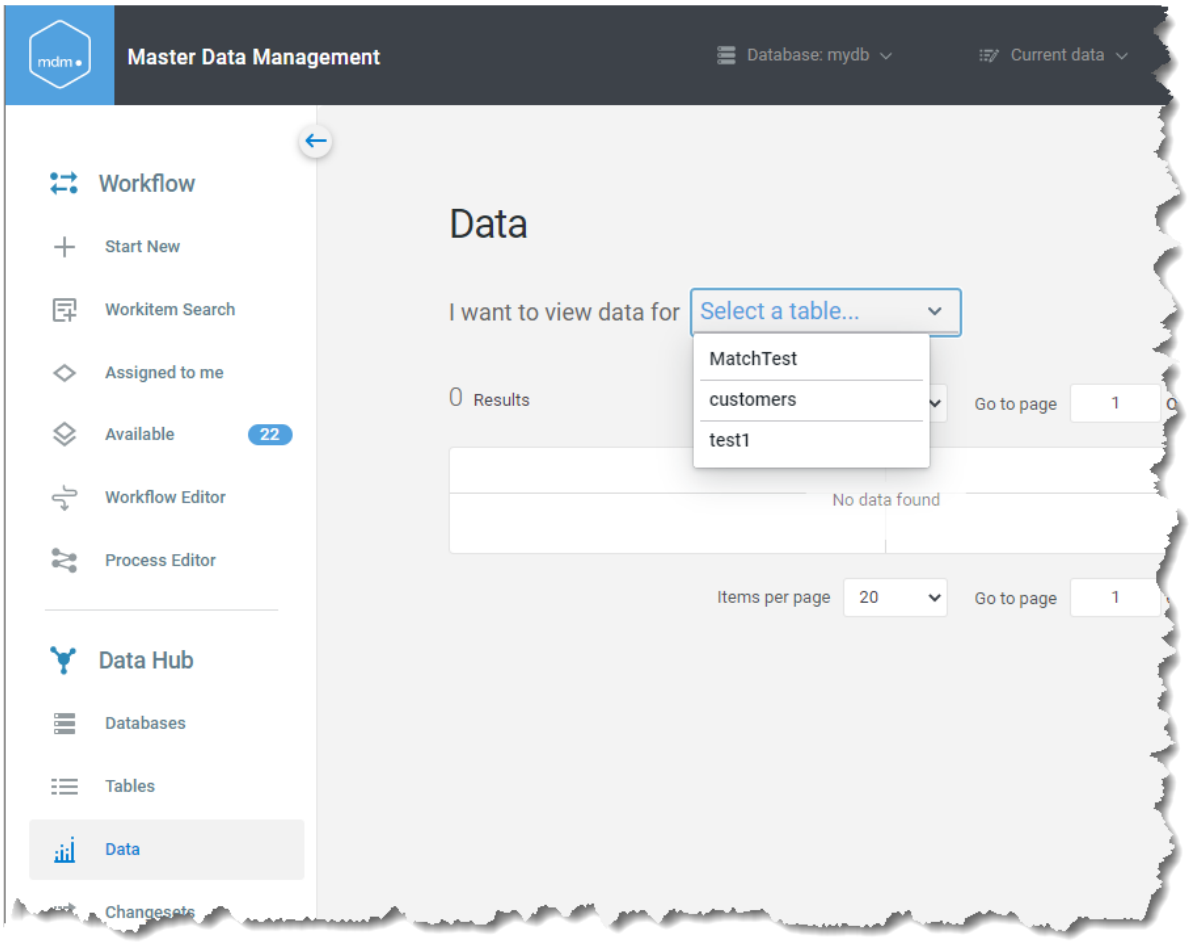
- You can control the number of tables listed per page by changing the **Items per page** option.
- You can use the **Go to page** option or previous [<] and next [>] buttons to move between table pages.

2. Click on the table name.

3.2.2. FROM THE DATA PAGE

To select a table from the **Data** page

1. In the **Data Hub** section of the MDM Web App, click **Data**.
2. On the **Data** page, click **Select a table**, and then choose the desired table from the list.



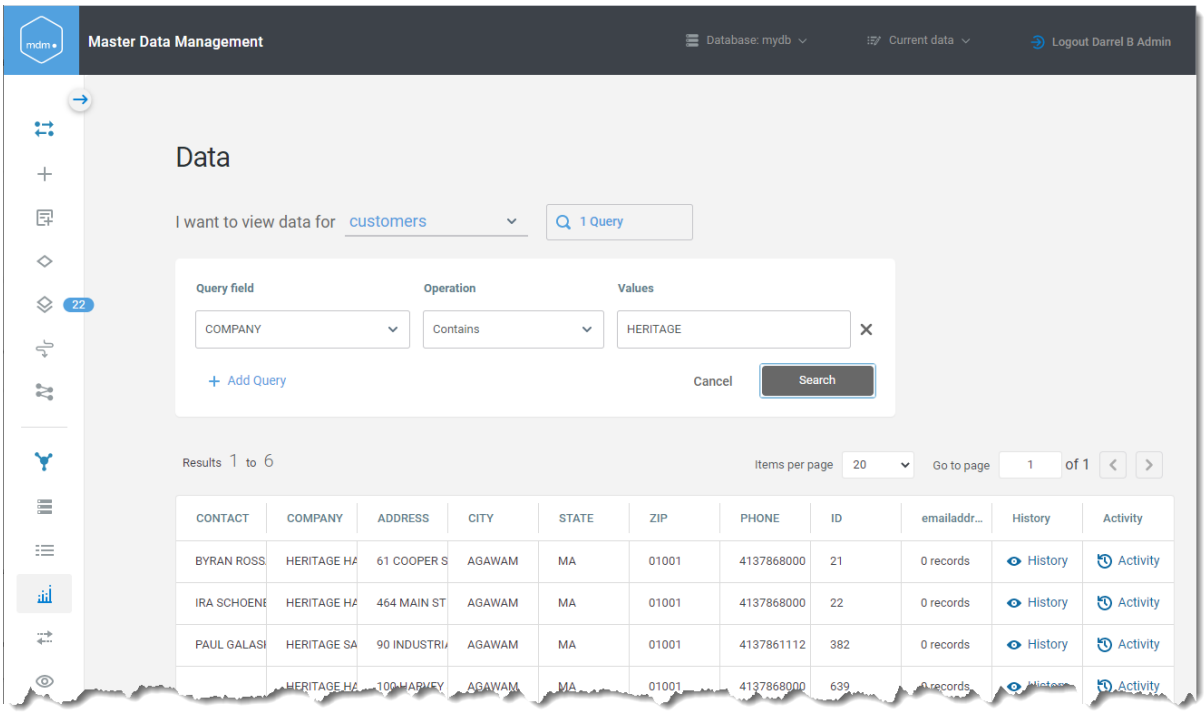
Selecting a table from the Data page

3.3. QUERY RECORDS IN A TABLE

Before you query table records, make sure to select the correct database and table.

To query records in a table

1. In the **Data Hub** section of the MDM Web App, click **Data**.
2. Click the **Query** button to display the query builder, construct a query, and then click the **Search** button.



Querying records in a table

3.3.1. HOW TO CONSTRUCT A QUERY

Various MDM Web App tasks and workflows require that you define a query in order to locate records.

A query is a list of query terms. A query term consists of three elements:

- The *field* that will be queried.
- The query *operation* to perform.
- The *values* to be tested with the query operation.

Records returned by the query must match all query terms (that is, logical *and* behavior terms).

MDM supports the following query operations:

Operation	Meaning
`field = value`	The set of records where field is equal to value.
`field <> value`	The set of records where field is not equal to value.
`field > value`	The set of records where field is greater than value.
`field < value`	The set of records where field is less than value.
`field >= value`	The set of records where field is greater than or equal to value.

Operation	Meaning
`field` **`<=`** `value`	The set of records where field is less than or equal to value.
`field` **`inrange`** `value1,value2`	The set of records where field is within the inclusive range of two comma-separated values.
`field` **`startswith`** `value`	The set of records where field starts with value.
`field` **`endswith`** `value`	The set of records where field ends with value.
`field` **`contains`** `value`	The set of records where field contains value.
`field` **`oneof`** `value1,value2,...`	The set of records where field is one of a comma-separated list of values.
`field` **`regex`** `value`	The set of records where field matches a standard regular expression value.

Notes

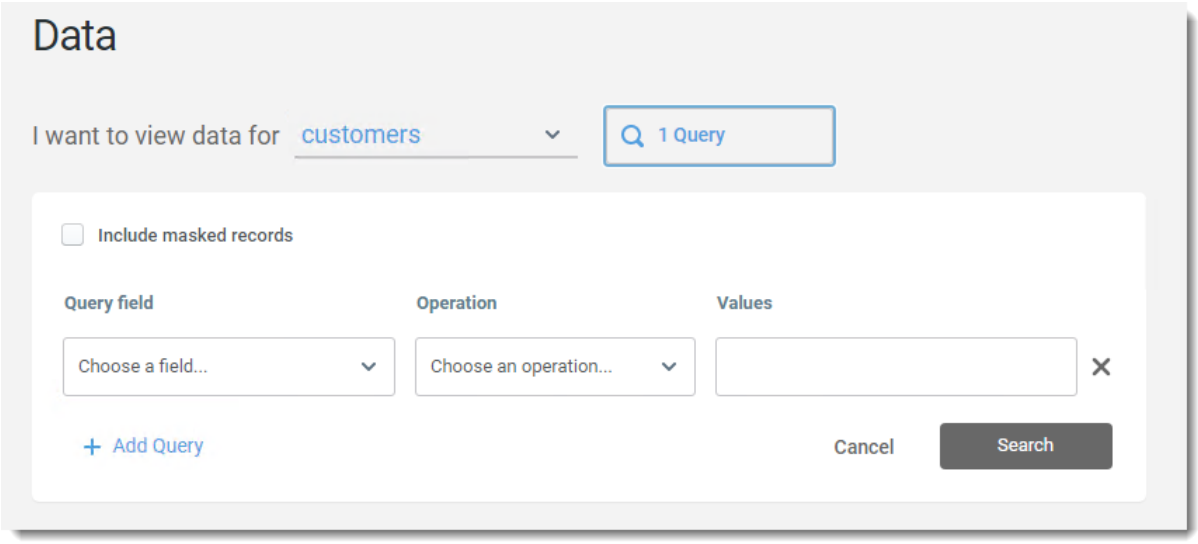
Queries are case-insensitive. The query `NAME contains John` will return records containing any of these values: `JOHN John john`.

Match group IDs (available if a table schema has **Display Match Groups** enabled) can only be queried for a single value in a single field using the equals operation (`=`).

3.3.2. CONSTRUCTING A QUERY IN THE CONSOLE

To construct a query in the MDM Web App

1. Select the table you want to query.
2. If the data being queried contains masked records, you may optionally select **Include masked records**.
   - Masked records are displayed in the data grid with asterisks (*) instead of data in PII fields.
   - See GDPR and privacy support for more information on record masking.
3. Choose a **Query field** and **Operation**, and enter one or more **Values.**
   - To delete a query, click the query's **Delete** button (**x**).
   - You can add additional queries by clicking **Add query**.
4. To execute the query, click the **Search** button.



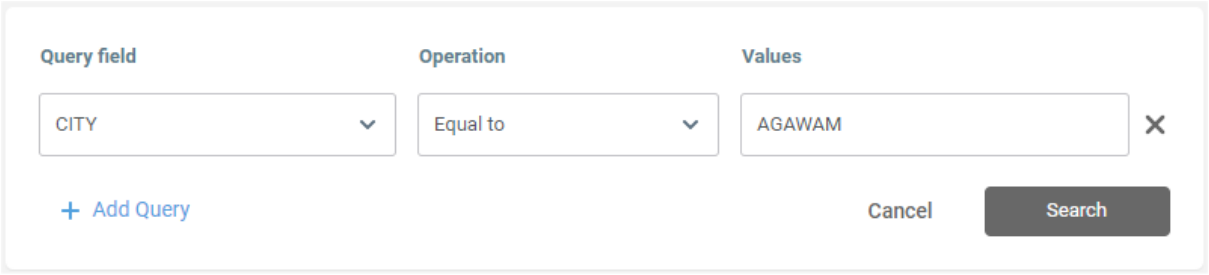
Constructing a query in the MDM Console

3.3.3. EXAMPLE: CONSTRUCTING A QUERY WITH TWO TERMS

Let's construct a query in which we search for records of companies that have the word FIRST in their name and are located in Agawam.

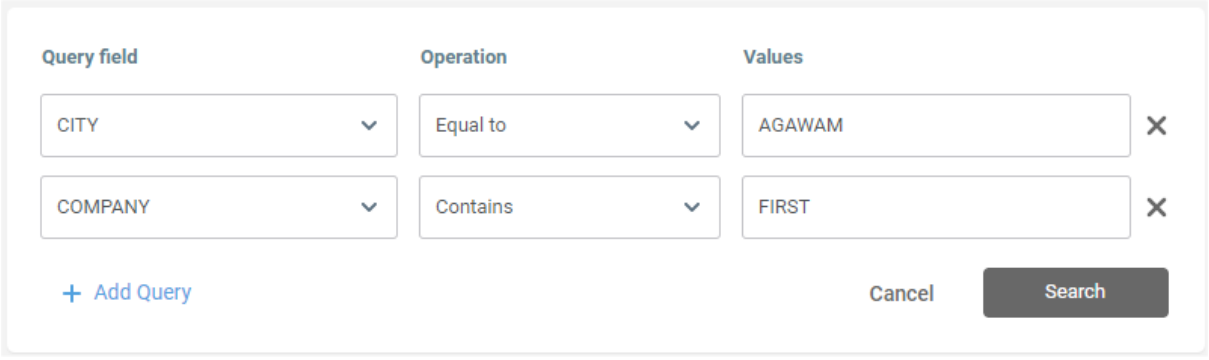
In the MDM Web App **Data Hub** > **Data** section, we need to define two query terms—one to specify the city of Agawam, and one to find company names containing the word FIRST.

For the first query term, we select **CITY** as the query field, and set that **Equal to** the value **AGAWAM**. The first query term looks like this:



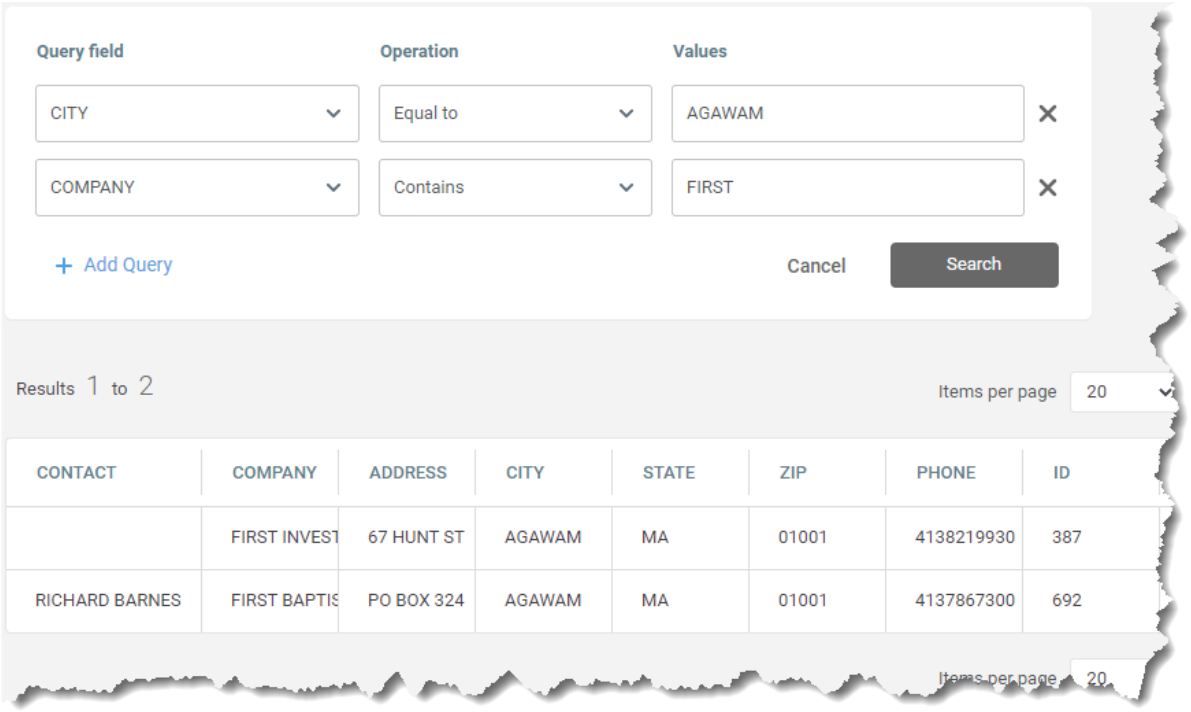
Constructing a query with one term

Next, we click the **Add Query** button. For the second query term, we select **COMPANY** as the query field, and set that as **Contains** the value **FIRST**. The second query term looks like this:



Constructing a query with two terms

Now we click the **Search** button to execute the query. The matching records are displayed (in our case, two records matched the query):



Results of a search with a two-term query

Notes

For a query, MDM returns a maximum of 1000 records at a time. Click the More button to fetch additional records from the server.

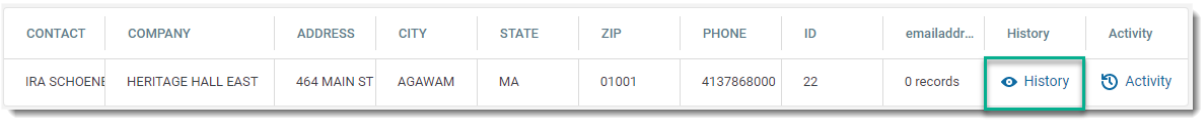
In record queries, only fields with a **display** field value of **true** are shown.

3.4. VIEW A RECORD'S HISTORY

On the MDM Web App's **Data Hub** > **Data** page, you can view the history for a given record.

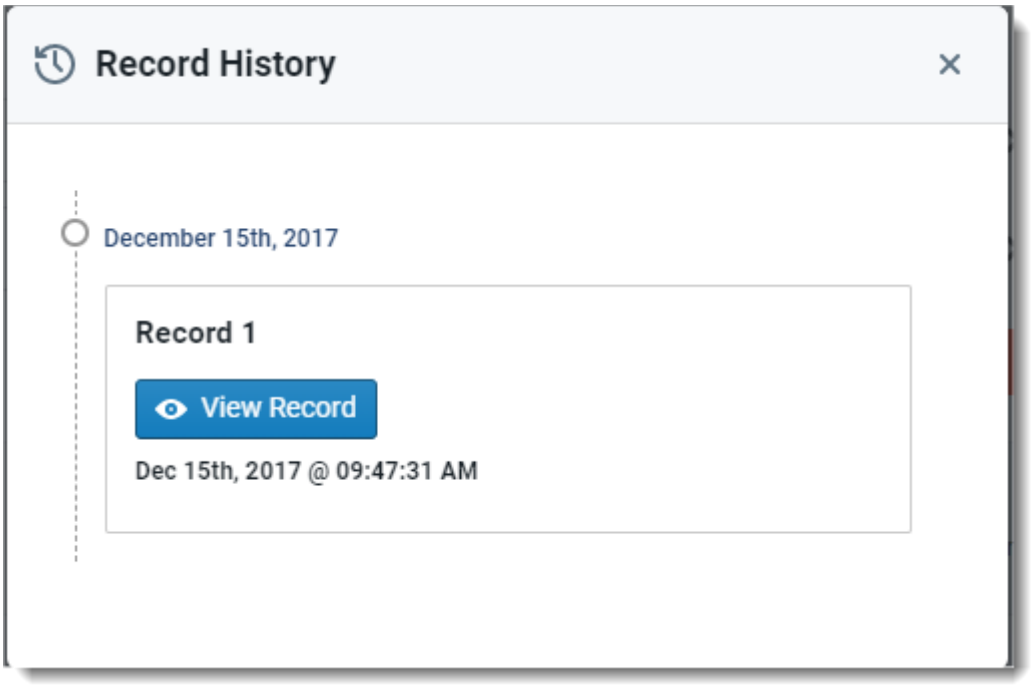
To view a record's history

1. In the **Data Hub** section of the MDM Web App, click **Data**.
2. Select a table.
3. Find the record of interest (you may have to define a record query), and then click the **History** link in its **History** column.



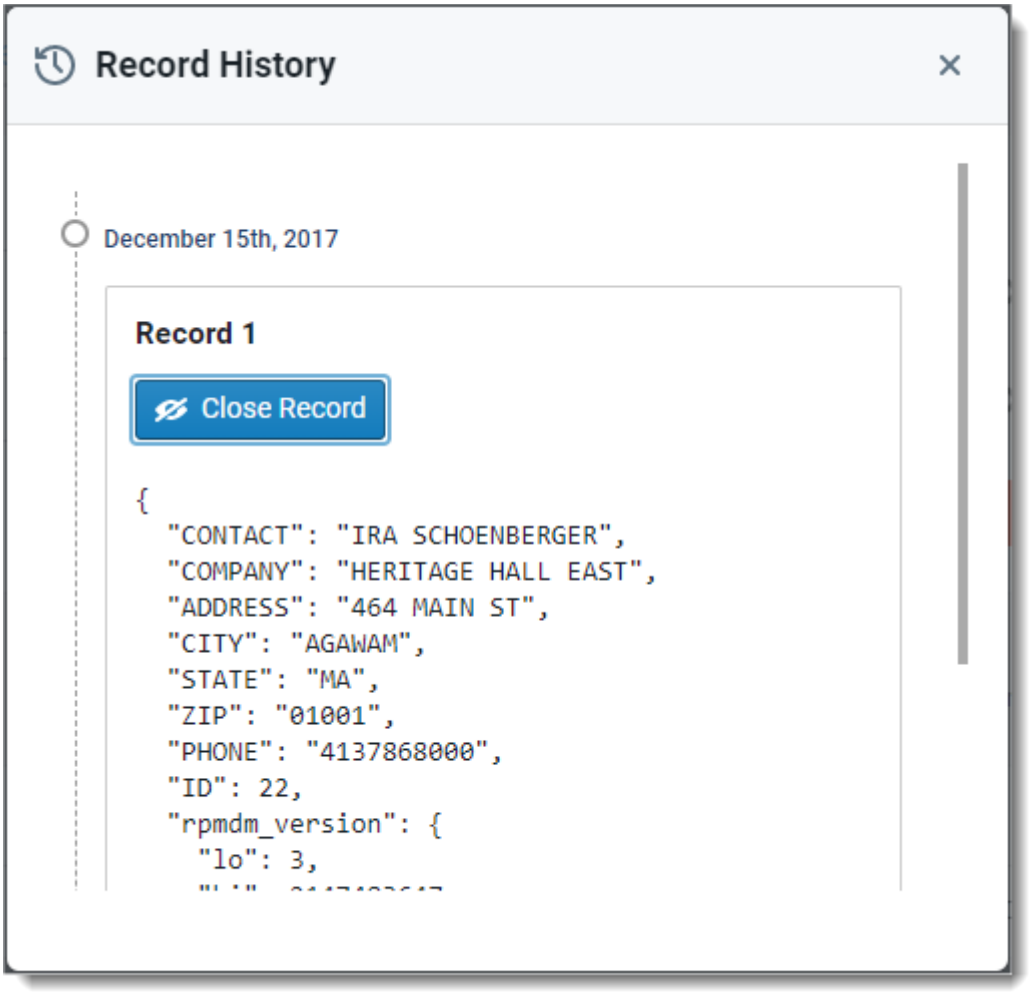
Record View History link

4. The **Record History** dialog displays each version of the record, along with a timestamp.



Record History dialog

5. To view the contents of a particular version of the record, click that version's **View Record** button.



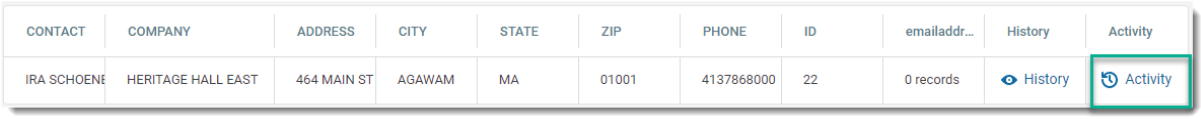
Views a specific record version

3.5. VIEW A RECORD'S ACTIVITY

On the MDM Web App's Data Hub > Data page, you can view the actions performed on a given record. These include the first and last step in the workitems associated with the record, and any other actions performed on the record by a user.

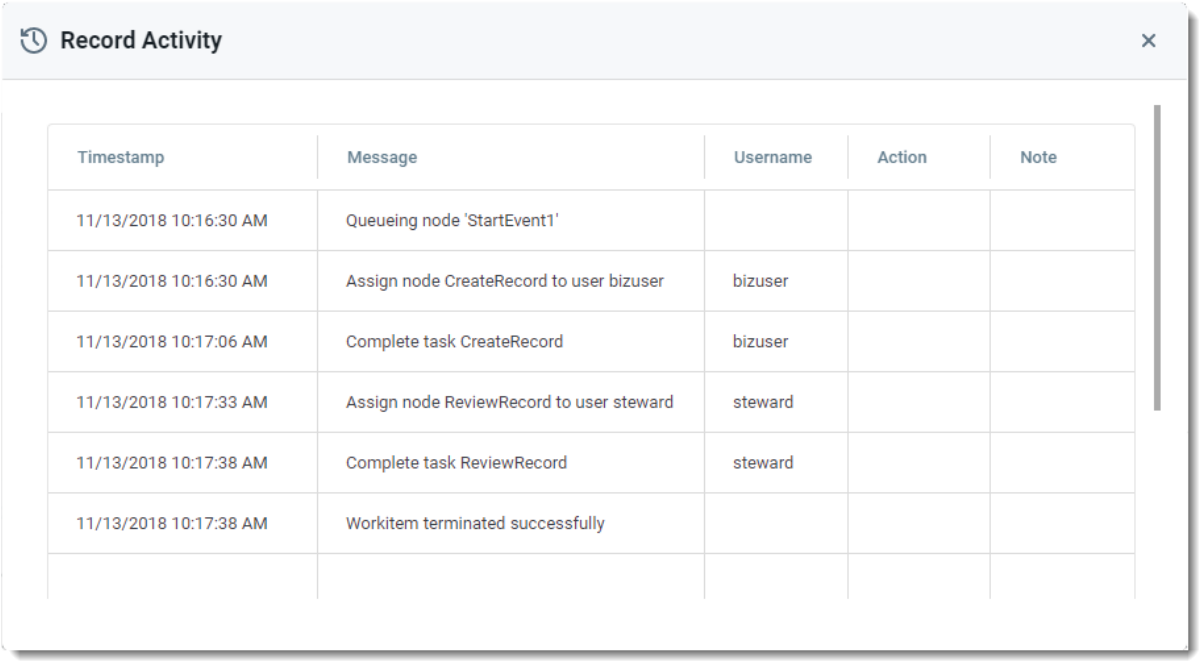
To view a record's activity

1. In the **Data Hub** section of the MDM Web App, click **Data**.
2. Select a table.
3. Find the record of interest (you may have to define a record query), and then click the **Activity** link in its **Activity** column.



Record View Activity link

4. The **Record Activity** dialog displays workitem activities along with the timestamp and username associated with each activity.



Record History dialog

3.6. CHANGE METADATA

The **Data Hub** section allows you to change database and table metadata (the data model).

Note

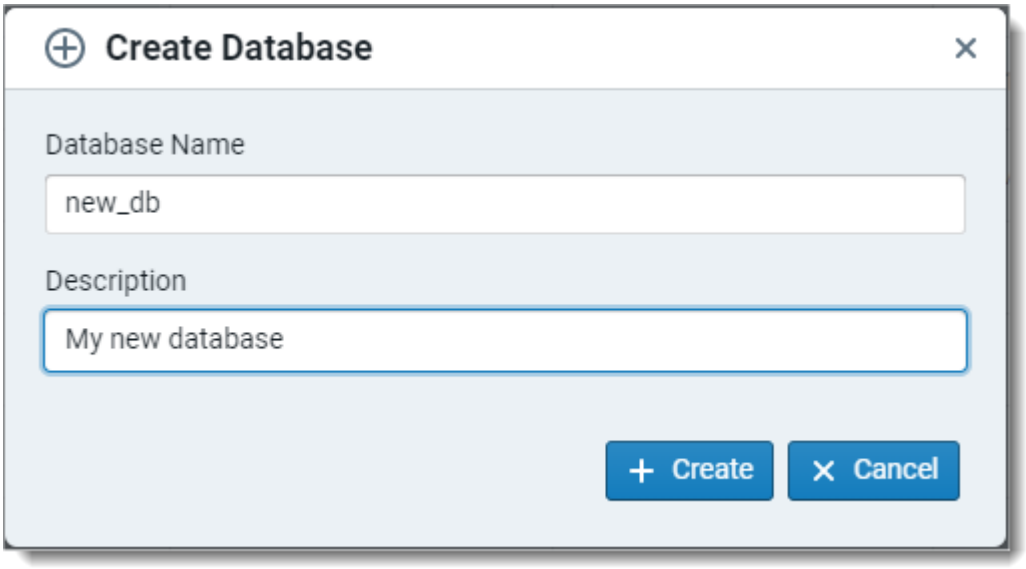
Only users in the **system** or **dbas** groups have authority to do this.

3.6.1. CREATE A DATABASE

To create a database

1. In the **Data Hub** section of the MDM Web App, click **Databases**.
2. On the **Databases** page, click **Create Database**.
3. Enter a **Database Name** and **Description**.

   Database Name must be unique and may not contain spaces. MDM is case-preserving but case-insensitive regarding database names, so "db1" is considered the same as "DB1".



Create Database dialog

3.6.2. DELETE A DATABASE

WARNING: Database deletion is not a versioned operation and **cannot** be undone!

To delete a database

1. In the **Data Hub** section of the MDM Web App, click **Databases**.
2. On the **Databases** page, locate the database you want to delete and click its **Delete** icon .
3. Click **OK** to confirm.

3.6.3. CREATE A TABLE

You can create a new table by defining its metadata interactively, or by importing a previously-defined table schema.

To create a table

1. In the **Data Hub** section of the MDM Web App, click **Tables**.
2. On the **Databases** page, click **Create Table**.
3. On the **Create Table** page, enter a **Name**. Table names must be unique, are case-insensitive, and may not contain spaces.
4. Configure table options.

5. You can add fields to the table now, or add them later by editing the table schema on the **Edit Table** page.
6. Click **Create Table**.

3.6.4. IMPORT A TABLE SCHEMA

You can create a new table by importing a table schema that has been saved as a JSON-formatted file.

To import a table schema

1. In the **Data Hub** section of the MDM Web App, click **Tables**.
2. On the **Tables** page, click **Import Schema**.
3. Enter a **Table Name**, and optionally a **Description**. *The **Table Name** must not already exist in the currently-selected database.*
4. Click **Browse**, and navigate to the desired JSON-formatted schema file.
5. Click **Import**.

3.6.5. DELETE A TABLE

Deleted tables are not truly deleted—they still exist in historical views. In fact, you can delete a table and then re-create a table with the same name, and they will both exist in the database, but at different points in time.

To delete a table

1. In the **Data Hub** section of the MDM Web App, click **Tables**.
2. On the **Tables** page, locate the table you want to delete and click its **Delete** icon  .
3. Click **OK** to confirm.

3.6.6. MODIFY A TABLE SCHEMA

After table creation, you can:

- Modify a subset of the table attributes
- Add fields
- Change the field order
- Delete fields

To modify a table schema

1. In the **Data Hub** section of the MDM Web App, click **Tables**.
2. On the **Tables** page, locate the desired table and click its **Schema** button  .

You cannot modify most of the table attributes after table creation, but you can modify the Enable Redpoint Data Management Matching Support and Display Match Group IDs options and the table Description.

You cannot modify a field's **Name**, **Type**, or **Details** settings after a field has been defined, but you can change the **PII**, **Index**, and **Display** settings.

3. To add a field, click the **Add** button  and configure the new field's options.
4. To remove a field, click the field's **Delete** button  .
5. Click **Update Table** when finished.

Note

Because table metadata is versioned, data model changes can have potentially unexpected effects:

- New fields must be nullable, because when historic records are queried the missing fields will be filled with null values.
- Deleted fields will still exist in historical data. Perform a historic query to see them.
- Deleting and re-creating a field of the same name actually results in two fields—the historic field with historic data, and the new field with new data. Which one you see depends on the timestamp of your query. Historic data is not automatically migrated to the new records. To do that, you must use an ETL process.

3.6.7. EXPORT A TABLE SCHEMA

You can export a table schema and save it as a JSON-formatted file.

To export a table schema

1. In the **Data Hub** section of the MDM Web App, click **Tables**.
2. On the **Tables** page, locate the desired table and click its **Export** button  .
3. Specify the name and location of the schema file, and then click **Save**.

4. WORKFLOW

A *workflow* is a process for performing tasks on a specific database and table. It:

- coordinates tasks between multiple users and the MDM system.
- handles decisions and error conditions.
- allows for entering of notes and feedback.
- gives users with low-level access rights a controlled path to data change.

A workflow *process* typically consists of a series of tasks, often requiring actions by multiple users. An instance of a workflow is called a *workitem*.

For example, the workflow for adding a new customer record might look something like this:

1. User "bizuser" starts a **New Customer** workflow, creating a new workitem.

2. "Bizuser" enters customer data on the workitem's **New Record** page and submits it.

   The workitem appears in the data steward's (user "steward") **Available** queue.

3. "Steward" clicks **Claim** to open the workitem and review the proposed customer record. Noticing a missing **PHONE** field, "steward" adds a comment "Correct missing phone" and rejects the record.

   The workitem appears in the "bizuser" **Workitems Assigned to Me** queue.

4. "Bizuser" clicks the workitem's **Continue** button. On the workitem's **Edit Record** page,"bizuser" reads the steward's comment, enters the customer's phone number, and submits the workitem.

   The workitem appears in the "steward" **Workitems Assigned to me** queue.

5. "Steward" clicks the workitem's **Continue** button to open the workitem and review the edited customer record. Confirming that the missing phone number was added, "steward" approves the record and it is added to the target table.

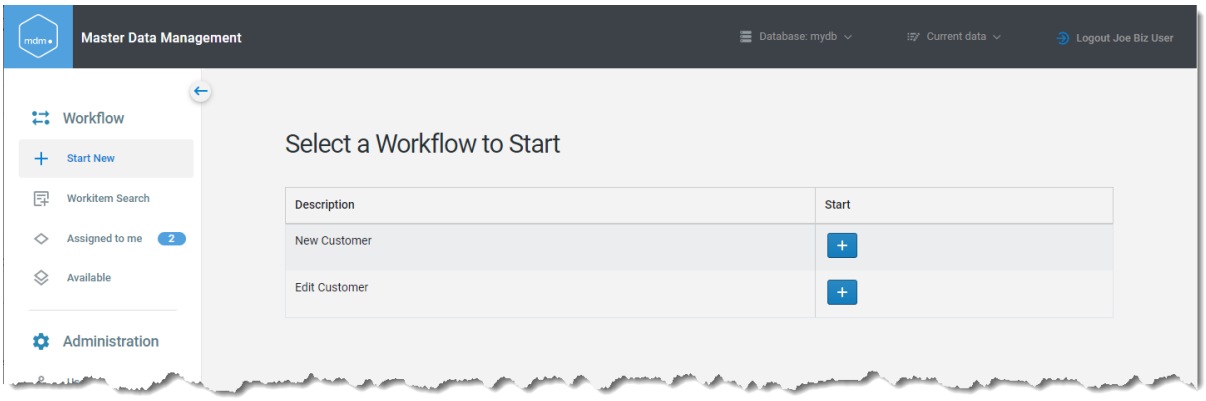
4.1. WORKFLOW WEB APP SECTION

The **Workflow** section of the MDM Web App is where you can do the following:

- Start new workflows. Different access levels allow the creation of different kinds of workflows.
- Search for workitems. You can specify whether to display workitems that are **Active** or **Finished**, and refine the results with additional query filters.
- Open workitems that are assigned to you and require your interaction.
- Claim available workitems. This includes all the workitems assigned to you, as well as all the workitems you have the authorization to claim.
- Create new workflows and processes, and edit existing ones (**dbas** and **system** groups only).
- For some users (especially users with low-level access rights), a workflow is the only way to change a database. Higher-level users (such as those in the **dbas** and **system** groups) can change metadata directly.
- Users in the **dbas** and **system** groups can define processes using BPMN (Business Process Modeling Notation) and create, edit, and delete workflows that implement these processes. See Advanced workflow.

Start New

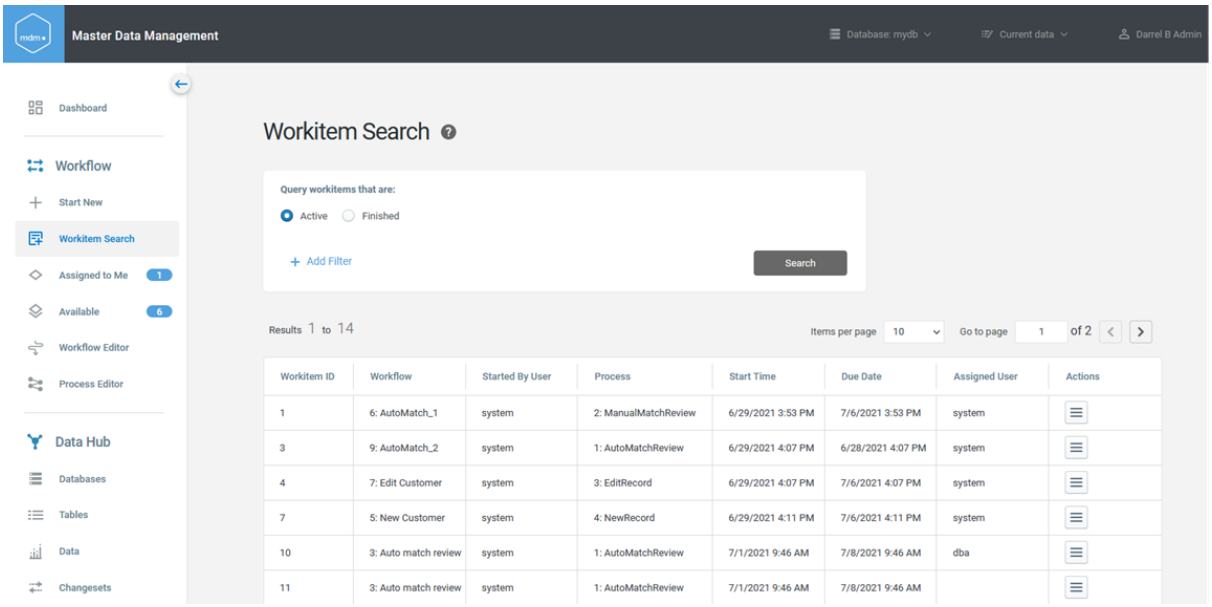
This section lists the available workflows for the currently-selected database. Different databases may have different workflows.



Starting a new workflow

Workitem Search

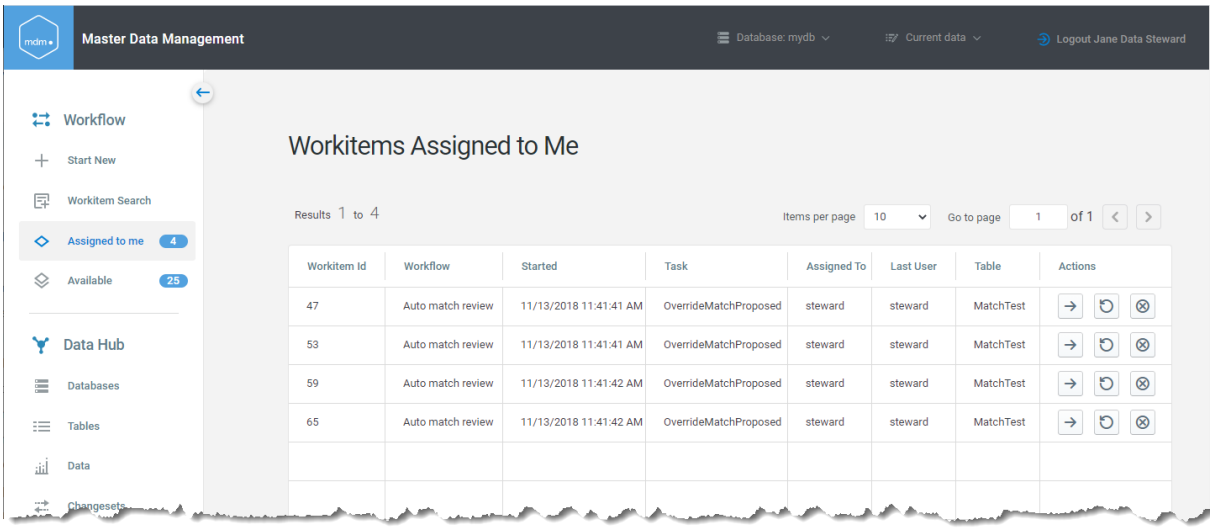
This section lets you search for workitems for the currently selected database. You can use various filter criteria. For details, see Search for workitem. Potential actions include the following: Edit, Delete, and Workitem activity.



Searching for workitems

Assigned to Me

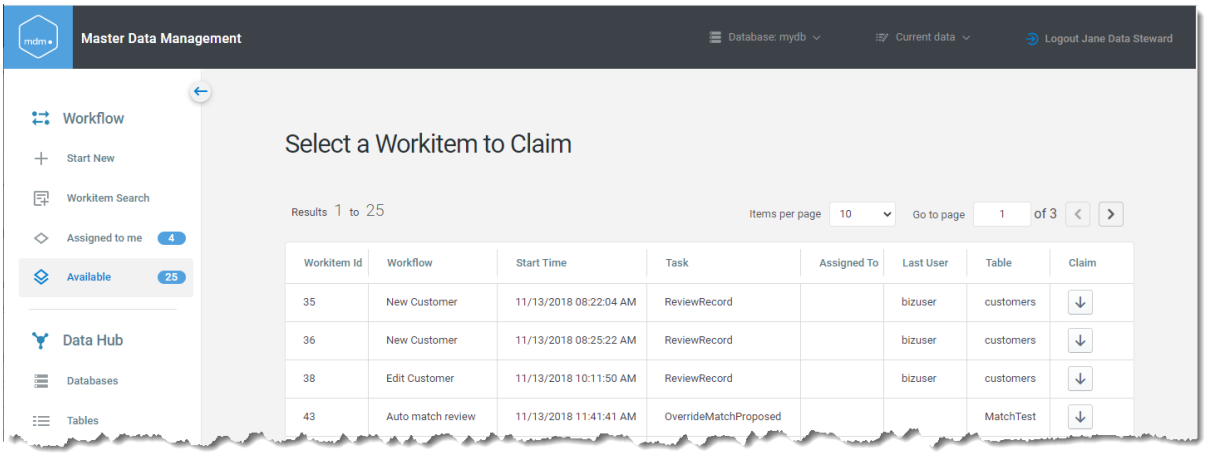
This section displays the workitems assigned to you for the currently-selected database. You may have workitems assigned to you in a different database. Potential actions include the following: Continue, Release, Edit, and Discard workitem.



Displaying workitems assigned to you

Available

This section displays the workitems available for you to claim for the currently-selected database. You may have workitems available to you in a different database. Potential actions include Claim and Edit workitem.



Displaying workitems available for you to claim

4.2. SEARCH FOR WORKITEM

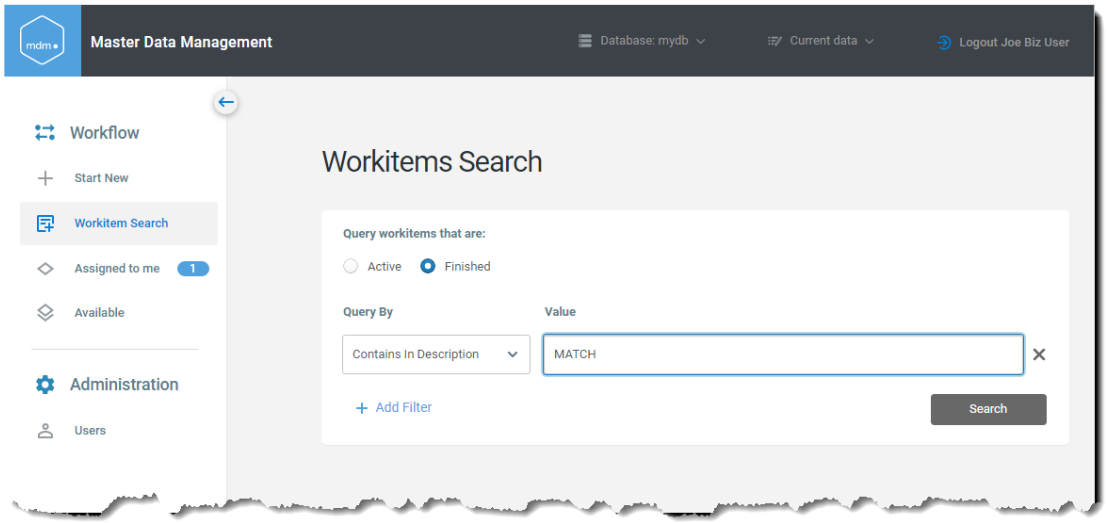
To search for a workitem

1. In the **Workflow** section of the MDM Web App, click **Workitem Search**.
2. Specify whether to find workitems that are **Active** or **Finished**.
3. Optionally, you can refine the results by defining additional query filters:
   a. Click **Add Filter**.
   b. Choose a **Query By** option:
      - **Started By Username**
      - **Is Assigned**

- **Start Timestamp**
- **Contains in Description**
- **Match Whole Description**
- **Due date**
- **Assigned To Username**
- **Available To Username**

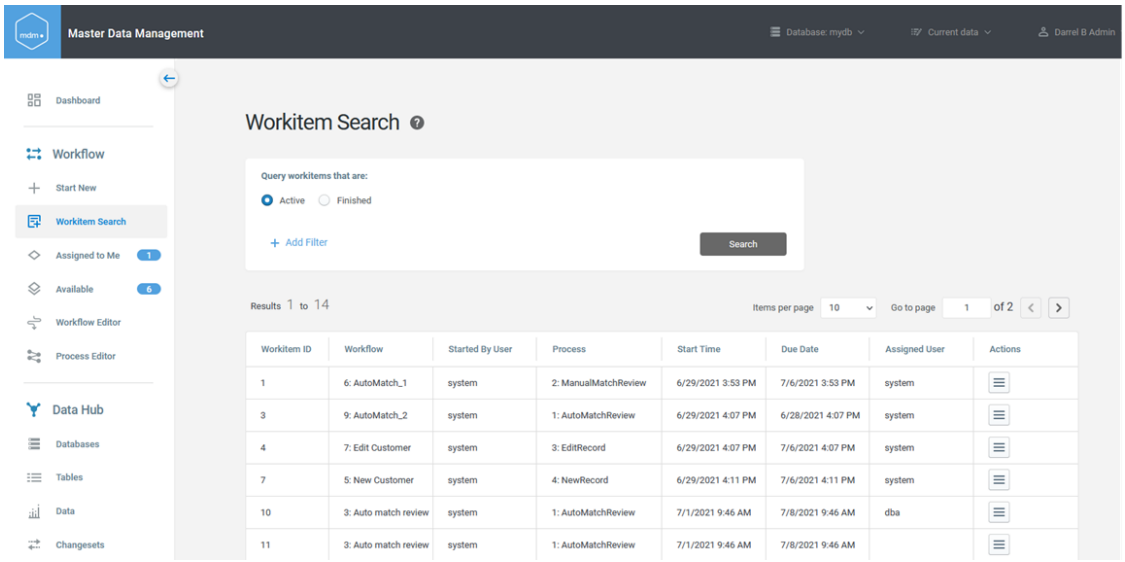
c. Enter or choose a **Value**.

d. Click **Add Filter** to define additional queries, or click **Search** to run the query.



Configuring a query

The search results display.



Search Results

4. Click the icon in the **Actions** column and select the action you want to perform. The actions that display depend on your role.

   - **Edit Workitem**

- **Delete Workitem**
- **Claim**
- **Workitem Activity**

4.3. ENTER A NEW RECORD

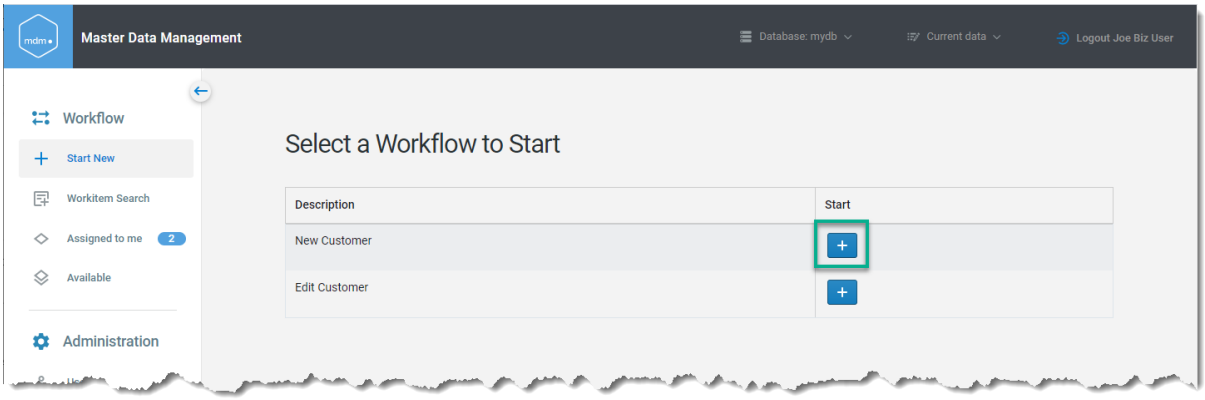
The workflow for entering a new record (creating a new customer record, for example) is as follows:

- Business user starts workflow.
- Business user enters data and submits.
- Steward reviews data and accepts or rejects.

4.3.1. BUSINESS USER WORKFLOW

An example business user workflow for creating a new record:

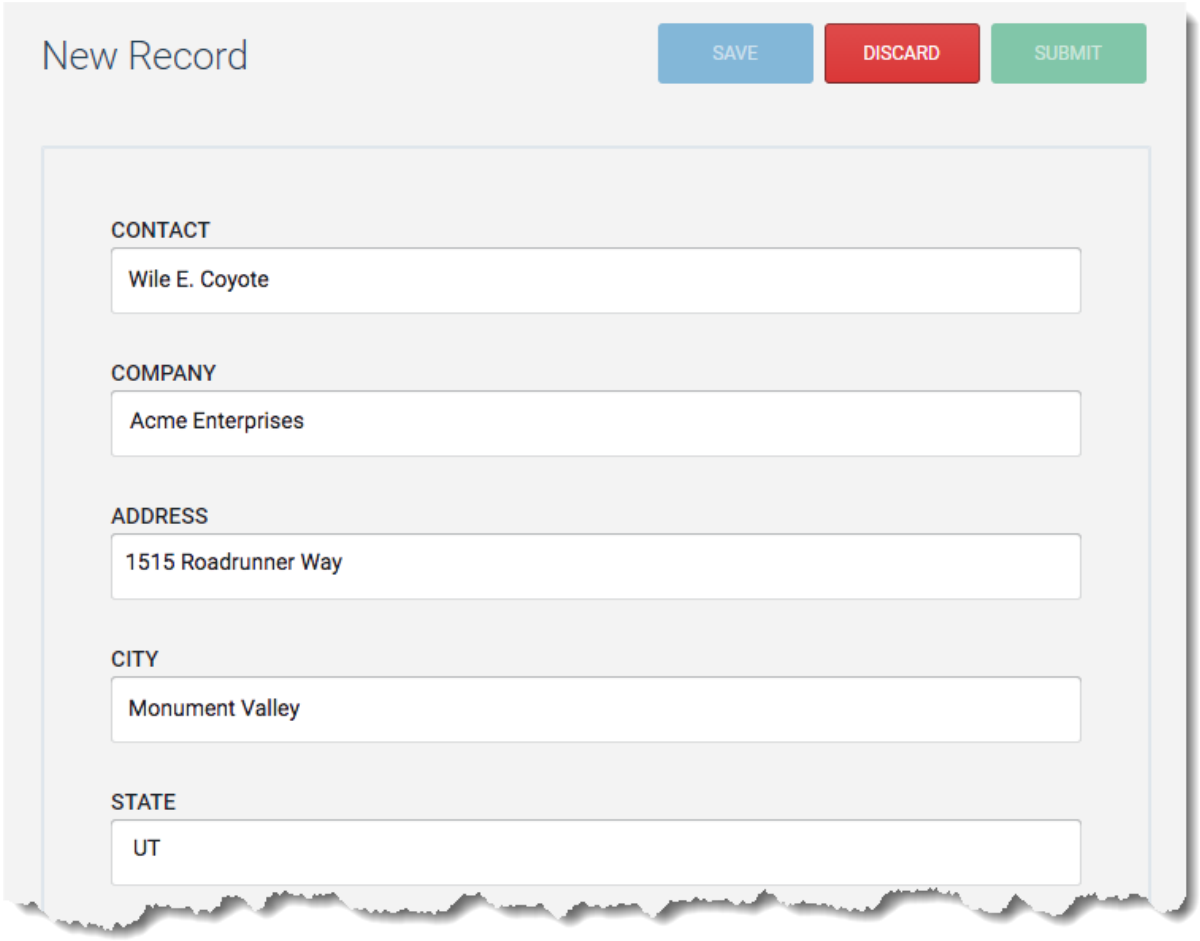
1. In the **Workflow** section of the MDM Web App, click **Start New**.
2. In the **New Customer** row, click the **Start** button **+**.



Starting a new customer record in the Business User workflow

The New Record page displays.

3. On the **New Record** page, enter the customer's data.
4. To automatically start the next workitem in your activity queue, select the **Start next** check box.



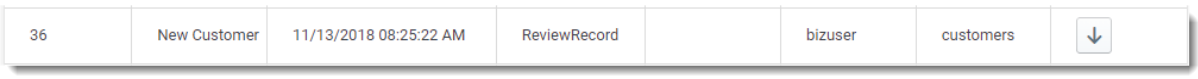
Entering customer data for a new record in the Business User workflow

5. Optionally, click in the **Comments** box and enter a comment.
6. Click **Submit** to place this workitem in the steward's queue for review. You can also **Save** this workitem and edit it later, or **Discard** it.

4.3.2. STEWARD WORKFLOW

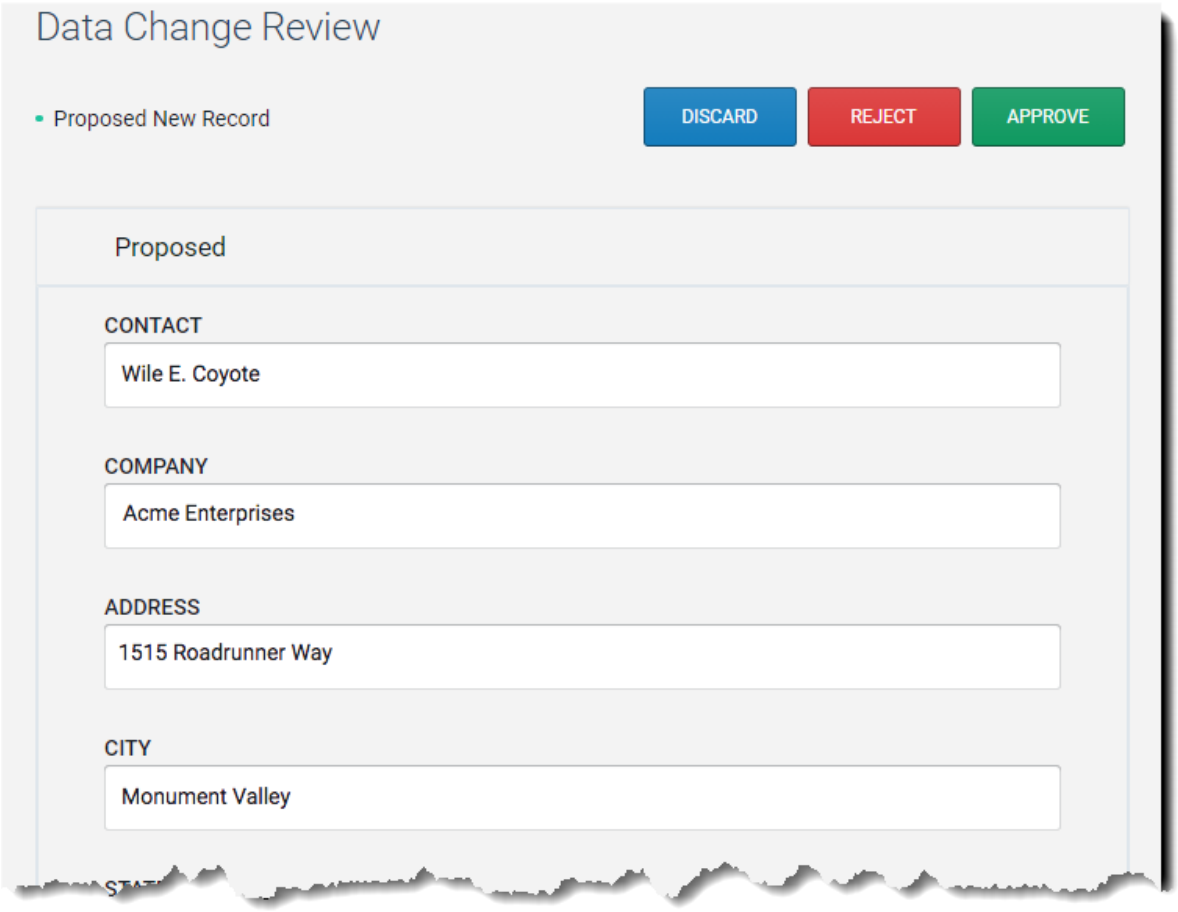
An example steward workflow for reviewing a new record:

1. In the **Workflow** section of the MDM Web App, click **Available**.
2. On the **Select a Workitem to Claim** page, find the workitem containing the record to review and click its **Claim** button . In our case, the workflow is "New Customer", the task is "ReviewRecord", and "bizuser" was the last user (the user who created the record).



Claiming a new record workitem in the Steward workflow

3. On the Data Change Review page, review the proposed record.



Reviewing the proposed new record in the Steward workflow

4. At this point, you can:
    - Leave the record as-is, or make any changes necessary, and click **Approve**.
    - Write a **Comment** to the user who created the record and click **Reject**. The record will be sent back to that user's queue, and the user can make the requested changes.
    - Click **Discard**, which deletes the proposed record.

4.4. EDIT AN EXISTING RECORD

The workflow for editing an existing record (changing a customer record, for example) is as follows:

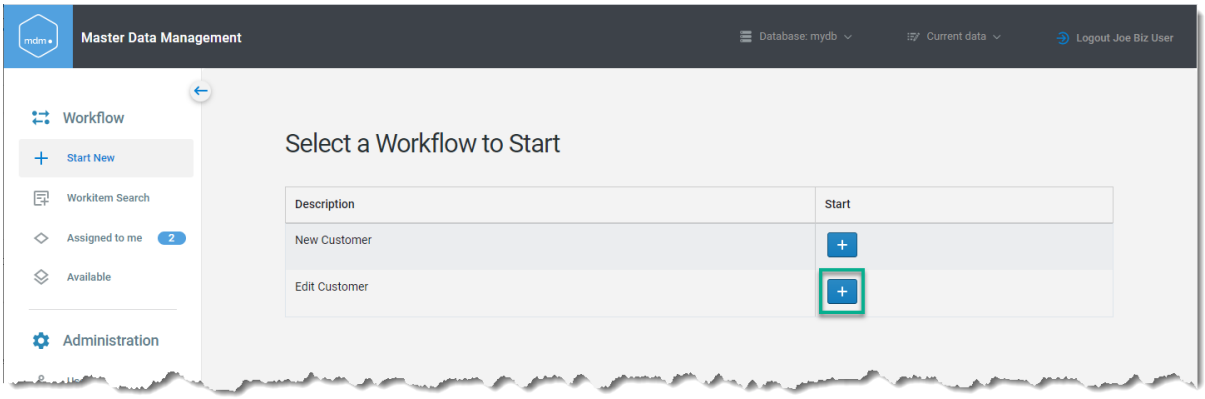
- Business user starts workflow.
- Business user finds record.
- Business user edits data and submits changes.
- Steward reviews changes and accepts or rejects.

4.4.1. BUSINESS USER WORKFLOW

An example business user workflow for editing a record:

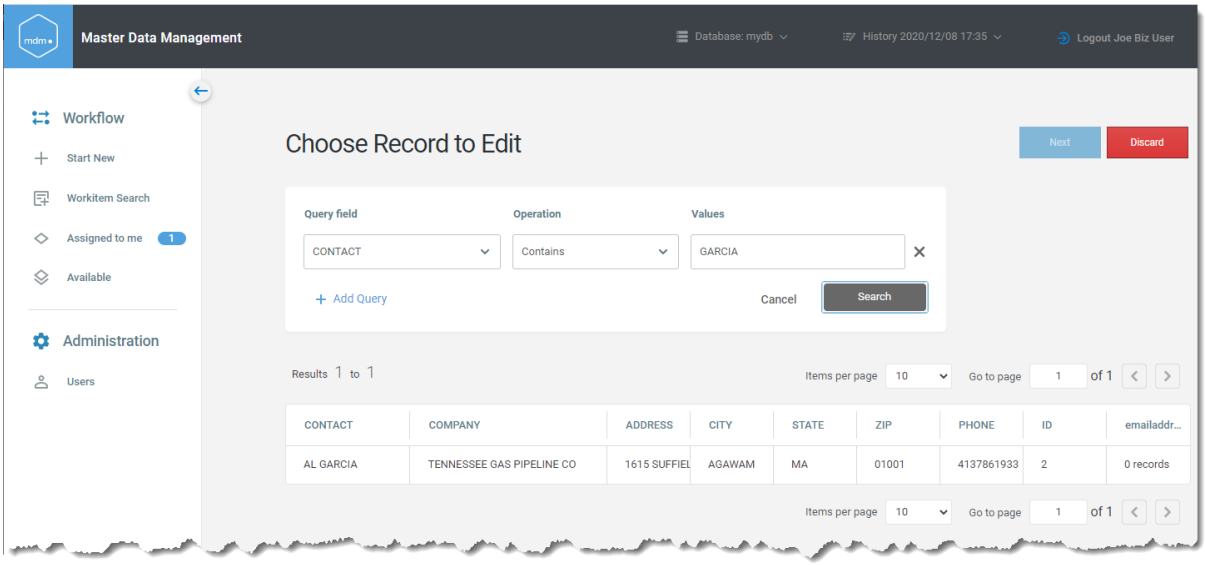
1. In the **Workflow** section of the MDM Web App, click **Start New**.

2. In the **Edit Customer** row, click the **Start** button .



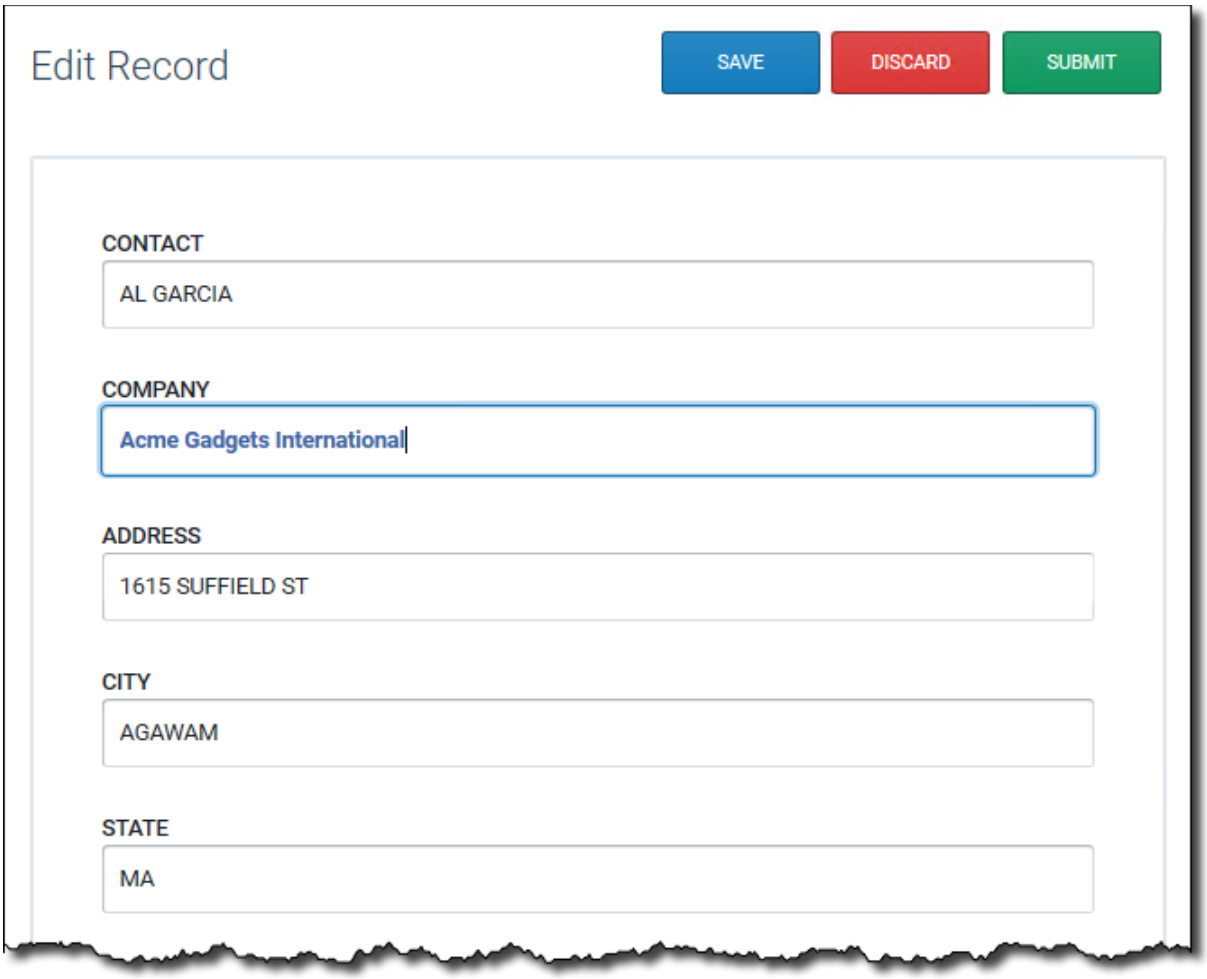
Starting a Business User workflow and editing an existing record

3. On the **Choose Record to Edit** page, construct a query and click the **Search** button.



Choosing an existing record to edit in the Business User workflow

4. Double-click on the record you want to edit.
5. On the **Edit Record** page, edit the customer's data.
6. To automatically start the next workitem in your activity queue, select the **Start next** check box.



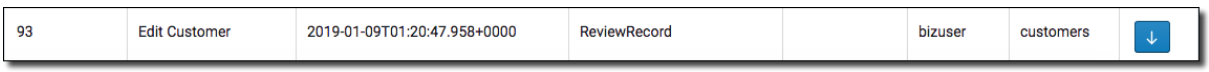
Allows you to edit customer data in an existing record in the Business User workflow

7. Optionally, click in the **Comments** box and enter a comment.
8. Click **Submit** to place this workitem in the steward's queue for review.
   You can also **Save** this workitem and edit it later, or **Discard** it.

4.4.2. STEWARD WORKFLOW

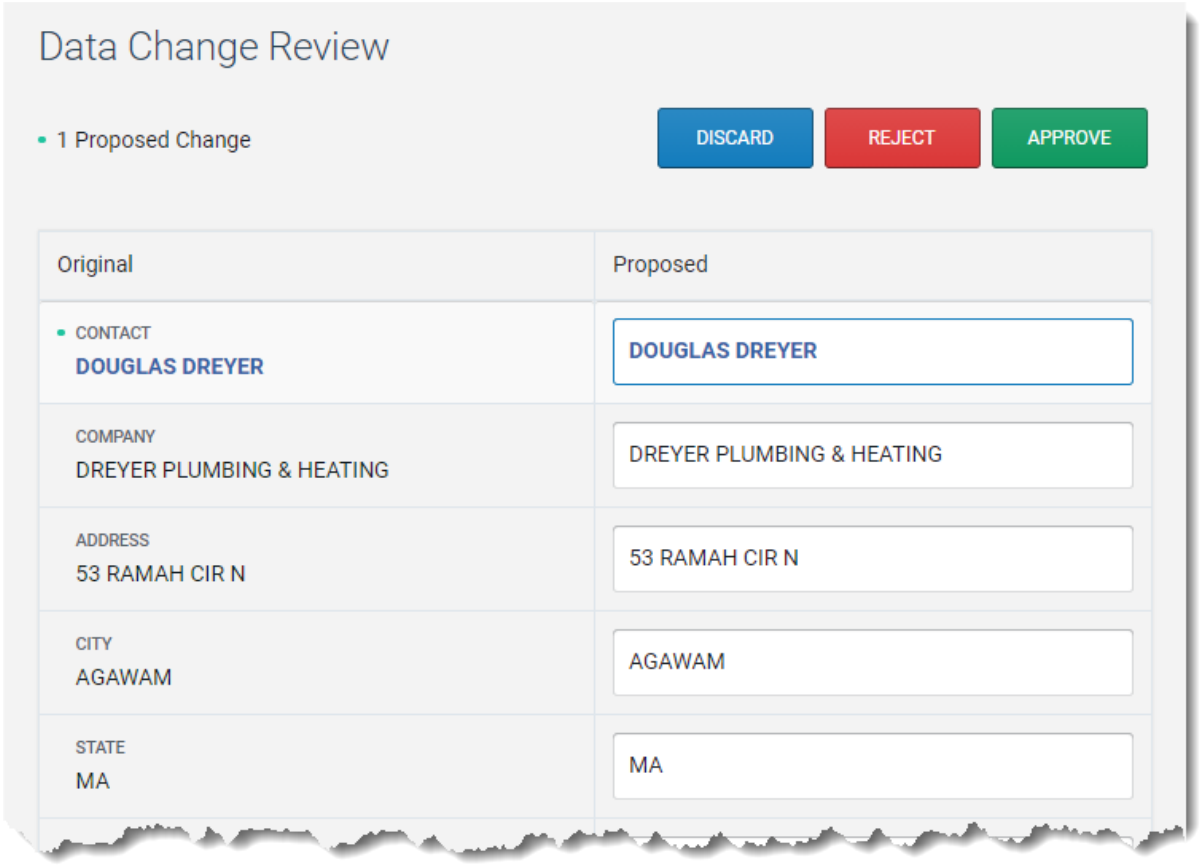
An example steward workflow for reviewing an edited record:

1. In the **Workflow** section of the MDM Web App, click **Available**.
2. On the **Select a Workitem to Claim** page, find the workitem containing the record to review and click its **Claim** button  . In this case the workflow is "Edit Customer," the task is "ReviewRecord," and "bizuser" was the last user (the user who created the record).



Claiming an existing record edit workitem in the Steward workflow

3. On the **Data Change Review** page, review the proposed record changes.



Reviewing proposed record changes in the Steward workflow

4. At this point, you can:

   - Leave the record as-is, or make any changes necessary, and click **Approve**.
   - Write a **Comment** to the user who created the record and click **Reject**. The record will be sent back to that user's queue, and the user can make the requested changes.
   - Click **Discard**, which deletes the proposed record.

4.5. REVIEW AUTOMATED MATCHES

If a Redpoint Data Management ETL process has automatically identified potential duplicate records, you can review these match groups and edit, accept, or reject them. For example, you could use this workflow to review individual match level for new customers.

The Redpoint Data Management ETL process compares both looser and tighter match passes. If these match passes produce different results, the process initiates a MDM review workflow so that a person can accept the looser or tighter match.

You can set the tight and loose match thresholds to any value you wish in Redpoint Data Management. For example, you could set a threshold value of 100% for accepted match groups (meaning a steward shouldn't need to review since the records 100% match). You could then lower the proposed match group threshold to something like 74%, creating a looser match that a steward should review.

The workflow for reviewing matches identified by a Redpoint Data Management ETL process is as follows:

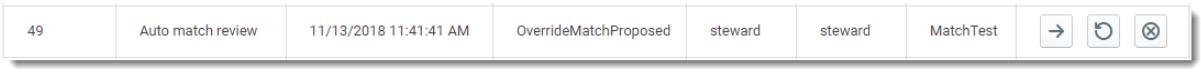
- Steward accepts workflow from the **Available** queue.

- Steward accepts loose or tight match, or customizes match groups.
- Steward commits changes.
- MDM stores steward decisions to prevent automatic matching from overturning decisions.

4.5.1. STEWARD WORKFLOW

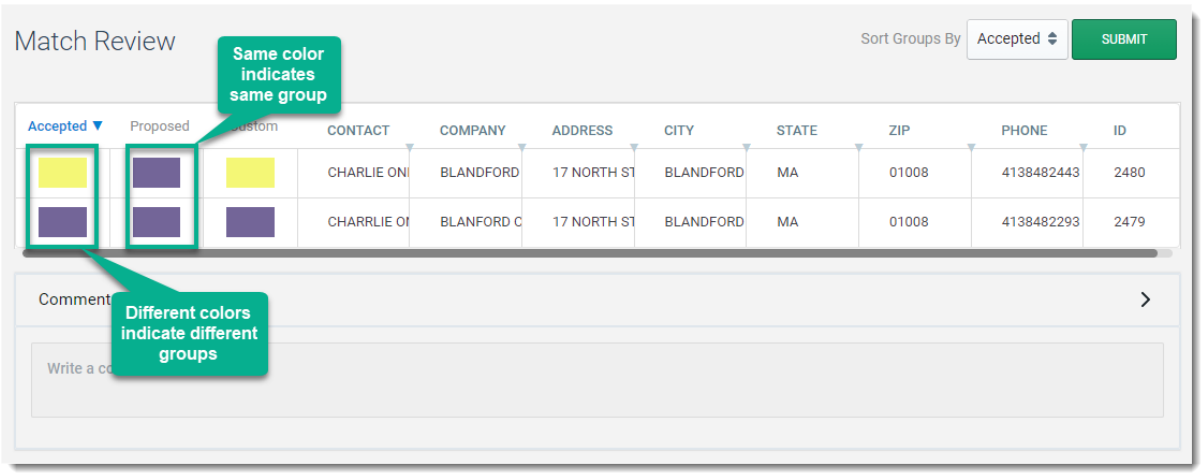
An example steward workflow for reviewing automated matches:

1. In the **Workflow** section of the MDM Web App, click **Available**.
2. On the **Select a Workitem to Claim** page, locate the workitem for the auto match review and click its **Claim** button . In this case the workflow is "Auto match review" and the task is "OverrideMatchProposed".



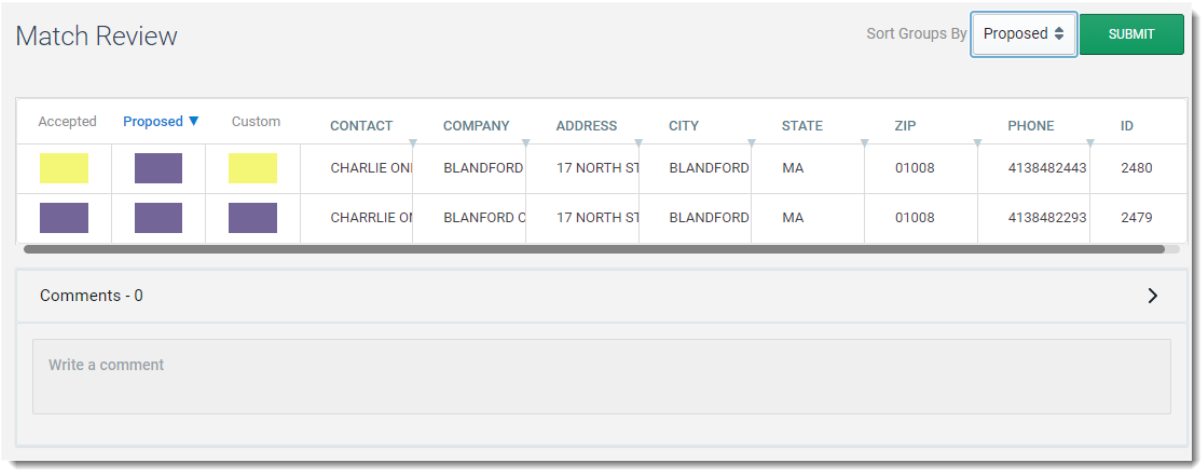
Selecting an auto match review workitem to claim in the Steward workflow

3. On the **Match Review** page, review the proposed matches:
   a. On the **Sort Groups By** menu, select **Accepted** to display the current (tight) match groups:



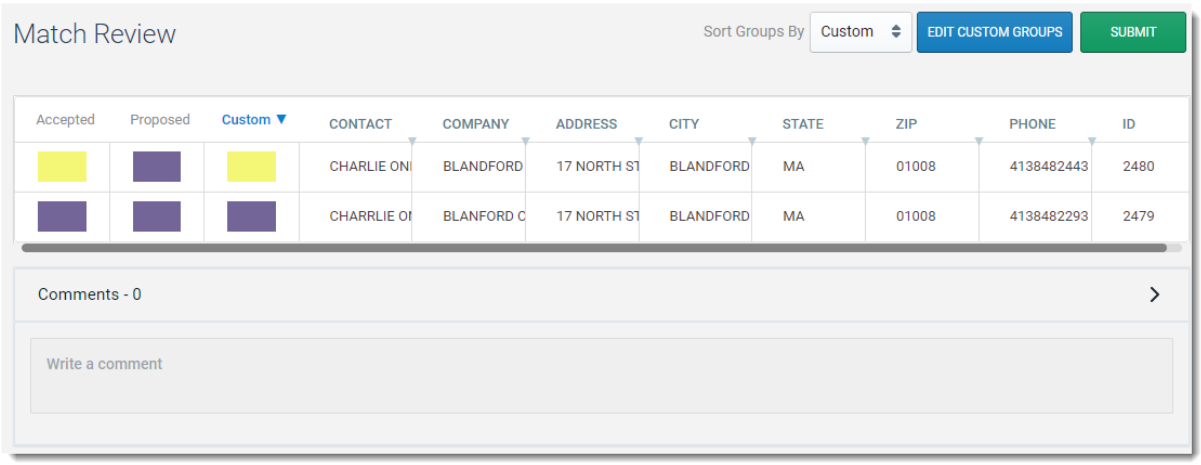
Viewing the current match groups in the selected auto match review (Steward workflow)

   b. On the **Sort Groups By** menu, select **Proposed** to display the proposed (loose) match groups:



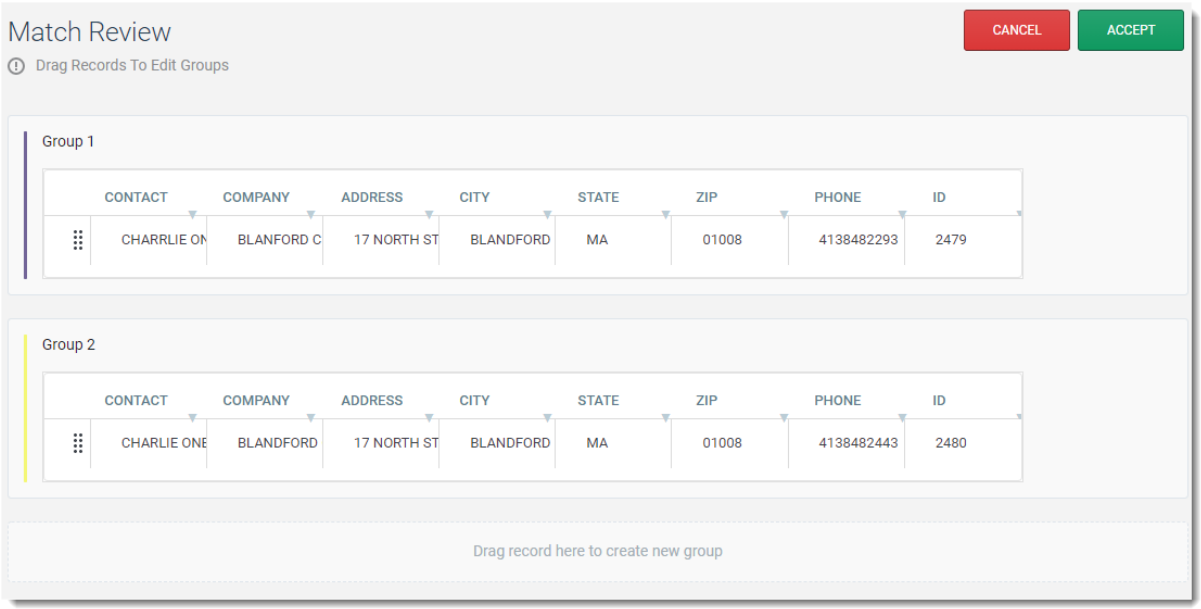
Viewing the proposed match groups in the selected auto match review (Steward workflow)

c. On the **Sort Groups By** menu, select **Custom** to define your own match groups:



Defining your own match groups in the selected auto match review (Steward workflow)

d. Click **Edit Custom Groups** to create your own match groups. When finished, click **Accept.** Click **Cancel** to return to the Match Review page.



Defining your own match groups in the selected auto match review (Steward workflow)

4. Click **Submit** to approve the new match groups. MDM will record match group overrides for future matching. Note that automatic match group review cannot be discarded.

4.6. REVIEW MANUAL MATCHES

You can manage duplicate records in a table by querying the table to identify likely duplicate records and manually grouping them into match groups. For example, this workflow allows you to review and change current customer matches at individual match level.

The workflow for reviewing manual matches is as follows:

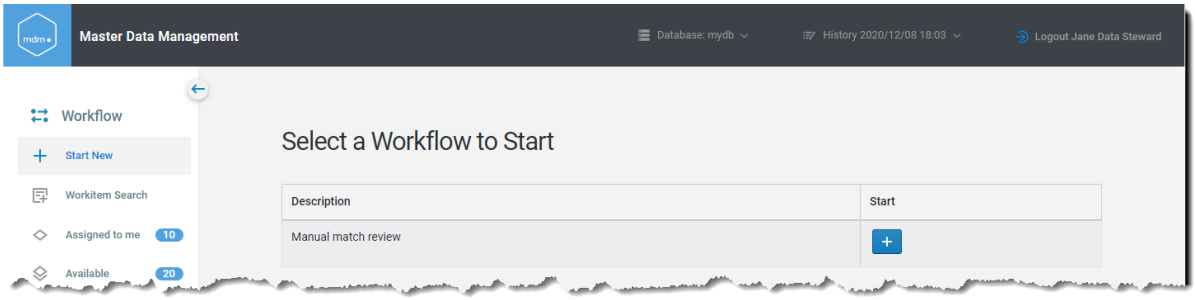
- Steward starts a new workflow
- Steward searches for existing records and match groups
- Steward recombines into correct match groups

- Steward commits changes
- MDM system stores steward decisions to prevent automatic matching from overturning decisions

4.6.1. STEWARD WORKFLOW

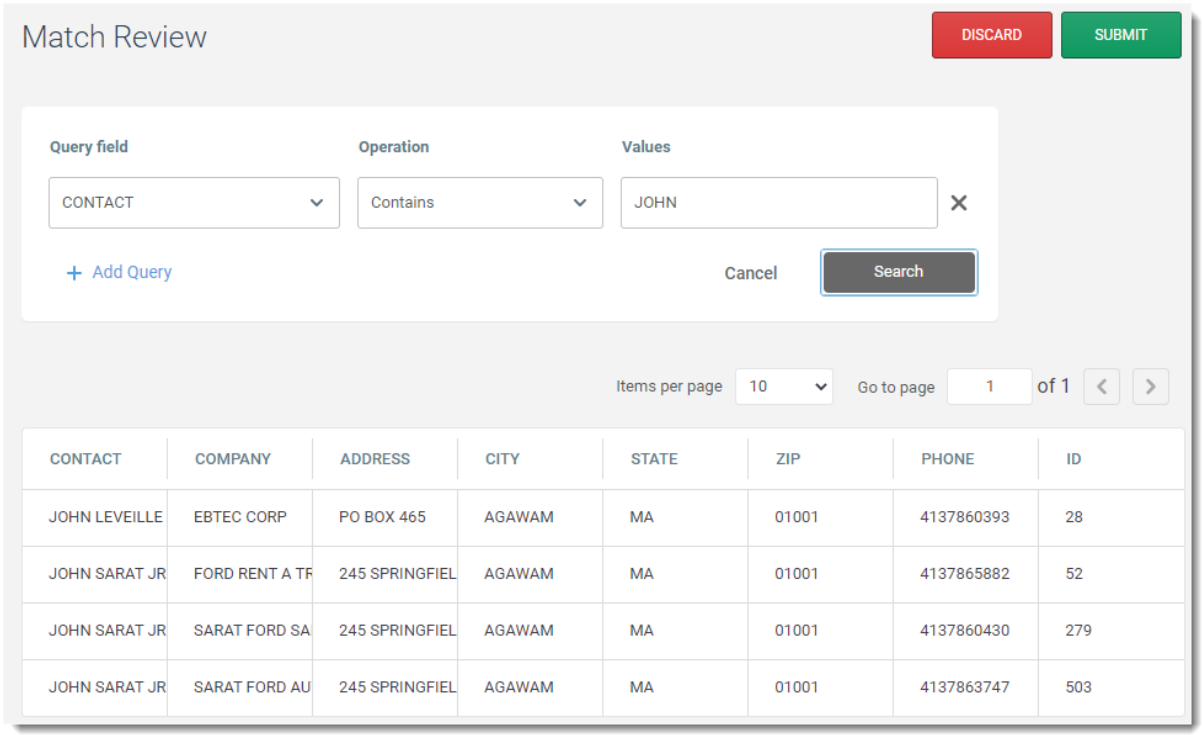
An example steward workflow for reviewing manual matches:

1. In the **Workflow** section of the MDM Web App, click **Start New**.
2. In the **Manual match review** row, click the **Start** button **+**.



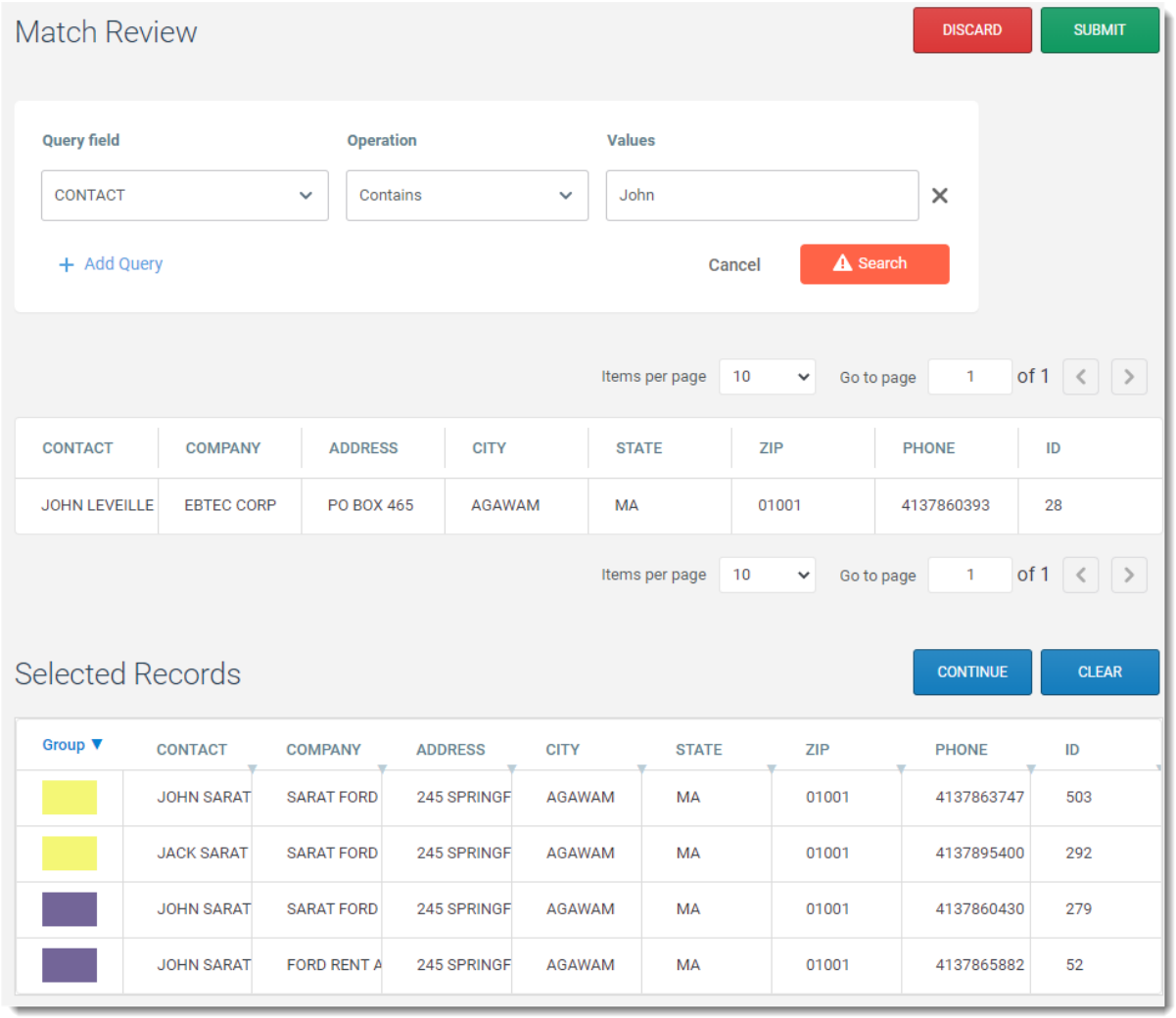
Starting a new manual match review workflow (Steward workflow)

3. On the **Match Review** page, construct a query to find the records you want to review, and click the **Search** button.



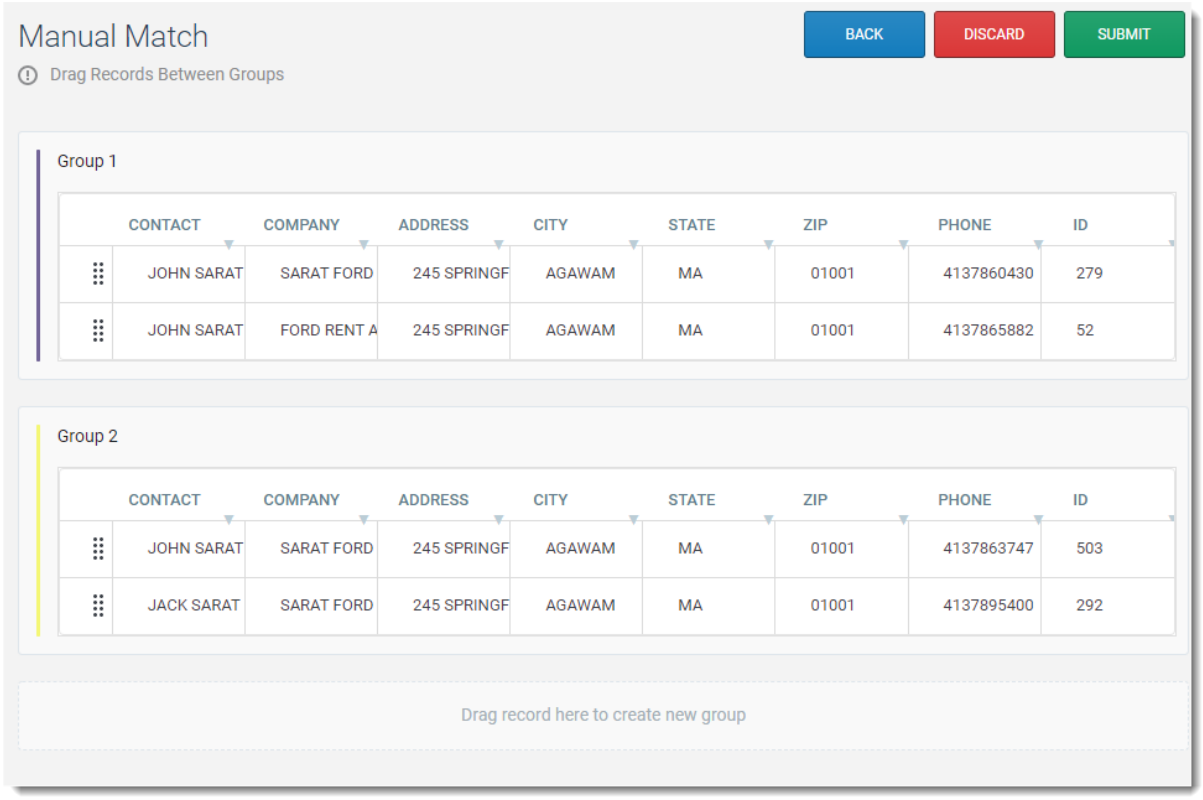
Constructing a query to find the records you want to review (Steward workflow)

4. Double-click on each record you want to review. This brings this record *and all of the records in its current match group* into the selected records list.

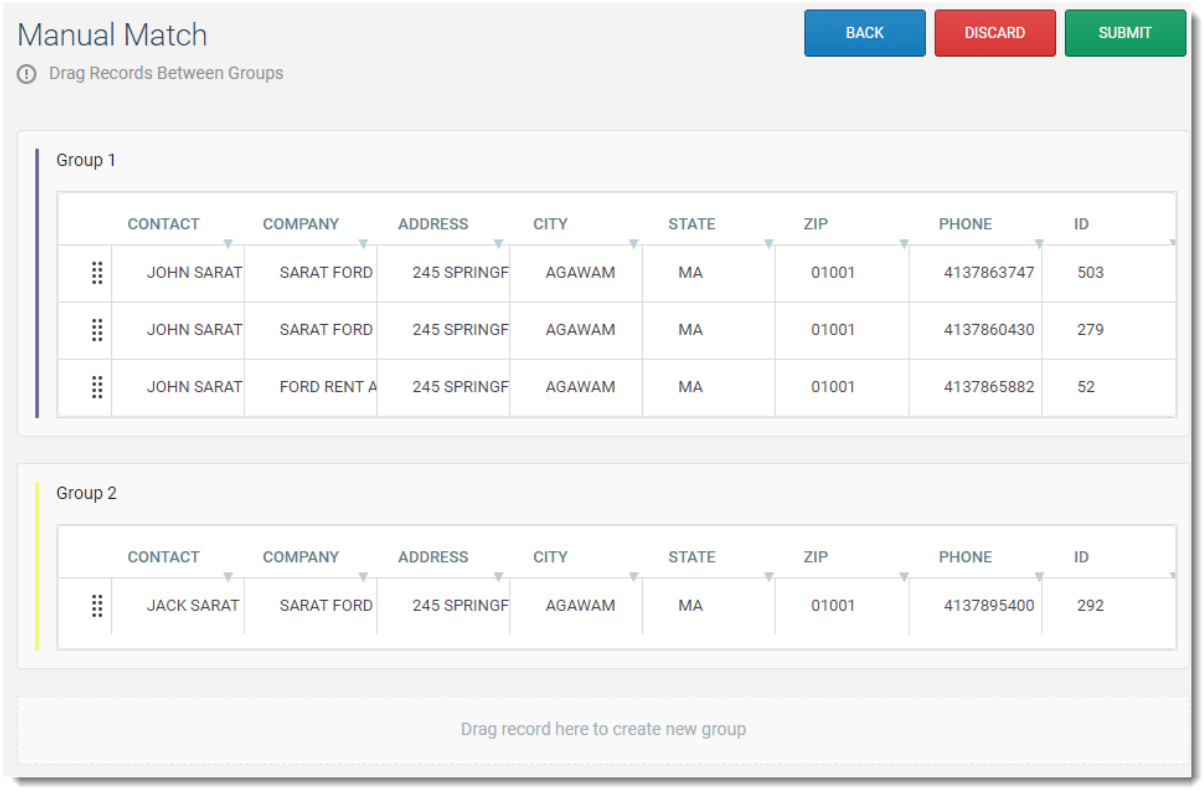


Selecting the records you want to review (Steward workflow)

5. When you have added all the desired records to the **Selected Records** group, click **Continue**.
6. On the **Manual Match** page, drag and drop matching records into the same group. You can create multiple groups of matching records. MDM displays previously-defined match groups (if any exist).



Moving matching records into the same group (Steward workflow)



Approving new manual match groups (Steward workflow)

7. Click **Submit** to approve the new match groups. MDM will record match group overrides for future matching.
   You can also click **Back** to return to the **Match Review** page, or **Discard** to abandon the match review workflow entirely.

5. REDPOINT DATA MANAGEMENT'S MDM TOOLS

MDM uses a set of Redpoint Data Management tools to perform "back end" ETL and matching functions. These tools require a license and are intended for use with Redpoint Master Data Management.

For a more general explanation of Redpoint Data Management tools, refer to the *Redpoint Data Management User Guide*.

To install MDM tools in Redpoint Data Management

1. Copy the Redpoint-provided "shaded" version of the MDM tools JAR file into the `\java_plugins` folder in the Redpoint Data Management installation folder. The filename is of the form `mdm-connectors-*-shaded.jar`.
2. In the Redpoint Data Management client, click the **Palette** menu button at the top of the tool palette ☰ and select **Reset Palette**.
   The MDM tools will appear in the **Master Data** tool group.
3. Click the **Repository** tab and go to **Settings > Tools**.
4. Click the **MDM** tab and enter your server and authentication credentials.
5. Close and re-open any projects that were active before installing the MDM tools JAR file.

5.1. MDM TOOL CONNECTION SETTINGS

Redpoint Data Management's MDM tools use shared settings, which allow you to define a single set of configuration properties (typically access credentials) to share across multiple tools in your Redpoint Data Management site. You can override these settings on a per-tool basis by opening the **Connection settings** section on the tool's Properties pane, selecting **Override**, and specifying values for that specific tool.

To define MDM shared tool settings

1. Open the **Tools** folder under **Settings** in the repository.
2. Click the **MDM** tab, and then view the Properties pane.
3. Configure the tool properties for your environment:

MDM server URL	URL for the MDM server, typically a MongoDB server URI. A default "local" install of MDM will use `http://localhost:9902/mdm`.
Authentication server URL	URL for the MDM authentication server. A default "local" install of MDM will use `http://localhost:9901/mdm`.

Username	User logon name. The `system` user is recommended because it has privileges to do anything.
Password	User logon password. The default password for the `system` user is "system".
Override webservice timeout	If selected, override the standard connection timeout of 60 seconds.
Webservice wait seconds	If **Override webservice timeout** is selected, the new connection timeout value (in seconds).

Tips

If you change the shared settings, you may need to close and re-open any active projects for them to take effect.

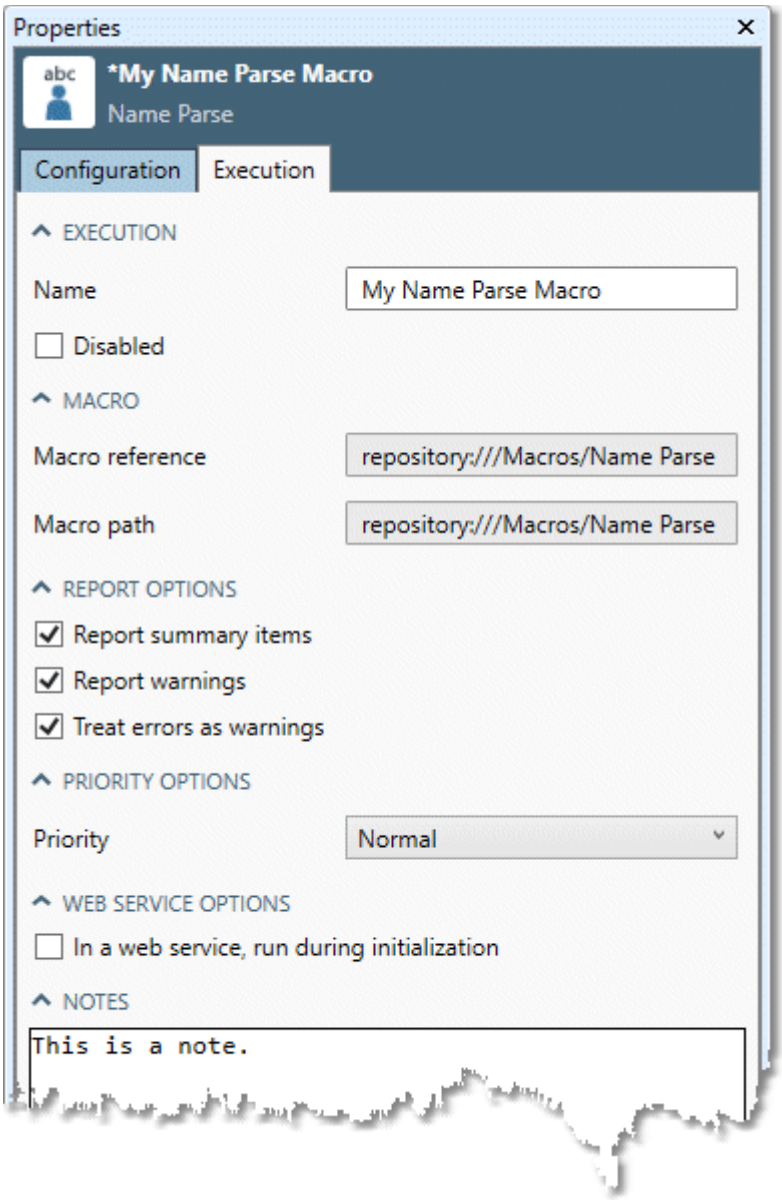
To configure default shared tool settings from an MDM tool's Properties pane, open the **Connection settings** section, and click **Edit default settings**.

To override MDM tool shared settings

1. Select the desired MDM tool, and then click the **Configuration** tab on the Properties pane.
2. Open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.2. MDM TOOL EXECUTION OPTIONS

When you configure Redpoint Data Management's tools and macros, you can click the **Execution** tab on their **Properties** pane and specify the tool name and how to report run-time details on the results of the tool's processing. These details appear in the Message Viewer.



Tool execution options

- All tools and macros have a **Disable** option, which you can access on the Execution tab or on the context menu.
- If the object is a macro, **Macro reference** defines the relative path to the macro, while **Macro path** defines the absolute path to the macro.
- Select **Report summary items** to report the results of the tool's process—for example, the number of records read from an input file.
- Select **Report warnings** to report minor problems as warning messages in the Message Viewer. If this option is *not* selected, problems that do not result in the project being aborted will not be reported.
- Select **Treat errors as warnings** if you want errors to be reported as warnings rather than causing the project to fail. If this option is *not* selected, errors will cause the project to be aborted, and will be reported as error messages in the Message Viewer.

5.3. TRIGGER INPUT AND OUTPUT

Many Redpoint Data Management MongoDB and MDM tools have an **Enable trigger output** option. When selected, this option causes the tool to sprout a T "Trigger" connector. A single record will be sent to this connection after all operations are complete. This can be used to sequence other operations by connecting to downstream tools that support a Trigger input connector. Selecting **Enable trigger input** in these tools causes the tool to sprout the T input connector, which can be used to make the tool wait for previous tools to complete before starting.

5.4. CONFIGURE SHARED SETTINGS

Redpoint Data Management's MDM tools use shared settings, which allows you to share a single set of default configuration properties (typically access credentials) across multiple tools in your site. You can override these settings on a per-tool basis by checking **Override** on a tool's Properties pane, and specifying values for that specific tool.

To define shared tool settings

1. Open the **Tools** folder under **Settings** in the Redpoint Data Management repository.
2. Click the tab for the desired tool name, and then view the Properties pane.
3. Configure the default tool properties for your environment.

Tip

To configure default shared tool settings from a tool's Properties pane, open the **Connection settings** or **Shared settings** section, and click **Edit default settings**.

5.5. MDM INPUT TOOL

This tool reads records from the MDM database.

- By default, the tool reads the current database "view."
- You can specify a changeset or timestamp to read from a database view at a specific point in history.
- You can specify a query filter to reduce the record set.
- Options to read the version, conflict, and matching sub-documents are available. These documents are produced as JSON strings.

5.5.1. MDM INPUT TOOL PARAMETERS

The MDM Input tool has four sets of parameters in addition to the standard execution options: *Configuration*, *Filters*, *Select*, and *Options*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database from which records will be read.

Table	The name of the table from which records will be read.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Select data from	Specifies which version of the data to read records from: • **Current**: the most recently-committed data. • **Timestamp**: the latest changeset committed before a specified timestamp. • **Pending changeset**: a specified pending changeset.
View timestamp	If **Select data from** is **Timestamp**, specifies read data as the table existed at that moment in time.
Changeset	If **Select data from** is **Pending changeset**, specifies the changeset.

Filters

Field filter terms	Use the grid to filter the desired input by defining **Field** - **Operation** - **Values** expressions. For example: `ZIP = "91520"` The terms in the grid are logically ANDed together.
Enable prior days limit	If set, only choose records which have occurred within the specified number of days prior to today.
Days prior to view date	If **Enable prior days** limit is set, limit records returned to those that were created within the specified date range. This is used for timeseries tables.
Enable recent commit filter	Limits input to records committed within the defined time period.
Committed within last	If **Enable recent commit filter** is selected, use the time picker to select the time period.

Select

Include match document	Select to return a JSON document containing match information for each record.
Include match group fields	Select to return the value of the match group id(s).
Select fields	Use this option to choose which fields are returned by the query tool.

Options

Limit records	If selected, limits the number of records read.
Record limit	If Limit records is selected, specifies the number of records to be read.
Block size	Size of a record block. Default is 250. This is a tuning parameter. Generally for narrow records you can make this larger and for wide records you should make this smaller.
Enable trigger input	If selected, this option causes the tool to sprout a T "Trigger" connector. A single record will be sent to this connection after all operations are complete. This can be used to sequence other operations by connecting to downstream tools that support a Trigger input connector. See Trigger input and output.

5.5.2. CONFIGURE THE MDM INPUT TOOL

Before configuring an MDM Input tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Input tool

1. Select the MDM Input tool, and then click the Configuration tab on the Properties pane.
2. Optionally, override shared settings: open the Connection settings section, click Override, and then specify new values for the tool.
3. Click **Connect/Refresh** to validate the connection and populate the database list.
4. Select a **Database** and **Table**.
5. Click **Select data** from and choose the desired view of the data:
   - **Current**: The most recently-committed data.
   - **Timestamp**: Data committed on or before the defined Timestamp.
   - **Pending changeset**: Data from the selected Changeset.
6. Optionally, click **Refresh sample data** to view a sample of the input data.
7. Optionally, click the **Filters** tab and define filters for the input data:
   a. Use the **Field value** filter grid to define one or more **Field** - **Operation** - **Values expressions**.
   b. Click **Enable recent commit filter** to configure a **Committed within** last time value.
8. Optionally, click the Options tab and configure processing options:
   a. Click **Limit records** and specify a **Record limit**.

b. Click **Return match info** to return a JSON document containing match information for each record.

c. Change **Block size** from the default value of 100.

d. Enable **Trigger input**.

9. Optionally, click the Execution tab, and then set **Report** options, **Priority** options, and **Web** service options.

5.6. MDM OUTPUT TOOL

This tool inserts, upserts, or modifies records in the MDM database. It can create a table if one does not yet exist. It cannot create or modify matching information (this is handled by a separate tool).

5.6.1. MDM OUTPUT TOOL PARAMETERS

The MDM Output tool has the following sets of parameters in addition to the standard execution options: *Configuration, Options, Create, and Mapping.*

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database to which records will be written.
Table	The name of the table to which records will be written.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.

Options

Processing mode	Specifies how Redpoint Data Management will output records to the target **Database** and **Table**. • **Create**: Inserts new records into the table. If a record with the same key field value already exists (or if there is another unique index violation), the operation fails and the project aborts. • **Upsert**: Attempts to insert new records into the table. If a record with the same key field value already exists (or if there is another unique index violation), the existing record is replaced by the new record, otherwise a new record is inserted. • **Replace**: Replaces each existing record with a new one. If records with the same primary key

	do not exist, the operation fails and the project aborts.
Block size	Size of a record block. Default is 100. This is a tuning parameter. Generally for narrow records you can make this larger and for wide records you should make this smaller.
Use shared changeset	All MDM tools with this option selected cooperate around a single changeset. Pending changes across all such tools are gathered into a single shared changeset. When the last MDM tool in the project with this option selected finishes, the shared changeset is committed.
Wait for changeset commit	Select this option to wait for the changeset to commit at the end of the tool's processing. If not selected, the changeset commit will be requested but the tool/project will not wait for it to complete.
Commit wait seconds	Time to wait for a changeset commit to complete. If the commit does not complete within this time, the project fails and aborts; however, the commit will continue to process.
Threads	Degree of parallel processing desired. Default is 3.
Enable trigger output	If selected, this option causes the tool to sprout a T "Trigger" connector. A single record will be sent to this connection after all operations are complete. This can be used to sequence other operations by connecting to downstream tools that support a Trigger input connector. For more information, refer to Trigger input and output.
Clear table before loading	If selected, deletes all existing records from the table before writing to it. This is usually appropriate only for development environments.
Use unversioned changesets	If selected, this option causes the changes to overwrite the current version of the record rather than create a new one. This is only meant for multi-step system processes.

Create

This tab allows you to specify how tables are created, and to create and delete them interactively.

Create from	• **JSON**: Table will be created from JSON schema that you specify below. • **Upstream**: Table will be created to mimic the schema of the input connector. Not all schema features are available when using this option.

Actions	• **Create table**: Create a table. Will fail if the table already exists. • **Delete table**: Delete a table (you will be asked to confirm). Note that when you delete a table, the historic table data is retained and can be queried using a historic view. • **Refresh**: Refresh the table list. • **Make JSON from upstream**: Create JSON that mimics the upstream connector schema, that you can then edit to add MDM-specific features such as indexes, sub-tables, and so on.
Enter JSON schema	If **Create table from** is **JSON**, the JSON schema to be applied when creating the table.
Primary key field	If **Create table from** is **Upstream**, specify the primary key field for the new table.
Table supports matching	If **Create table from** is **Upstream**, whether the new table supports matching.
Choose display fields	If **Create table from** is **Upstream**, the fields to be displayed by default in the MDM web application. These are typically the "common" or "identifying" fields.

Mapping

Mapping type	• **Implicit**: Mapping between upstream fields and MDM fields is based on case-insensitive name. • **Explicit**: Mapping between upstream fields and MDM fields is entered here manually.
Actions	If **Mapping** type is **Explicit**, specify how to map input fields to MDM fields: • **Refresh**: Read the MDM table schema again. • **Default**: Match up field names and create entries. • **Clear**: Clear all mapping.
Field mapping grid	Displays the mapping of **Input** fields to **MDM** fields and allows editing of that mapping.

5.6.2. CONFIGURE THE MDM OUTPUT TOOL

Before configuring an MDM Output tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Output tool

1. Select the **MDM Output** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.
3. Click **Connect/Refresh** to validate the connection and populate the database list.
4. Select a **Database** and **Table**.
5. Choose a **Processing mode**:
6. Configure **Use shared changeset**, **Wait for changeset commit**, and **Commit wait seconds** options to specify how the changeset is handled.
7. To enable parallel loading, set **Threads** larger than 1.
8. To support sequencing of later tools after the load is complete, check Enable trigger output.
9. Optionally, select **Clear table before loading**. This is generally used only for development and is not recommended for production processing.
10. If the target table does not yet exist, click the **Create** tab and create the table using the **Actions**. See the "JSON: Schemas" section of the MDM API documentation if you require more granular control over table structure.
11. To use an input-to-MDM mapping other than the default **Implicit** name matching, click the **Mapping** tab and configure a detailed field mapping.
12. To set the tool name or message levels, configure the Execution tab properties.

5.7. MDM UPDATE TOOL

This tool updates fields in existing MDM records. If you know you only need to update a single field (for example, recomputing the "closest store ID" in batch), the MDM Update tool is a safer and faster alternative to modifying the entire record.

5.7.1. MDM UPDATE TOOL PARAMETERS

The MDM Update tool has two three of parameters in addition to the standard execution options: *Configuration, Options,* and *Mapping*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database to which records will be written.
Table	The name of the table to which records will be written.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.

Options

Block size	Size of a record block. Default is 100.
Use shared changeset	All MDM tools with this option selected cooperate around a single changeset. Pending changes across all such tools are gathered into a single shared changeset. When the last MDM tool in the project with this option selected finishes, the shared changeset is committed.
Wait for changeset commit	Check this box to wait for the changeset to commit at the end of the tool's processing. If unchecked, the changeset commit will be requested but the tool/project will not wait for it to complete.
Threads	Degree of parallel processing desired. Default is 3.
Enable trigger output	If selected, this option causes the tool to sprout a T "Trigger" connector. A single record will be sent to this connection after all operations are complete. This can be used to sequence other operations by connecting to downstream tools that support a Trigger input connector. See Trigger input and output.
Use unversioned changesets	If selected, this option causes the changes to overwrite the current version of the record rather than create a new one. This is only meant for multi-step system processes.

Mapping

Actions	If Mapping type is Explicit, specify how to map input fields to MDM fields: • **Refresh**: Read the MDM table schema again. • **Map all fields**: Match up field names and create entries. • **Clear mapping**: Clear all mapping.

5.7.2. CONFIGURE THE MDM UPDATE TOOL

Before configuring an MDM Update tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Update tool

1. Select the **MDM Update** tool, and then click the **Configuration** tab on the Properties pane.

2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.8. MDM DELETE TOOL

This tool deletes records from the MDM database that match a series of primary key values. The tool does not currently support finding (and then deleting) records by anything other than primary key.

5.8.1. MDM DELETE TOOL PARAMETERS

The MDM Delete tool has two sets of parameters in addition to the standard execution options: *Configuration* and *Options*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database to which records will be written.
Table	The name of the table to which records will be written.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Select input field	The primary key field that identifies records to be deleted.

Options

Block size	Size of a record block. Default is 250.
Use shared changeset	All MDM tools with this option checked cooperate around a single changeset. Pending changes across all such tools are gathered into a single shared changeset. When the last MDM tool in the project with this option checked finishes, the shared changeset is committed.
Wait for changeset commit	Check this box to wait for the changeset to commit at the end of the tool's processing. If unchecked, the changeset commit will be requested but the tool/project will not wait for it to complete.
Threads	Degree of parallel processing desired. Default is 3.
Enable trigger output	If selected, this option causes the tool to sprout a T "Trigger" connector. A single record will be sent to this connection after all operations are complete. This can be used to sequence other operations by connecting to downstream tools that support a Trigger input connector. See Trigger input and output.
Use unversioned changesets	If selected, this option causes the changes to overwrite the current version of the record rather than create a new one. This is only meant for multi-step system processes.

5.8.2. CONFIGURE THE MDM DELETE TOOL

Before configuring an MDM Delete tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Delete tool

1. Select the **MDM Delete** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.9. MDM KEY QUERY TOOL

This tool reads records from the MDM database that match a series of primary key values. The tool joins the input records to the MDM table on the primary key fields. It does not currently support query by anything other than primary key.

5.9.1. MDM KEY QUERY TOOL PARAMETERS

The MDM Key Query tool has three sets of parameters in addition to the standard execution options: *Configuration, Select,* and *Options*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database to query.
Table	The name of the table to query.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Select data from	Specifies which version of the data to query: • **Current**: the most recently-committed data. • **Timestamp**: the latest changeset committed before a specified timestamp. • **Pending changeset**: a specified pending changeset.
Timestamp	If **Select view date by** is **Timestamp**, specifies the timestamp.
Changeset	If **Select data from** is **Pending changeset**, specifies the changeset.
Select input field	Input field containing the primary key values you want to query.

Select

Include match document	Select to return a JSON document containing match information for each record.
Include match group fields	Select to return the value of the match group id(s).
Select fields	Use this option to choose which fields are returned by the query tool.

Options

Limit records	If selected, limits the number of records read.
Record limit	If Limit records is selected, specifies the number of records to be read. Default is 1,000.
Block size	Size of a record block. Default is 250. This is a tuning parameter. Generally for narrow records you can make this larger and for wide records you should make this smaller.
Threads	Degree of parallel processing desired. Default is 3.

5.9.2. CONFIGURE THE MDM KEY QUERY TOOL

Before configuring an MDM Key Query tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Key Query tool

1. Select the **MDM Key Query** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.10. MDM DYNAMIC QUERY TOOL

This tool reads records from the MDM database with fields that match a series of values. For example, this tool is useful for querying "match candidates" by segmentation key. The fields should be indexed in the schema or performance will be poor.

5.10.1. MDM DYNAMIC QUERY TOOL PARAMETERS

The MDM Dynamic Query tool has two sets of parameters in addition to the standard execution options: *Configuration, Select,* and *Options*.

Configuration

Override	See MDM tool shared settings.

Database	The name of the MDM database to query.
Table	The name of the table to query.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Select data from	Specifies which version of the data to query: • **Current**: the most recently-committed data. • **Timestamp**: the latest changeset committed before a specified timestamp. • **Pending changeset**: a specified pending changeset.
Timestamp	If **Select view date by** is **Timestamp**, specifies the timestamp.
Changeset	If **Select data from** is **Pending changeset**, specifies the changeset.
MDM field	MDM field you want to query.
Input field	Input field containing the primary key values you want to query in the MDM field.

Select

Include match document	Select to return a JSON document containing match information for each record.
Include match group fields	Maximum number of records per transaction/request. Default is 100. This is a tuning parameter. Generally, you can make this larger for narrow records and smaller for wide records.
Select fields	Use this option to choose which fields are returned by the query tool.

Options

Limit records	If selected, specifies the maximum number of records to be read. Default is 1,000.
Record limit	If **Limit records** is selected, specifies the number of records to be read.
Block size	Size of a record block. Default is 250. In general, the larger your records are, the smaller this value should be.
Threads	Degree of parallel processing desired. Values up to three may help speed processing. Default is 3.

5.10.2. CONFIGURE THE MDM DYNAMIC QUERY TOOL

Before configuring an MDM Dynamic Query tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Dynamic Query tool

1. Select the **MDM Dynamic Query** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.11. MDM QUERY VIEW TOOL

This tool queries results of a computed view, reading all records in batch. It can either read the "current" data, or it can query a historical view of the data.

5.11.1. MDM QUERY VIEW TOOL PARAMETERS

The MDM Query View tool has three sets of parameters in addition to the standard execution options: *Configuration, Select,* and *Options*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database from which records will be read.
View	The name of the table from which records will be read.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Select data from	Specifies which version of the data to query: • **Current**: the most recently-committed data. • **Timestamp**: the latest changeset committed before a specified timestamp. • **Pending changeset**: a specified pending changeset.
Timestamp	If **Select data from** is **Timestamp**, specifies the timestamp.
Changeset	If **Select data from** is **Pending changeset**, specifies the changeset.

Select

Select fields	Use this option to choose which fields are returned by the query tool.

Options

Limit records	If selected, limits the number of records read.
Record limit	If **Limit records** is selected, specifies the number of records to be read.
Block size	Maximum number of records per transaction/request. Default is `250`. In general, the larger your records are, the smaller this value should be.
Enable trigger input	If selected, causes the tool to sprout a T "Trigger" input connector. This can be used to sequence operations by connected to upstream tools that support a Trigger output connector. See Trigger input and output.

5.11.2. CONFIGURE THE MDM QUERY VIEW TOOL

Before configuring an MDM Query View tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Query View tool

1. Select the **MDM Query View** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.12. MDM SET MATCH GROUPS TOOL

This tool sets match group IDs on existing records. It accepts a sequence of primary key values and group IDs.

It is assumed that the Redpoint Data Management projects are well-behaved:

- The tool works with any "upstream" manual matching performed in the MDM web application. It honors any suppression information already attached to the records, and will not match records that have overrides or are in match-review.
- Use the MDM Generate Match Group IDs tool to obtain the "next match group" IDs. Data Management projects should not make up their own IDs using their own rules. Even if the matching macros produce match group IDs, they must be re-mapped onto MDM match group IDs.

5.12.1. MDM SET MATCH GROUPS TOOL PARAMETERS

The MDM Set Match Groups tool has a single set of parameters in addition to the standard execution options: *Configuration*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database to query.
Table	The name of the table to query.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Match type	If blank, use the default match group. The match type for this workflow (for example, **individual** or **household**).
Primary key input field	The primary key field for the input records, which must be an `integer` or `long`.
Group ID input field	The group ID field for the input records.
Score field	The score field for the input records.
Block size	Size of a record block. Default is 1,000. In general, the larger your records are, the smaller this value should be.
Use shared changeset	All MDM tools with this option checked cooperate around a single changeset. Pending changes across all such tools are gathered into a single shared changeset. When the last MDM tool in the project with this option checked finishes, the shared changeset is committed.
Wait for changeset commit	Select this option to wait for the changeset to commit at the end of the tool's processing. If not selected, the changeset commit will be requested but the tool/project will not wait for it to complete.
Commit wait seconds	If **Wait for changeset commit** is selected, the time to wait for a changeset commit to complete.
Threads	Degree of parallel processing desired. Values up to three may help speed processing. Default is 3.
Enable trigger output	If selected, this option causes the tool to sprout a T "Trigger" connector. A single record will be sent to this connection after all operations are complete. This can be used to sequence other operations by connecting to downstream tools that

	support a Trigger input connector. See Trigger input and output.
Use unversioned changesets	If selected, changed records are modified "in place" rather than created as a new version of the record.

5.12.2. CONFIGURE THE MDM SET MATCH GROUPS TOOL

Before configuring an MDM Set Match Groups tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Set Match Groups tool

1. Select the **MDM Set Match Groups** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.13. MDM GENERATE MATCH GROUP IDS TOOL

This tool gets the next available match group IDs for a table and match level. Use this tool ahead of the MDM Set Match Groups tool to ensure that you are using non-conflicting and unique match group IDs that are guaranteed not to collide with any other match process.

5.13.1. MDM GENERATE MATCH GROUP IDS TOOL PARAMETERS

The MDM Generate Match Group IDs tool has a single set of parameters in addition to the standard execution options: *Configuration*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database from which records will be read.
Table	The name of the table from which records will be read..
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Match type	The match type for this workflow (for example, **individual** or **household**).
ID count field	Match group ID field for the given table.

Set minimum ID value	Generate match group IDs, starting at a given minimum value.
Minimum ID value	The minimum ID value that will be generated.

5.13.2. CONFIGURE THE MDM GENERATE MATCH GROUP IDS TOOL

Before configuring an MDM Generate Match Group IDs tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Generate Match Group IDs tool

1. Select the **MDM Generate Match Group IDs** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.14. MDM MATCH GROUP QUERY TOOL

This tool reads records from the MDM database that are members of the list of match groups read from the input. It is useful for querying the existing members of match groups in anticipation of re-matching triggered by identifying information changes.

5.14.1. MDM MATCH GROUP QUERY TOOL PARAMETERS

The MDM Match Group Query tool has three sets of parameters in addition to the standard execution options: *Configuration, Select,* and *Options*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database from which records will be read.
Table	The name of the table from which records will be read.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Select data from	Specifies which version of the data to query: • **Current**: the most recently-committed data. • **Timestamp**: the latest changeset committed before a specified timestamp. • **Pending changeset**: a specified pending changeset.

Timestamp	If **Select data from** is **Timestamp**, specifies the timestamp.
Changeset	If **Select data from** is **Pending changeset**, specifies the changeset.
Select input match group ID	The field that contains the match group ID values.
Match type	The match type for this workflow (for example, **individual** or **household**).

Select

Include match document	Select to return a JSON document containing match information for each record.
Include match group fields	Select to return the value of the match group id(s).
Select fields	Use this option to choose which fields are returned by the query tool.

Options

Limit records	If selected, limits the number of records read.
Record limit	If Limit records is selected, specifies the number of records to be read.
Block size	Size of a record block. Default is 250. This is a tuning parameter. Generally for narrow records you can make this larger and for wide records you should make this smaller.
Enable trigger input	If selected, this option causes the tool to sprout a T "Trigger" input connector. This can be used to sequence operations by connected to upstream tools that support a Trigger output connector. See Trigger input and output.

5.14.2. CONFIGURE THE MDM MATCH GROUP QUERY TOOL

Before configuring an MDM Match Group Query tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Match Group Query tool

1. Select the **MDM Match Group Query** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.15. MDM MATCH SUPPRESSION QUERY TOOL

This tool queries the suppression information to support Data Management-based matching:

- **Never-match overrides**: This is a series of primary-key pairs. These records must never be matched, and should not end up in the same group.
- **Always-match overrides**: This is a series of primary-key pairs. These records must always be matched and end up in the same group. In theory there should be no conflict between these and the never-match overrides, but if there is the never-match overrides take precedence.
- **Suppress**: This is a series of primary-key values. These records should not be processed at all, because they are part of an ongoing match review workflow.

5.15.1. MDM MATCH SUPPRESSION QUERY TOOL PARAMETERS

The MDM Match Supression Query tool has two sets of parameters in addition to the standard execution options: *Configuration* and *Options*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database from which records will be read.
Table	The name of the table from which records will be read.
Match type	If a table has match types other than default, chose which match type to use, such as **household** or **individual**.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Description equals	Filter changesets by text contained in the changeset description. In this case, the string value must be an exact match.
Description contains	Filter changesets by text contained in the changeset description. In this case, the string value must be contained within a description.

Options

Enable trigger input	If selected, this option causes the tool to sprout a T "Trigger" connector. A single record will be sent to this connection after all operations are complete. This can be used to sequence other operations by connecting to downstream tools that support a Trigger input connector. See Trigger input and output.

5.15.2. CONFIGURE THE MATCH SUPPRESSION QUERY TOOL

Before configuring an MDM Match Suppression Query tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Match Suppression Query tool

1. Select the **MDM Match Suppression Query** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.16. MDM START MATCH REVIEW WORKFLOW TOOL

This is an all-in-one specialty tool with the purpose of supporting match-override-review workflows. When you run this tool, it is assumed that the last committed changeset contains the "accepted" match groups. These match groups are typically created by the MDM Set Match Groups tool tool. In fact, setting the accepted match groups and initiating the review workflows can be done in the same project if you set the tool priorities correctly.

This tool requires four fields to be mapped on the input:

- Review set ID: This identifies the "review set", which will become a single workitem for review. Often, this consists of a single "loose" match group, which can be assumed to represent the consolidation of two or more "tight" match groups or singletons. However, there are corner cases where this logic isn't sufficient. For correct behavior, this ID should be the result of AO Associate Match IDs run to combine the loose and tight group IDs.
- Primary key field: The primary key of each record, which must be an integer or long.
- Proposed group ID: The "loose" group ID that is being proposed, relative to the accepted "tight" group IDs
- Score: The score given to the entire review set, if it is accepted. Only the first score of each review set is processed, although they should all be the same value within a review set.

5.16.1. MDM START MATCH REVIEW WORKFLOW TOOL PARAMETERS

The MDM Start Match Review Workflow tool has a single set of configuration parameters in addition to the standard execution options: *Configuration*

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database from which records will be read.
Table	The name of the table from which records will be read.

Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Workflow	The review workflow to start.
Match type	The match type for this workflow (for example, **individual** or **household**).
Review set ID field	This identifies the "review set", which will become a single workitem for review. Often, this consists of a single "loose" match group, that can be assumed to represent the consolidation of two or more "tight" match groups or singletons. However, there are corner cases where this logic isn't sufficient. For correct behavior, this ID should be the result of AO Associate Match IDs run to combine the loose and tight group IDs.
Primary key field	The record primary key field, which must be a 4-byte or 8-byte signed integer value.
Group ID field (proposed)	The loose group ID that is being proposed, relative to the accepted tight group IDs.
Score field	The score given to the entire review set, if it is accepted. Only the first score of each review set is processed, although they should all be the same value within a review set.
Wait for changeset commit	Select this option to wait for the changeset to commit at the end of the tool's processing. If not selected, the changeset commit will be requested but the tool/project will not wait for it to complete.
Commit wait seconds	Time to wait for changeset commit to complete. If the commit does not complete within this time, the project fails and aborts; however, the commit will continue to process.
Threads	Degree of parallel processing desired. Default is 3.
Enable trigger output	If selected, this option causes the tool to sprout a T "Trigger" connector. A single record will be sent to this connection after all operations are complete. This can be used to sequence other operations by connecting to downstream tools that support a Trigger input connector. See Trigger input and output.

5.16.2. CONFIGURE THE MDM START MATCH REVIEW WORKFLOW TOOL

Before configuring an MDM Start Match Review Workflow tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Start Match Review Workflow tool

1. Select the **MDM Start Match Review Workflow** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.17. MDM SHARED CHANGESET COMMITTER

This tool causes all of the other MDM connector tools to use a single shared changeset for everything they do rather than causing a new version of each modified record to be created. It accepts a trigger input.

Shared changesets can only be used within the same database.

5.17.1. MDM SHARED CHANGESET COMMITTER TOOL PARAMETERS

The MDM Shared Changeset Committer tool has two sets of parameters in addition to the standard execution options: *Configuration* and *Options*.

Configuration

Override	See MDM tool shared settings.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.

Options

Wait for changeset commit	Select this option to wait for the changeset to commit at the end of the tool's processing. If not selected, the changeset commit will be requested but the tool/project will not wait for it to complete.
Commit wait seconds	If **Wait for changeset commit** is selected, the time to wait for a changeset commit to complete. If the commit does not complete within this time, the project fails and aborts; however, the commit will continue to process.
Use unversioned changesets	If selected, changed records are modified "in place" rather than created as a new version of the record.

5.17.2. CONFIGURE THE MDM SHARED CHANGESET COMMITTER

Before configuring an MDM Shared Changeset Committer tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Shared Changeset Committer tool

1. Select the **MDM Shared Changeset Committer** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.
3. Click **Connect/Refresh** to validate the connection and populate the database list.
4. In the **Select input field**, select the primary key values you want to delete.
5. Optionally, click the **Options** tab and configure processing options:
   - Configure **Wait for changeset commit** and **Commit wait seconds** options to specify how the changeset is handled.
   - To modify records "in place" rather than creating a new version of the record for each change, select **Use unversioned changesets**.
6. Set any desired execution options.

5.18. MDM DIFF REPORT TOOL

The MDM Report tool generates a report of changes contained in the changeset shared by other tools that have the **Use shared changeset** option enabled.

5.18.1. MDM DIFF REPORT TOOL PARAMETERS

The MDM Diff Report tool has a single set of parameters in addition to the standard execution options: *Configuration*

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database from which records will be read.
Table	The name of the table from which records will be read.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Return full record for delete operations	If a record is to be deleted, return the full record.
Return old record for modify operations	If a record is to be modified, return the old record.
Return new record for modify operations	If a record is to be modified, return the new record.

Return field diffs for modify operations	If a record is to be modified, return a list of field diffs.
Return match group IDs	Return the match group IDs for all the match groups involved.
Return match documents	Select to return JSON documents containing match group IDs, match scores, and user overrides for all match types.

5.18.2. CONFIGURE THE MDM DIFF REPORT TOOL

Before configuring an MDM Diff Report tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Diff Report tool

1. Select the **MDM Diff Report** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.19. MDM PII QUERY TOOL

This tool reports mask and purge activity for a given database and table in MDM.

5.19.1. MDM PII QUERY TOOL PARAMETERS

The MDM PII Query tool has three sets of parameters in addition to the standard execution options: *Configuration*, *Filter*, and *Options*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database to query.
Table	The name of the table to query.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Order by	If specified, the generated records are sorted in ascending order by one of the following: Timestamp, (Primary) Key, Action (mask or purge), Username.

Filter

Filter by timestamp	If selected, only return records that were created between the **Before** and **After** timestamps.
Username	Only return records associated with the given username.
Keys	Only return mask/purge activity for records which have the given Primary Keys.
Action	Return all activity or just mask or purge activity.

Options

Limit records	If selected, limits the number of records returned.
Record limit	If **Limit records** is selected, specifies the number of records to be read. The default is 1,000 records.
Block size	Size of a record block. Default is 250. In general, the larger your records are, the smaller this value should be.
Enable trigger input	If selected, this option causes the tool to sprout a T "Trigger" connector. A single record will be sent to this connection after all operations are complete by another tool. This can be used to sequence other operations by connecting to downstream tools that support a Trigger input connector. See Trigger input and output.

5.19.2. CONFIGURE THE MDM PII QUERY TOOL

Before configuring an MDM PII Query tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM PII Query tool

1. Select the **MDM PII Query** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

5.20. MDM MASK/PURGE TOOL

Use this tool to remove PII (Personally Identifiable Information) from MDM records. Masking a record will remove fields marked as PII from a table in all versions of the record. Purging a record will delete all versions of a record. In both cases MMDMDM will create mask hash records and a mask-

purge activity records. The mask hash records are so a user can verify that a record was masked or purged using the field values associated with the mask-hash. An activity record (see the MDM PII Query tool) is also created to keep track of who masked or purged a record as well as when that happened.

5.20.1. MDM MASK/PURGE TOOL PARAMETERS

The MDM Mask/Purge tool has two sets of parameters in addition to the standard execution options: *Configuration* and *Options*.

Configuration

Override	See MDM tool shared settings.
Database	The name of the MDM database in which records will be masked or purged.
Table	The name of the table from which records will be read.
Detailed logging	If selected, sends detailed activity logs to the Message Viewer.
Select input field	Input field containing the primary key values you want to mask or purge.
Action	Choose Mask or Purge.

Options

Block size	Size of a record block. Default is 100.
Use shared changeset	All MDM tools with this option checked cooperate around a single changeset. Pending changes across all such tools are gathered into a single shared changeset. When the last MDM tool in the project with this option checked finishes, the shared changeset is committed.
Wait for changeset commit	Check this box to wait for the changeset to commit at the end of the tool's processing. If unchecked, the changeset commit will be requested but the tool/project will not wait for it to complete.
Threads	Degree of parallel processing desired. Values up to 3 may help speed processing. Default is 1.
Enable trigger output	If selected, this option causes the tool to sprout a T "Trigger" connector. A single record will be sent to this connection after all operations are complete. This can be used to sequence other operations by connecting to downstream tools that

	support a Trigger input connector. See Trigger input and output.
Use unversioned changesets	If selected, this option causes the changes to overwrite the current version of the record rather than create a new one. This is only meant for multi-step system processes.

5.20.2. CONFIGURE THE MDM MASK/PURGE TOOL

Before configuring an MDM Mask/Purge tool, you should have an MDM database connection defined in tool connection settings.

To configure the MDM Mask/Purge tool

1. Select the **MDM Mask/Purge** tool, and then click the **Configuration** tab on the Properties pane.
2. Optionally, override shared settings: open the **Connection settings** section, click **Override**, and then specify new values for the tool.

6. REFERENCE

This section contains technical details on MDM's data models, matching, views and subscriptions, workflow creation, and support for GDPR/privacy mandates.

6.1. DATABASES AND DATA MODELS

Currently, MDM only supports MongoDB NoSQL databases.

The table options, field options, and field data types you configure in MDM collectively define your data model.

Note

In a well-formed database ,the **system** user, or any user who is a member of any group that has the DBA role, can modify schemas.

6.1.1. TABLE OPTIONS

When you create a table in MDM, the following options are available:

Table Option	Details
Name	Table name. *Required.*
Type	**versioned** tables have a primary key and retain historical versions of data. See Versioned and time-series tables for more information.

Table Option	Details
	timeseries tables store transactional or event data, and share the primary key of an associated versioned table.
Primary Key	*Required* if **Type** is **versioned**. Field that holds the primary key value, which is used to uniquely identify records. Defaults to **id**.
Sequence Field	*Optional.* If **Type** is **versioned**, you can select this to prevent overwriting of new data with old data. If enabled, replace and upsert operations will not create new versions of records where the sequence field would be smaller than the previous sequence field. The modify operation will also honor sequence semantics if the sequence field is part of the modification. The primary motivation for using a sequence value is when a service has multiple asynchronous sources where only the latest update is applied, and if updates come in out of order, the "previous" updates are not applied since they would just be overwritten.
Enable DM Matching Support	*Optional.* If **Type** is **Versioned**, you can select this to enable matching operations in Redpoint Data Management. For information concerning Redpoint Data Management matching, refer to the *Redpoint Data Management User Guide*.
Display Match Group IDs	If **Enable DM Matching Support** is selected, you can select this to display match group ID fields in the Data and Match Workflow pages. The default or latest match group ID is displayed as `Default Group ID`, or simply `Group ID` if there are no other groups. All other group IDs appear as `${CapitalizedName} Group ID`, for example `Individual Group ID` and `Household Group ID`.
Description	*Optional.* Description of the table.

Note

Names in MDM are case-preserving but case-insensitive.

The simplified regular expression (*regex*) for a name is:

```
[_a-zA-Z][_a-zA-Z0-9]*
```

The formal definition is:

```
^[\\p{L}_][\\p{L}\\p{Nd}_]*$
```

which supports the full unicode letter set, allowing tables and field names such as:

Œèñ

6.1.2. FIELD OPTIONS

When you add a field to a table in MDM, the following options are available:

Field Option	Details
Field	Field name. Each field in a table must be unique. A name must start with an alphabetic character or underscore and all remaining characters must be alphanumeric or underscore. MDM treats **Field** in a case-insensitive manner.
Nullable	If selected, **Field** may contain null or empty values.
PII	If selected, **Field** contains Personally Identifiable Information (PII). For more information on PII and data visibility in MDM, refer to GDPR and privacy support.
Index	If selected, the field will be indexed (a table index will be added for **Field** in the database), which reduces query search times. Be aware that indexing many fields may slow insert operations on the table. You should enable indexing for fields you expect to query often.
Display	If selected, **Field** will be displayed in data views. Note that fields that are not display-enabled cannot be queried.
Type	See the Field data types table for a list of permissible field types in MDM.
Details	If **Type** is **text**, **integer**, or **float**, the size of the field (in characters or bytes).
Signed	If **Type** is **integer**, whether the field is signed.

Note

Names in MDM are case-preserving but case-insensitive.

The simplified regular expression (*regex*) for a name is:

```
[_a-zA-Z][_a-zA-Z0-9]*
```

The formal definition is:

```
^[\\p{L}_][\\p{L}\\p{Nd}_]*$
```

which supports the full unicode letter set, allowing tables and field names such as:

Œèñ

6.1.3. FIELD DATA TYPES

When you add a field to a table in MDM, the following field data types are available:

Field Type	Details
text	**String** consisting of Unicode characters. Size may be 1 to 1,000,000. **Null** and "" are treated as the same value.
money	Fixed-precision number with four decimal places, with an effective range of -922,337,203,685,477.5808 to +922,337,203,685,477.5807.
integer	Size is either 1, 2, 4, or 8 bytes. Signed is **true** or **false**. The combination { **"size":8, "signed":false** } is not allowed due to restrictions in the underlying database.
float	Floating-point number. • If **Size** is 4 bytes, the range of values that can be stored is -3.402823466e+38 to 3.402823466e+38. The smallest allowable value is 1.175494351e–38F. • If **Size** is 8 bytes, the range of values that can be stored is -1.7976931348623158e+308 to 1.7976931348623158e+308. The smallest allowable value is 2.2250738585072014e–308.
date **time** **datetime**	**Date**, **Time**, and **DateTime** are all represented as text using 24-hour ISO formatting. **DateTime** is a local date/time without an explicit timezone, and is interpreted according to the usage context. It should be of the form: • Date: **yyyy-MM-dd** • Time: **HH:mm:sss** • DateTime: **yyyy-MM-ddTHH:mm:ss.SSS**
boolean	Text characters or strings used as **True** and **False** values (0/1, true/false, T/F).
table	See Table options.
linkto	See Linkage fields for more information.

6.1.4. VERSIONED AND TIME-SERIES TABLES

MDM supports the following kinds of tables:

- Versioned tables
- Time-series tables

6.1.4.1. VERSIONED TABLES

Versioned tables are central to MDM and are used for the creation and maintenance of a *golden record*, typically comprised of entity data.

- All data or metadata that can be manipulated directly by MDM users are stored in versioned tables.
- Versioned tables have a primary key.
- Every metadata or data create/modify/delete operation performed on a versioned table is retained. See Historical data model for more information.
- Every version of every record and table is associated with a changeset. See Changesets for more information.
- You can view snapshots of the data at any point in time. See Data view for more information.

6.1.4.2. TIME SERIES TABLES

Time series tables can be thought of as child tables of versioned tables. They support the capture of many small timestamped records representing transactional or event data. Examples of such data are purchases, website visits, and instrumentation data. Time series tables are accessed more quickly than versioned tables and are useful when you need to insert large volumes of data. Time series tables lend themselves to views, which are lists of aggregation operations grouped by primary key and filtered by timestamp bounds (see Views and subscriptions for more information).

- A time series table is associated with a versioned table. For example, **customer_transactions** would be associated with **customers**.
- A time series table shares the primary key of its associated versioned table, which is defined using a **link to** field type.
- A time series table has a timestamp field, which you can query to generate a historic view of the data.
- A time series table may not contain nested tables or structured fields.
- You cannot update time series data—only insert and append operations are supported.

6.2. NESTED TABLES AND LINKAGE FIELDS

In MDM you can link a table field to either a sub-table or a field in another table. Sub-tables are called nested tables, and linked fields are called linkage fields.

Because of their simpler nature, we recommend that you use nested tables instead of linkage fields whenever possible. However, if you have static reference data that can be shared among multiple tables or multiple records, use linkage fields.

6.2.1. NESTED TABLES

In MDM tables you can define a **table** field type that contains an array of sub-records. In effect, a **table** field references a sub-table within a table.

- This is the recommended method for referencing live data.
- Sub-documents don't have a life independent of the containing document.
- Sub-documents can't be shared (linked to) by other documents or other tables.
- Use cases:
  - addresses
  - email addresses
  - phone numbers
  - business codes

For example, you might have a list of addresses that you want to store within a customer table. You could create an **addresses** field of type **table** and add a different address string value to each individual sub-record. By doing this, you're actually storing an array of sub-documents in your customer document.

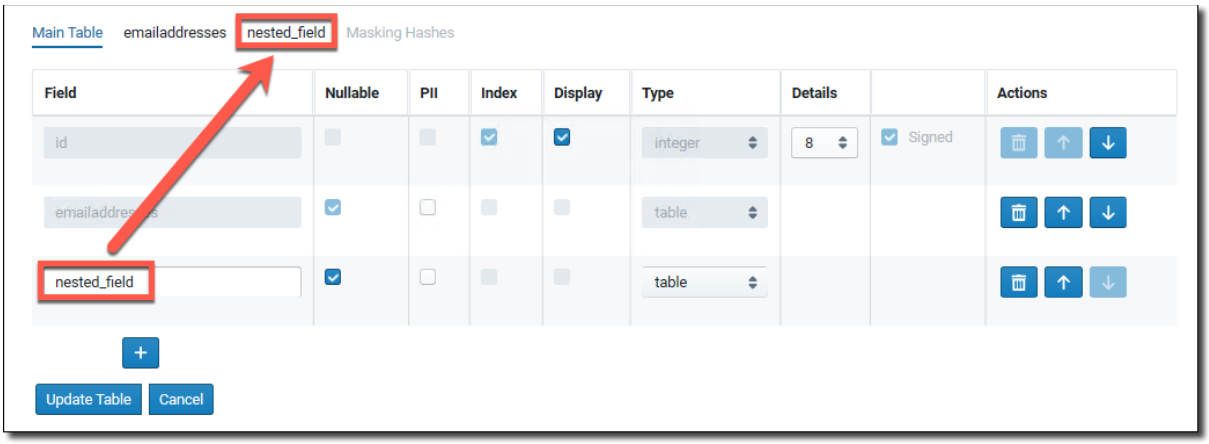
Note

We recommend that you limit the number of records in a nested table to less than 100, and overall record size to less than 10 KB. No record (including its sub-records) can exceed 2 MB in size. Using very large documents may cause problems for web services and browsers. See https://docs.mongodb.com/manual/reference/limits/#bson-documents for the current design limits of these applications and services.

6.2.1.1. ADD A NESTED FIELD TO A TABLE

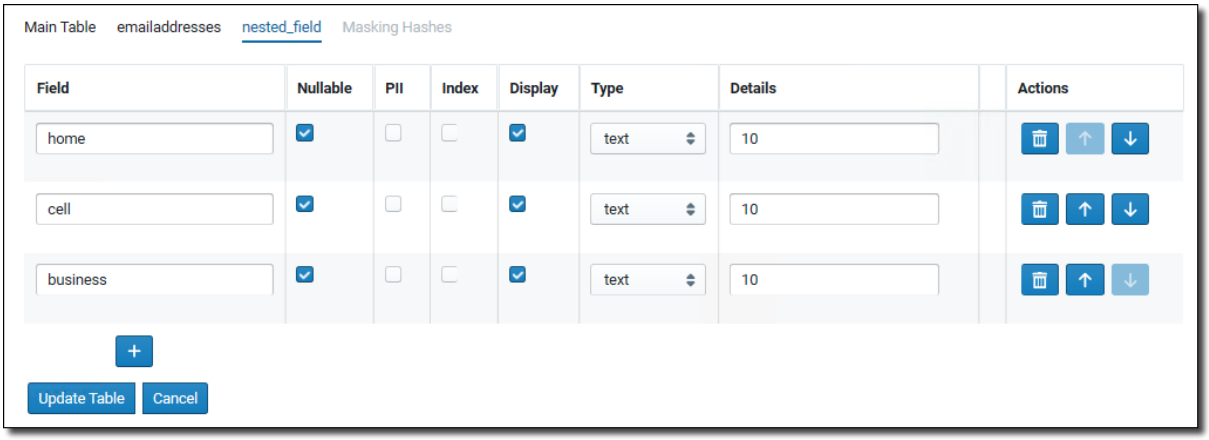
To add a nested field to a table

1. In the MDM Web App, select **Data Hub** > **Tables.**
2. Locate the desired table and click its **Schema** button  .
3. Click  to add a field.
   a. Enter a field name.
   b. Select the field attributes (be sure to select the type **table**).
   The new field name appears as a tab at the top of the table grid.



A table nested field

4. Click the new field tab.
5. Click + to add a sub-field.
    - Enter a sub-field name.
    - Select the sub-field attributes.



Nested field sub-fields

6. Click **Update Table** to finish.

6.2.2. LINKAGE FIELDS

In MDM tables you can define a **link to** (linkage) field type that contains the primary key of another table. You can also display a different field from the reference table other than the primary key. For example, you could link on country abbreviation ("FSM") but display the country's name ("Federated States of Micronesia").

The target field must be of the same type as the linked table's primary key, and both the linked table and the displayed field must exist.

- Recommended method for referencing:
    - Static reference data.
    - Entities that are related to the primary table (for example, a patient table might link to a table of primary care providers).
- Linked-to fields have a life independent of the linking document.

- Linked-to fields can be shared by other documents or other tables.
- Use cases:
  - transactions
  - history
  - friendship graphs
  - Static reference data such as North American Industry Classification System (NAICS) codes and lists of states or countries

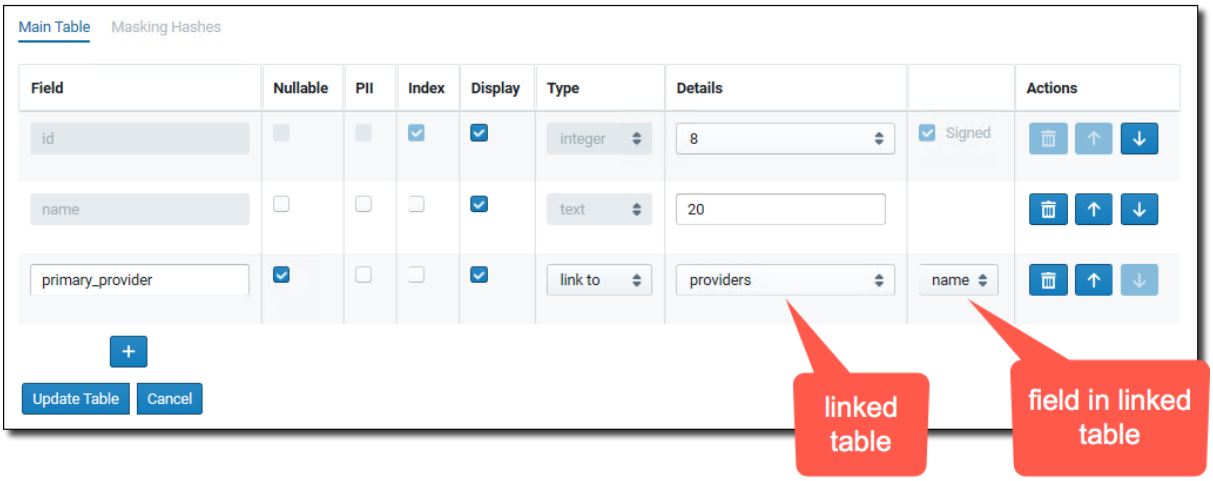
Note

We recommend that you *do not* use linkage fields to decompose entity data into a normal form and spread it out across multiple tables. MDM uses a document database that makes using contained sub-tables and sub-records preferable to using separate daughter tables and foreign key linkages to those sub-records.

6.2.2.1. ADD A LINKAGE FIELD TO A TABLE

To add a linkage field to a table

1. In the MDM Web App, select **Data Hub** > **Tables**.
2. Locate the desired table and click its **Schema** button .
3. Click + to add a field.
   a. Enter a field name.
   b. Select the field attributes (be sure to select the type **link to**).
   c. In the **Details** column, select the MDM table to which you want to link.
   d. In the column to the right of the **Details** column, choose the field that you want to display. The final column is the "displayed" column, not the linked table primary key column. The schema tells MDM so the user does not have to.



A table linkage field

4. Click **Update Table** to finish.

6.3. HISTORICAL DATA MODEL

The following topics cover how MDM handles data throughout the lifespan of a table.

6.3.1. CHANGESETS

In MDM, the data change paradigm is *pending changes are isolated until committed*.

For example, when a user makes a change to a record, the database isn't immediately updated. Instead, MDM creates a *changeset*, and a changeset must be committed for the changes to be implemented.

Think of a changeset as basket for holding related changes, with the changes being anything you can to do to the system. You can also think of a changeset as a unique view of the data (whatever data you started with before you opened the changeset, plus the changes you made).

Creating a new table creates a changeset. Modifying a field in the table schema creates another changeset. Adding records to the table creates yet another changeset.

There are two kinds of changesets:

- Those comprised of metadata changes (such as creating or modifying a table).
- Those comprised of data changes (such as updating fields in a record).

After creation, a changeset may be viewed, deleted, or committed. A data changeset must be explicitly committed by a user, while a metadata changeset is automatically committed by MDM.

Once a changeset is committed, the changes can't be undone or rolled back.

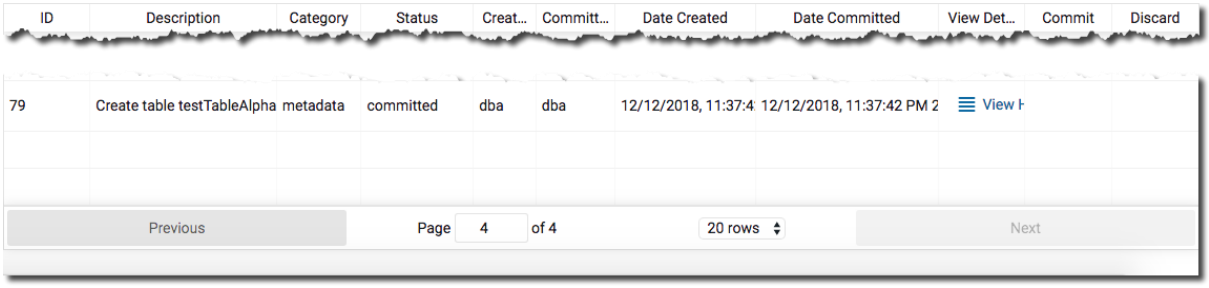
Notes

A separate changeset list is maintained for every database.

Creating or deleting a database does not result in a changeset. Database deletion physically purges all tables and history!

6.3.1.1. EXAMPLE: METADATA CHANGESET (CREATE A TABLE)

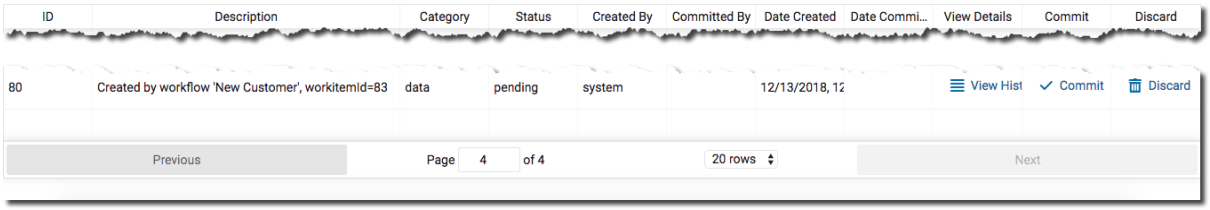
If user "dba" creates a new table **testTableAlpha**, the changeset is automatically committed because it is a metadata changeset.



Metadata changesets are automatically committed

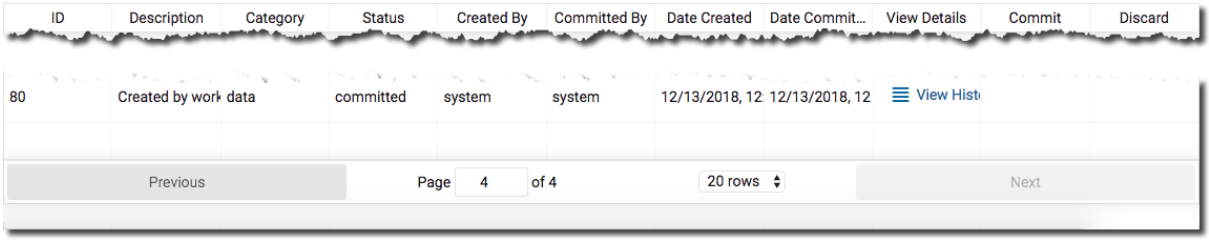
6.3.1.2. EXAMPLE: DATA CHANGESET (ADD A RECORD TO A TABLE)

If user "bizuser" adds a record to a table through an existing workflow, the changeset is not automatically committed because it is a data changeset. Note that the changeset is created by user "system" because all data changesets are created by MDM on behalf of the user.



When a user adds a new record, the data changeset is created by MDM on behalf of the user, but is not automatically committed

Once the record has been reviewed by user "steward" and approved, the changeset is committed. Note that the changeset is committed by "system" because all data changesets are committed by MDM on behalf of the user.



Once a new record has been approved by a data steward, the changeset is committed by user "system" (MDM) on behalf of the user

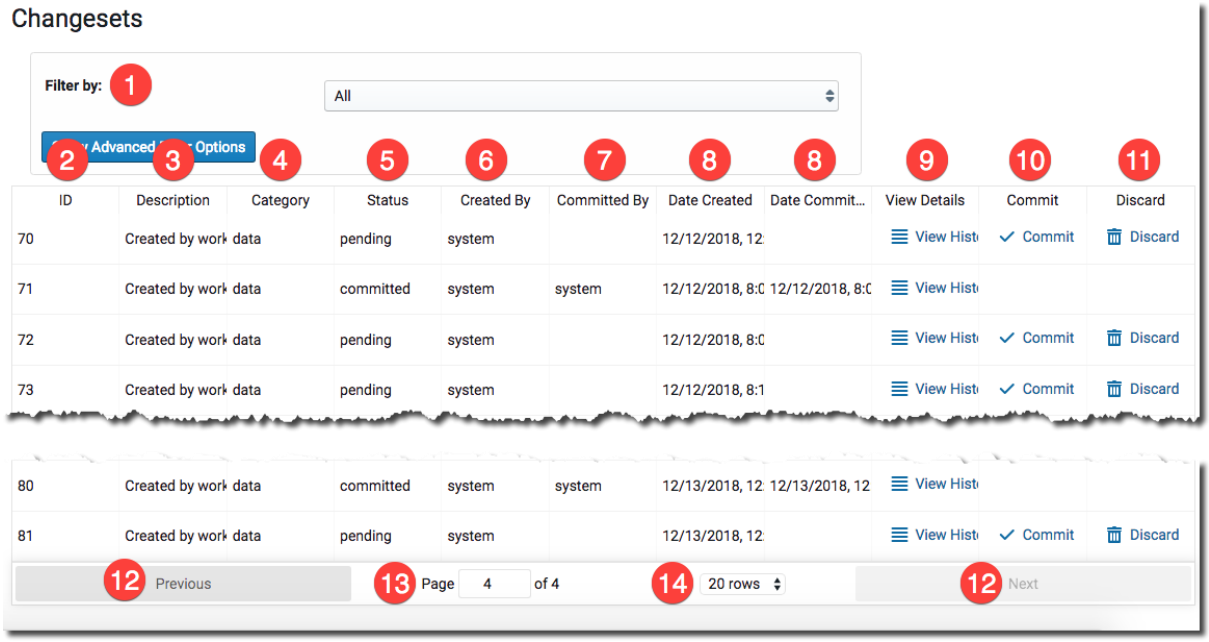
6.3.1.3. CONFLICT RESOLUTION

If multiple users (or systems) try to make a change to the same record(s) simultaneously, the mechanism for conflict resolution is as follows:

- If changes are compatible, the changes are merged automatically. For example, in one record a user changes the name, and another user changes the address.
- If changes are not compatible (for example, in one record a user changes the name to "Jane" and another user changes the name to "Mary Jane"), a *conflict resolution policy* is used. While somewhat configurable, generally this is:
  - For ETL operations, the latest change wins.
  - For interactive operations via workflow or the data hub, this results in a *conflict state*, which must be manually resolved.

6.3.1.4. VIEWING CHANGESETS

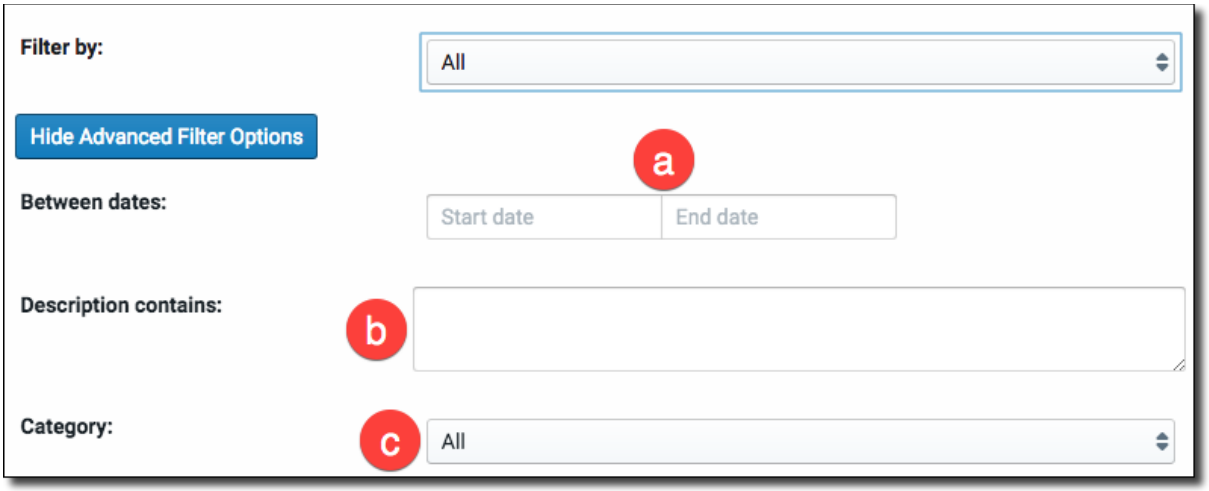
You can view a list of a table's changesets (or a subset) from the **Data Hub** > **Changesets** page.



The Data Hub > Changesets page displays a list of a table's changesets

To view a table's changesets

1. **Filter**—Use the pull-down menu to filter the changeset table according to changeset status (pending, queued, committed, and so on). Click **Show Advanced Filter Options** to expand the **Filter** section and display more options:



Dialog used to filter table changesets

   a. **Between dates**—Click inside the **Start date** and **End date** boxes to display a date picker pop up. The pop up also gives you time period options such as **today**, **past week**, **past year**, and so on.

   b. **Description contains**—Filter changesets by text contained in the changeset description.

   c. **Category**—Filter changesets by category (**data** or **metadata**). Changes involves adding, deleting, or modifying records are **data** changes. Changes that are not record-specific (such as adding a database, adding a table, or modifying a table schema) are **metadata** changes.

2. **ID**—Unique identifier assigned by MDM upon changeset creation.

3. **Description**—Short description of the changeset created by MDM.

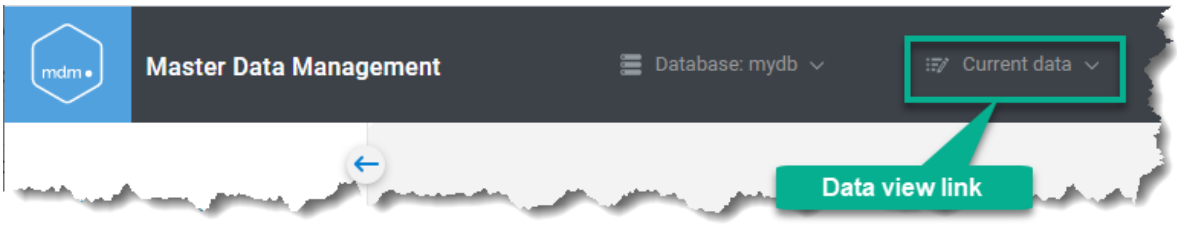
4. **Category**—Values are **data** and **metadata**.
5. **Status**—Values are **pending**, **committed**, **error**, **committing**, and **queued**. **Queued** and **committing** are transitory states as a changeset goes from **pending** to **committed**. Sometimes a changeset will go into the **error** status if the changes cannot be resolved because of conflicting changes made and committed by other users.
6. **Created by**—Which user created the changeset. All data changesets are created by **system** on behalf of the user.
7. **Committed by**—Which user committed the changeset. All data changesets are committed by **system** on behalf of the user.
8. **Date Created** / **Date Committed**—The dates the changeset was created and committed, respectively.
9. **View Details**—Click a changeset's **View History** link to display the changeset's **table**, **operation**, and record **count**.
10. **Commit**—If the changeset has not already been committed, you can **Commit** it here.
11. **Discard**—If the changeset has not already been committed, you can **Discard** it here.
12. **Previous** / **Next**—Click **Previous** or **Next** to display additional changeset pages.
13. **Page**—Which page is currently displayed.
14. **Rows**—Set the number of table rows displayed per page.

6.3.2. CHANGE HISTORY

From the moment a table is created, every change made to that table is recorded by a changeset. It follows that every table record has a change history, and you can View that history.

6.3.3. DATA VIEW

The data view link at the top of the Web App describes the type of data currently displayed.



The data view link displays the type of data currently selected

The data views are:

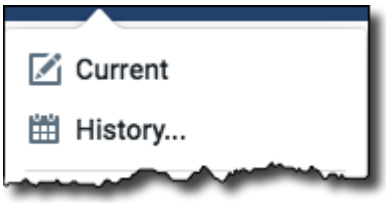
- Current data
- Historical data
- Data inside an open changeset

6.3.3.1. VIEW CURRENT DATA

Current data is the default data view.

To view current data if historical or changeset data is displayed

1. On any Web App page, click the data view link.
2. On the menu, select **Current**. You are now viewing current table data.



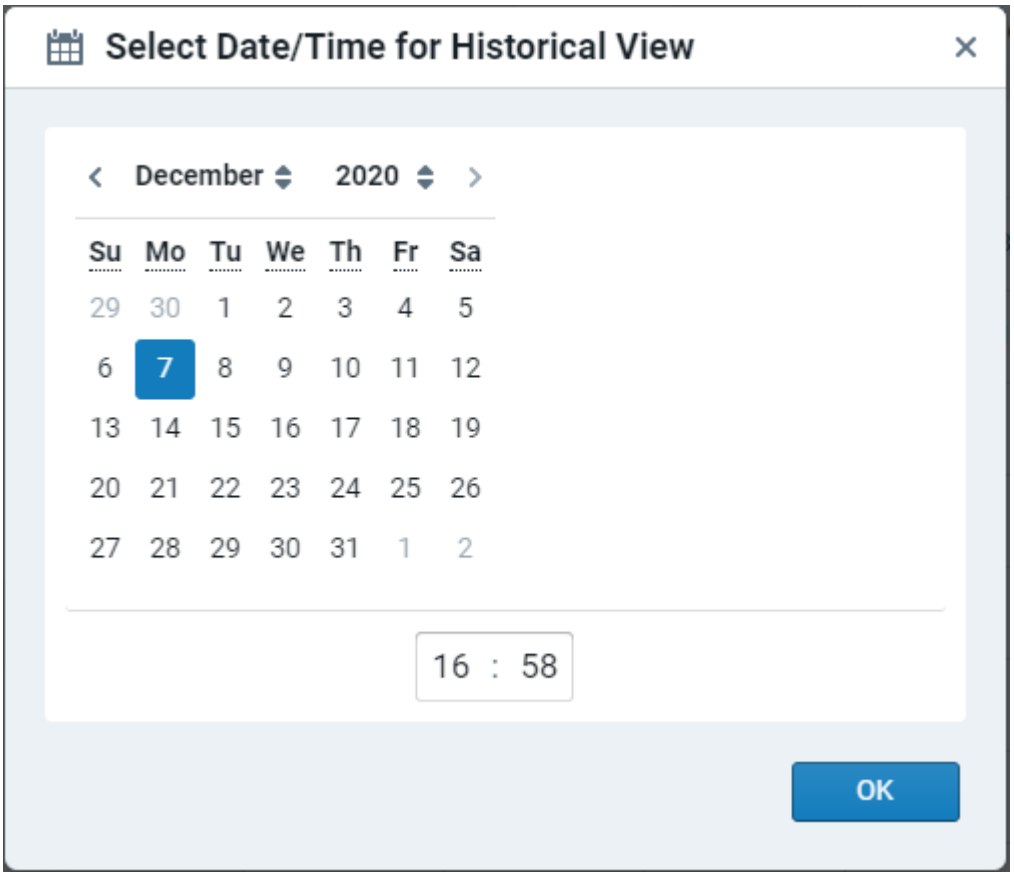
Select Current to view current table data

6.3.3.2. VIEW HISTORICAL DATA

Since every record in a table has a change history, you can pick a historical date and view the state of the table at that point in time.

To view historical data

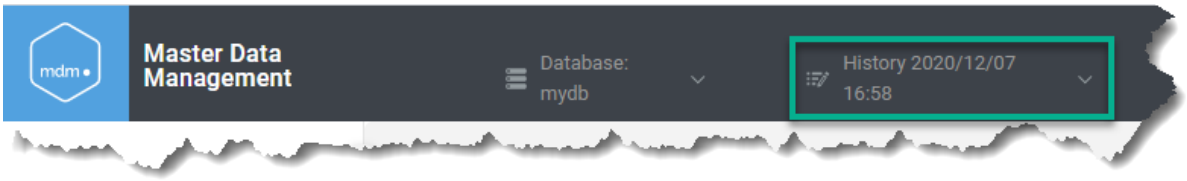
1. On any Web App page, click the data view link.
2. On the menu, select **History**.



Using the date picker to view table data from a specific date

3. Use the date picker to choose a date (you can also select a time of day in 24-hour (**hh**:**mm**) format) and click **OK**.

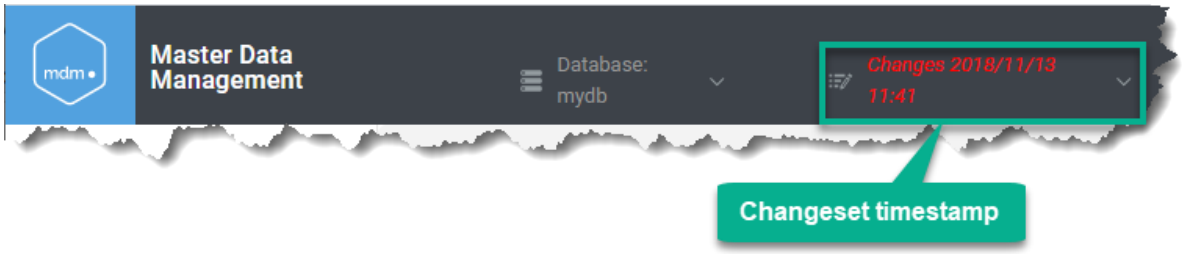
4. You are now viewing table data from the chosen date. The data view link now displays the date from which the data was taken:



When MDM displays historical data, the data view link displays the date from which the data was taken

6.3.3.3. CHANGESET DATA

If a changeset is currently open and uncommitted, the data view link displays that status:



In MDM, metadata changes (database create/delete, table create/edit/delete) are accomplished through an established workflow, and the workflow automatically creates and commits its own changeset.

Lower-level access users make data changes (table record create/edit/delete) through an established workflow, and the workflow automatically creates and commits its own changeset.

However, high-level access users (such as those in the **dbas** or **system** groups) can explicitly open a new changeset, make changes directly to data, and explicitly commit the changeset. Explicitly opening a new changeset and then closing it is done through the data view link.

After opening a new changeset, you can:

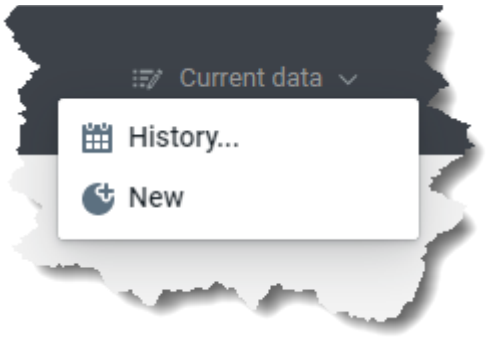
- Make direct changes to table data (outside of a workflow).
- Commit the changeset. All data changes made between opening and committing are included in the changeset.
- Discard the changeset (discard all data changes made after opening the changeset).
- Switch to another data view (current or historical). If you switch to a current or historical data view, the changeset remains open, and you can switch back to it by clicking the data view link and selecting **My uncommitted changes** on the menu.

6.3.3.3.1. OPEN A NEW CHANGESET WITH THE DATA VIEW LINK

Note that you can explicitly open only one changeset at a time.

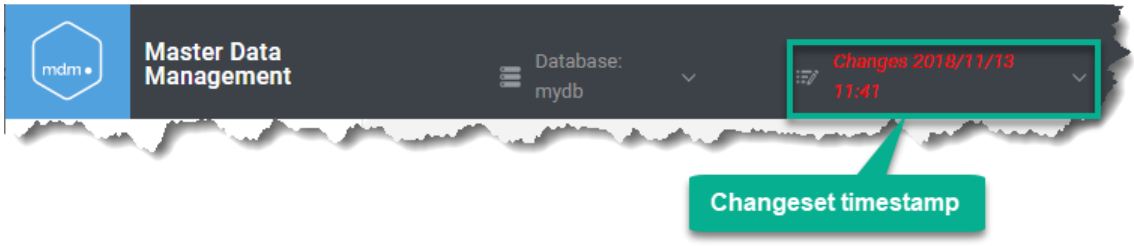
To open a new changeset with the data view link

1. On any Web App page, click the data view link.
2. On the menu, select **New**.

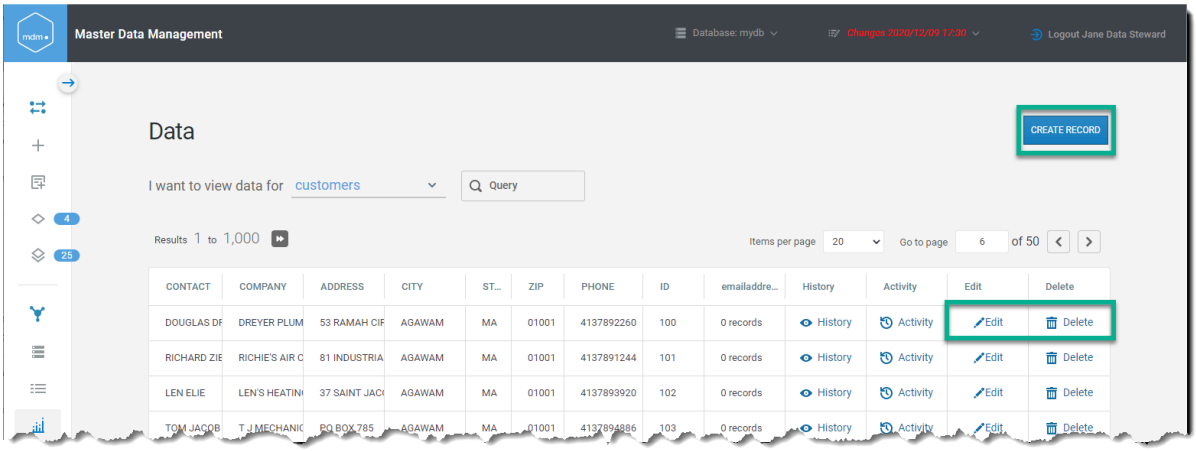


Select New to open a new changeset

3. A new changeset is created. Any changes you make beyond this point are included in the changeset. The data view link text changes to red, with the displayed date and time indicating when the changeset was opened:

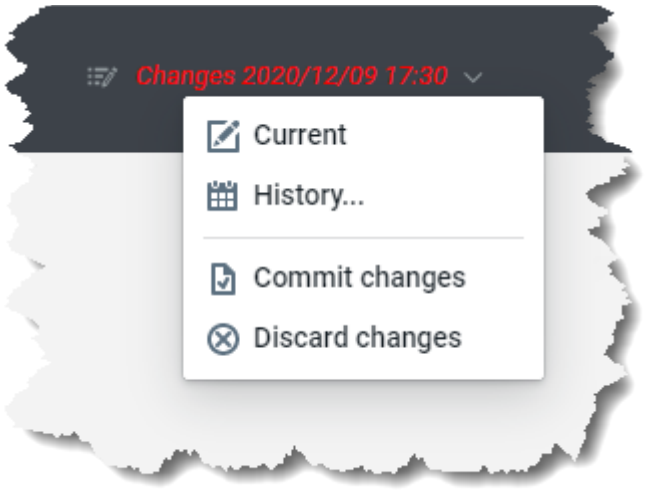


4. Make changes to the desired table. After opening a changeset, a table's data page will display a **Create Record** button and **Edit** and **Delete** buttons on each record row.



After opening a changeset, a table's data page displays a Create Record button, and Edit and Delete buttons on each record row

5. After making the desired table changes, click the data view link and select **Commit changes** or **Discard changes**.



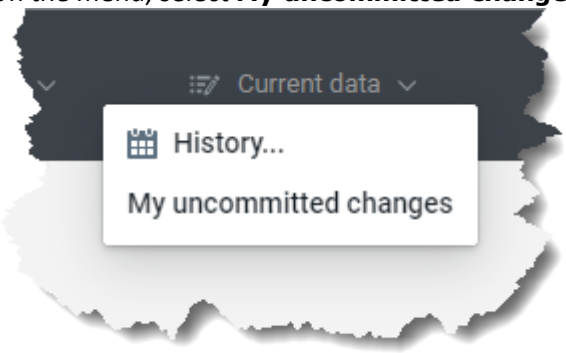
After making table changes, you can commit or discard the changes

Instead of immediately committing or discarding a changeset, you can also leave the changeset open and switch to the **Current** or a **History** view. If you switch to another view, the changeset will remain open until you switch back to it.

6.3.3.3.2. SWITCH BACK TO AN OPEN CHANGESET

To switch back to an open changeset view from a current or a history view:

1. Click the data view link.
2. On the menu, select **My uncommitted changes**. You are now viewing the open changeset data.



Select My uncommitted changes to view the open changeset data

3. At this point you can make further changes (and add to the changeset), commit the changeset, or discard the changeset.

6.3.4. VERSIONED METADATA

In MDM, table schemas are versioned. However, you cannot change a table schema in such a way as to render historical data invalid. Put another way, table representation continuity must be maintained for the entire table history. You can go from less representable to more representable, but not from more to less.

The rules for changing table metadata and maintaining valid historical data are:

- While you can add a new field, the new field must be able to able to accept a **null** value.

If you look at a record's history, values of the added field before the field was added can simply be represented as null.

- You cannot change a field type to another type.

  If you tried to change a field type from text to date, values from before the field change and after the field change could not be represented the same way. For example, the value "monday" cannot be represented as a date.

- Enlarging a field is allowed.

  A text field can be enlarged from 10 to 20 characters, because all field versions can be represented with 20 characters or less.

- Make a field value able to accept a **null** value is allowed.

  Pre-modified fields can be represented as the original field type, while post-changed fields can be represented as the original field type or null.

- Deleted (dropped) fields are allowed.
  - Dropped fields still exist in historical data. After a field is dropped, all values can be represented as null.
  - A new field with the same name as a dropped field is distinct (treated as a different field).
  - Use a Redpoint Data Management ETL process to propagate and convert dropped fields to new fields.

6.4. INTEGRATING REDPOINT DATA MANAGEMENT AND MDM

Redpoint Data Management supports a set of connector tools that enable a variety of operations on MDM, including:

- Querying records with static query parameters
- Querying records that match a series of field values
- Loading new records or replacing existing records
- Bulk-updating records
- Bulk-deleting records
- Querying view results
- Support for various fuzzy-matching operations
- DBA operations such as creating and dropping tables

By using these connector tools, you can leverage the full power of Redpoint Data Management's ETL, hygiene, and matching capabilities. Redpoint Data Management is the preferred method for making batch changes to MDM.

In addition, you can create Redpoint Data Management-hosted web services using these same connector tools, which perform incremental queries from and changes to MDM. This is a powerful technique for exposing MDM as a data service, because it can support higher-level operations (such as "cleanse, match, and integrate new customer data") than can the raw MDM API.

For a detailed explanation of these tools, see Redpoint Data Management's MDM tools.

6.4.1. SETTINGS AND CREDENTIALS

In order to connect Redpoint Data Management to MDM, you will need to know:

- The MDM authentication service URI. By default this is http://localhost:9901/mdm.
- The MDM data service URI. By default this is http://localhost:12345/mdm.
- The MDM username and password. The **system** username can perform all operations, and it is often used for ETL. The password for the system logon is initially set to `system` but this should be changed administratively.

You set the default and shared settings and credentials for MDM connector tools in the Redpoint Data Management repository under `/Settings/Tools`, on the **MDM** tab. These settings can be overridden in each Redpoint Data Management MDM connector tool.

6.4.2. SELECT DATABASES AND TABLES

In every Redpoint Data Management MDM connector tool, you must choose a database from the **Database** drop-down on the **Properties** pane. If this drop-down contains the message "**ERROR**", check and correct the settings and credentials. After selecting the database, most connector tools will also require that you select a **Table** on which to operate. Once you select the database and table, other settings are available and metadata will be automatically queried from MDM as needed.

6.4.3. QUERY RECORDS

Use the MDM Input tool to read records from the MDM database. You can:

- Read an entire table.
- Query based on a simple form using field/value comparisons.

When querying records, you can also specify a timestamp to query historical data (for example, "the customer table at the end of last quarter").

6.4.4. QUERY RECORDS MATCHING INPUT DATA

Use the MDM Key Query or MDM Dynamic Query tools to return all records matching a sequence of input values.

- MDM Key Query operates only on primary keys.
- MDM Dynamic Query supports queries on any field.

6.4.5. INSERT OR REPLACE RECORDS

Use the MDM Output tool to create, replace, or upsert records into the MDM database. Configure this tool for the desired **Mode**:

- **Create**: A record with a matching primary key must not already exist.
- **Replace**: A record with a matching primary key must already exist.
- **Upsert**: If a record with a matching primary key exists, it will be replaced (otherwise a new record will be inserted).

6.4.6. QUERY VIEWS

Use the MDM Query View tool to return the results of a view in batch. It can read:

- Current data
- The latest changeset committed before a specified timestamp
- A specified pending changeset

6.4.7. EXTRACT HISTORICAL DATA

Since every record in a table has a change history, it is possible to pick a historical date and view the state of the table at that time.

Use the MDM Input tool to read records from the MDM database. A changeset or timestamp can be specified to read from a specific point in history.

6.4.8. SHARED CHANGESETS

A typical Redpoint Data Management project may have several MDM connector tools that change the database, including:

- MDM Output tool
- MDM Update tool
- MDM Set Match Groups tool

All of these tools perform their operations on a *changeset*, which is automatically committed at the end of processing. However, each tool by default will create and commit its own changeset, isolating its changes from the changes of other tools in the project. Often, you will want all of the tools in a Redpoint Data Management project (or web service) to cooperate on a single set of changes. To enable that behavior, select the **Use shared changeset** option on all MDM tools in a project.

6.4.9. DETECT AND PROCESS DUPLICATES DURING LOAD INTO MDM

When loading records into an MDM table, detecting record duplicates that already exist in the table gives you the option of deciding how you want to handle this situation.

1. If the input record does not have a unique primary key, use one of Redpoint Data Management's hashing functions (with a hashing algorithm such as MD5) to synthesize a primary key value from existing field values. Note that there are many ways a primary key value could be constructed; it is up to you to determine what makes the most sense for a given table.
2. If the input record does not have a sequence field (a timestamp or number), define a sequence field in the origin table that identifies a record version.
3. Use the MDM Key Query tool (with the embedded or constructed primary key value) to determine if the MDM table has a duplicate of the record being read.
4. If the record has no duplicate in the MDM table, use the MDM Output tool to insert the record.
5. If the record has a duplicate in the MDM table, you can:
    - Ignore the record just read.

- Use the MDM Output tool to upsert the newly-read record into the MDM table. (The MDM Output tool recognizes sequence fields, and will automatically upsert a record in the MDM table if the record being read is the latest version.)

6.4.10. ADVANCED MATCHING

MDM matching is actually done in Redpoint Data Management. The following topics drill down into the mechanics of that process.

6.4.10.1. SUPPORT FOR DIFFERENT MATCH TYPES

MDM supports multiple match types (for example, individual or household). Each match type supports match review in two modes:

- Review of automated decisions
- Manual review

By default, in either of these review modes, the match review workflow uses the match-type value **default**. A workflow can be configured to use a specific match-type value (such as **individual**).

Even if an MDM table supports multiple match types (for example, individual, household, and email), a workflow is bound to a single match type. A workflow cannot support multiple, specific match types at the same time.

6.4.10.2. MATCH GROUP IDS AND MATCH TABLE PRIMARY KEYS

Match group IDs must be 8-byte integers. Even if match group ID isn't part of an MDM table schema definition, you can still extract them and set them via the Redpoint Data Management connector tools (and Redpoint Data Management deals only with 8-byte integer match IDs).

Also, you should create match tables with a primary key of type **integer**. Match table primary keys are stored in the **always** and **never** tables, and the Redpoint Data Management match tools deal only with 8-byte integer record IDs. If you create an MDM table that supports matching, things are easier if the primary key is of type **integer** (it avoids one additional key-mapping step in the project flow).

6.4.10.3. MATCH-SEGMENTATION KEY FIELDS

Match segmentation keys must be defined and managed between Redpoint Data Management and MDM as follows:

1. In the automatic matching logic you build in Redpoint Data Management, you define one or more segmentation keys. This is standard practice for Redpoint Data Management matching.
2. Segmentation keys are strings.
3. While Redpoint Data Management is entirely in charge of matching during the initial load, it must cooperate with MDM for incremental changes because new/modified records are potentially re-matched against all match candidates and all records in the current match group.

   To support this, you must define fields in your MDM table schema to hold the segmentation keys. This field should be indexed because:

   - When Redpoint Data Management loads new records and matches them, it saves the segmentation keys along with the record data.
   - When Redpoint Data Management processes incremental new/modified records, it calculates the segmentation keys for these new records and performs a dynamic query against the

MDM table to pull all match candidates before doing the rematching. Redpoint Data Management also pulls the records for the existing match groups because they may also be affected and need to be rematched.

6.4.10.4. MATCH INFORMATION DOCUMENT

The MDM Input tool in Redpoint Data Management returns the defined fields (and any nested tables) in the MDM table schema by default. There is an option to return the match document as a JSON file that contains the match levels, scores, and overrides. This document is defined in Matching sub-document.

6.4.10.5. HANDLING MATCH SUPPRESSION

See Automatic matching is performed using Redpoint Data Management/MDM integration for an explanation of how steward decisions are honored.

Call the MDM Match Suppression Query tool to retrieve the primary keys of never-match overrides, always-match overrides, and in-match-review-records.

- **Never-match overrides**—Add the never-match primary key pairs to the match macro itself (AO Business Match, for example).
- **Always-match overrides**—Treat these primary key pairs as hard matches. These key pairs go into a post-match process (AO Associate Match IDs).
- **In-match-review records**—Use a Redpoint Data Management join to exclude these records from matching.

6.4.10.6. AUTOMATIC MATCHING

Automatic matching is performed via Redpoint Data Management-MDM integration:

1. Redpoint Data Management executes automatic matching jobs, and you set match parameters inside Redpoint Data Management.
2. Match reviews are performed by humans (data stewards) in MDM.

When stewards do a match review, their decisions are stored in the **always** and **never** keys of the match document, and this records the user preferences. In order to keep Redpoint Data Management from rematching (or failing to match) combinations that the steward has set, the Redpoint Data Management MDM Match Suppression Query tool returns lists of "always match" and "never match" pairs of record IDs, as well as a list of "under review" record IDs.

6.4.10.7. BATCH MATCHING

In MDM, batch matching means either:

- Matching records in MDM with records from another source.
- Rematching all records already inside MDM.

In either case, load records first and then run the MDM matching process:

1. Read all records out of MDM.
2. Match.
3. Figure out new match group values and set them with the MDM Set Match Groups tool.

4. Don't generate new match groups every time—use AO Constant key to incorporate them into the matching process.

6.4.10.8. REAL-TIME MATCHING

To do real-time matching:

1. For the incoming record, compute the segmentation keys.
2. Take the segmentation keys and use the MDM Dynamic Query tool to pull matching records.
3. Query match groups (find every record that's in the current record's match group). Now you have every record to throw into the match hopper.
4. Do matching.
5. Put the results into AO Constant Key—have any match IDs changed?
6. For any match IDs that have changed:
    a. Go back to MDM and generate new match group IDs.
    b. Apply these to records to set new match group IDs.

6.4.10.9. MATCH REVIEW IS CONDUCTED VIA WORKFLOW

Use the following workflow to support automatic match review from Redpoint Data Management projects (this flow can be seen in the **mdm_matchtest_with_workflow.dlp** sample project):

1. Set up two classes of matching, tight and loose. The tight matches are always acceptable (low false positive rate), but loose matches should also be considered.
2. Perform both classes of matching in the same project (parallel flows).
3. Set the tight match groups for the data using the MDM Set Match Groups tool. These will be committed and become The Truth if no further action is taken.
4. Use AO Match Group Differences to find where the tight and loose results disagree.
5. Process the differences with the MDM Start Match Review Workflow tool. This tool does several things:
    - For each match group to review, it opens a changeset.
    - In the newly-opened changeset, it changes the match group IDs to be the loose IDs.
    - It does NOT commit the changeset (it remains pending).
    - It starts an Automatic Match Review workflow that compares the accepted/committed match groups to the proposed/pending match groups.

    In this way, a single project may create many match review workitems.
6. When the steward pulls an MDM Match Review workitem from the queue, she is presented with a UI allowing her to choose between the workitem's accepted and proposed matches.

6.5. VIEWS AND SUBSCRIPTIONS

A *view* is a directed graph of data transformations (aka *projections*) that supports complex queries.

A view can be as simple as displaying a table. You can define more complex views by chaining together three basic operations:

- **Table**: Read data from a table.
- **Join**: Join two data sources together.
- **Aggregate**: Summarize groups of records, calculating a set of output variables according to rules you define.

Combining these basic operations allows you to make useful calculations. For example, you could create views to track customer value metrics over time:

- *N*-day total purchases
- Number of web visits in last *N* days
- Number of email-sent events
- Number of email-responded events

You can also create a *subscription* to a view that notifies you when the data in a view changes. See Subscribing to a view for more information.

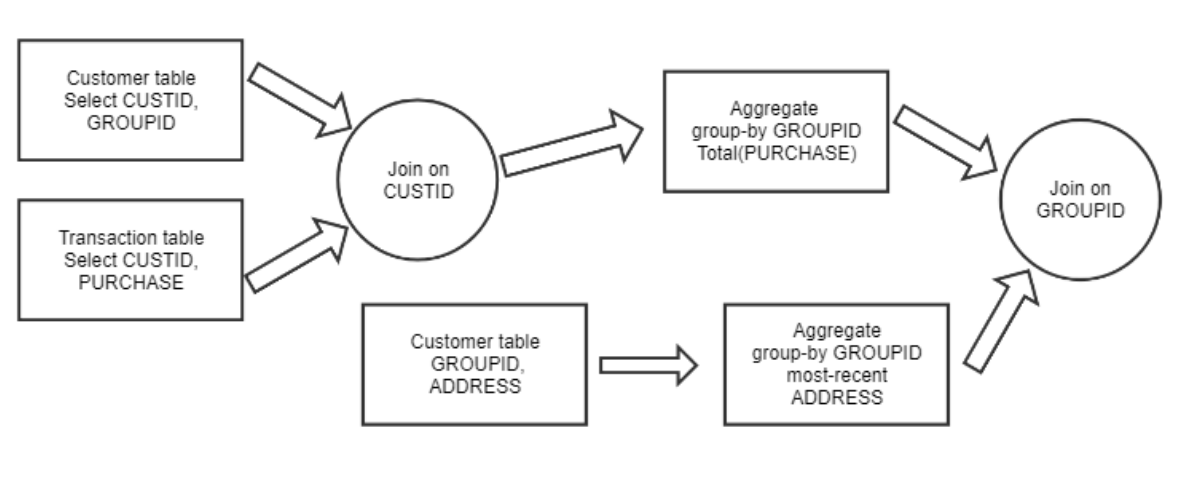
Views and subscriptions must be created and modified by a user assigned sufficient roles and permissions (for example, a user in the **dbas** group when using the demo database setup).

6.5.1. DEFINING A VIEW

Under the covers, a view is a piece of code written in Redpoint Global Inc.'s proprietary, JSON-based Aggregation Expression Language (AGL) that is entered on the **Data Hub** > **Views** page. Currently, each view must be custom-written. See Aggregation language for a detailed explanation of the language.

When you define a view, you're constructing a directed graph—an inverted tree. Inputs must be tables, which can be joined together and aggregated. Aggregations and tables may also be joined. For example, the view described in the graph below follows these steps:

1. Join customers and transactions by customer ID.
2. Aggregate the result on the customer group ID, and get the total purchase for the group of customers.
3. Join that to a different aggregation on customer group ID that gets the most recent address of the customer.



A view is described by a directed graph

Each step in the view graph is called a *projection*. Projections form the vertices of the graph.

6.5.1.1. TYPES OF PROJECTIONS

MDM supports the following projection types:

Aggregation

An aggregation projection has a single upstream vertex in the graph, and always starts with "group by" as in "group by customer ID". It consists of combining and chaining operations such as:

- Select by date
- Filter
- Top
- Sort
- Unique
- Count
- Total, Average, Min and Max
- Expression

These operations may be combined in any way that forms a legal aggregation language expression, such as:

```
$Sum($AgeDays($timestamp, 0, 30).$Sort($PurchaseAmount, Descending).$Top(10),
$PurchaseAmount) / $Count($AgeDays($timestamp, 0, 60) + Math.PI)
```

Join

A join projection has two or more upstream vertices in the graph. It is a simple join by join-key equality. There are no Cartesian joins (that is, many-to-many joins), but there are many-to-one joins. The join keys must be indexed in the upstream data.

Table

A table view simply selects columns from a table and potentially maps them to new names. It passes primary key and indexed field metadata to the downstream projection.

6.5.1.2. VIEW EXAMPLES

The following examples assume that we have two tables:

The **Customers** table is versioned, with these fields:

- id
- text1
- match_group_id

The **Transactions** table is a time-series table, with these fields:

- timestamp
- cust_id
- value
- department

6.5.1.2.1. SIMPLE TABLE VIEW

The following aggregation language code creates a view that displays all fields of the **customer** table:

```
{
  "description": "Customer Table View. Subscribe to this to get customer
changes",
  "projections": [
    {
      "projection": "customers",
      "type": "table",
      "table": {
        "table": "customers",
        "select_all": true
        }
      }
    ],
  "source": "customers"
}
```

The code is composed of the following parts:

Part	Explanation
`"description"`	A description of what the view does.
`"projections"`	The list of projections defines the data sources for data and for the chain of projections. Dependent projections are named first. This must be a tree with no dependency loops.
`"projection":"customers"`	Name your projection something meaningful to you. Projection names must be unique inside this array of projections.
`"type":"table"`	Projections are tables, joins, or aggregations. In this case, this projection is simply the customer table.

Part	Explanation
`"table":"customers"`	Table projections simply load the data from the named database collection.
`"select_all":true`	Load all of the fields. If you want to limit which columns are loaded, set `"select_all"` to `false` and set the select field to an array of the column names to be loaded.
`"source":"customers"`	Tell the view where in the projection graph to get its data. It must be the final projection in the graph.

Creating a subscription to this view will notify you of changes to the view (in this case, changes to the **customers** table). See Subscribing to a view for more information.

6.5.1.2.2. AGGREGATING DATA FROM SEVERAL TABLES TO A VIEW OF TRANSACTIONS BY CUSTOMER

The following aggregation language code creates a view of transactions by customer:

```
{
  "description": "customer_transactions",
  "projections": [
    {
      "projection": "trans_proj",
      "type": "table",
      "table": {
        "table": "transactions",
        "select_all": true
      }
    },
    {
      "projection": "cust_proj",
      "type": "table",
      "table": {
        "table": "customers",
        "select_all": true,
        "match_group_id": [
          "individual"
        ]
      }
    },
    {
      "projection": "join1",
      "type": "join",
      "join": {
        "inputs": [
          {
            "source": "trans_proj",
            "key": "cust_id",
            "map_all": true
          },
          {
            "source": "cust_proj",
            "key": "id",
            "map_all": false,
            "mapping": [
              {
                "in": "id",
                "out": "id"
              },
              {
                "in": "text1",
                "out": "text1"
              },
              {
                "in": "match_group_id",
                "out": "groupid"
              }
            ]
          }
        ]
      }
    },
    {
      "projection": "aggregation1",
      "type": "aggregation",
      "aggregation": {
        "source": "join1",
```

```
        "group_field": "groupid",
        "sub_group_field": "department",
        "output_group_field": "group_n_dept",
        "output_fields": [
          {
            "field": "sumv1",
            "datatype": {
              "type": "float",
              "size": 8
            },
            "expression": "$sum($AgeDays($timestamp,5,10),$value1)"
          }
        ]
      }
    }
  ],
  "source": "aggregation1"
}
```

The code is composed of the following parts:

Part	Explanation
`"table":"transactions"`	Get transactions from the time-series table named "transactions".
`"table":"customers"`	Get customers from the versioned table named "customers".
`"type":"join"`	Joining the customers to transactions makes a flattened record with customer information added on each transaction.
`"source":"trans_proj"`	In this case, the transaction projection is one side of the join.
`"key":"cust_id"`	The field used to match the other input.
`"map_all":true`	Use all fields and keep the same field names for the output.
`"source":"cust_proj"`	The name of the second source for the join.
`"key":"id"`	Primary key field for this table. It should match the primary key field of the other table.
`{"in":"match_group_id", "out":"groupid"}`	Rename the individual match group ID field for the output.

6.5.2. SUBSCRIBING TO A VIEW

You can create a subscription to a view that notifies you when the data in the view changes. For example, you could set up a subscription to a table that notifies you when any table records are added, modified, or deleted.

- Subscriptions provide real-time updates to view changes.
- Subscriptions support push operations to external databases.
- Each view can have multiple subscriptions

6.5.2.1. CREATE A SUBSCRIPTION

To create a subscription for a view

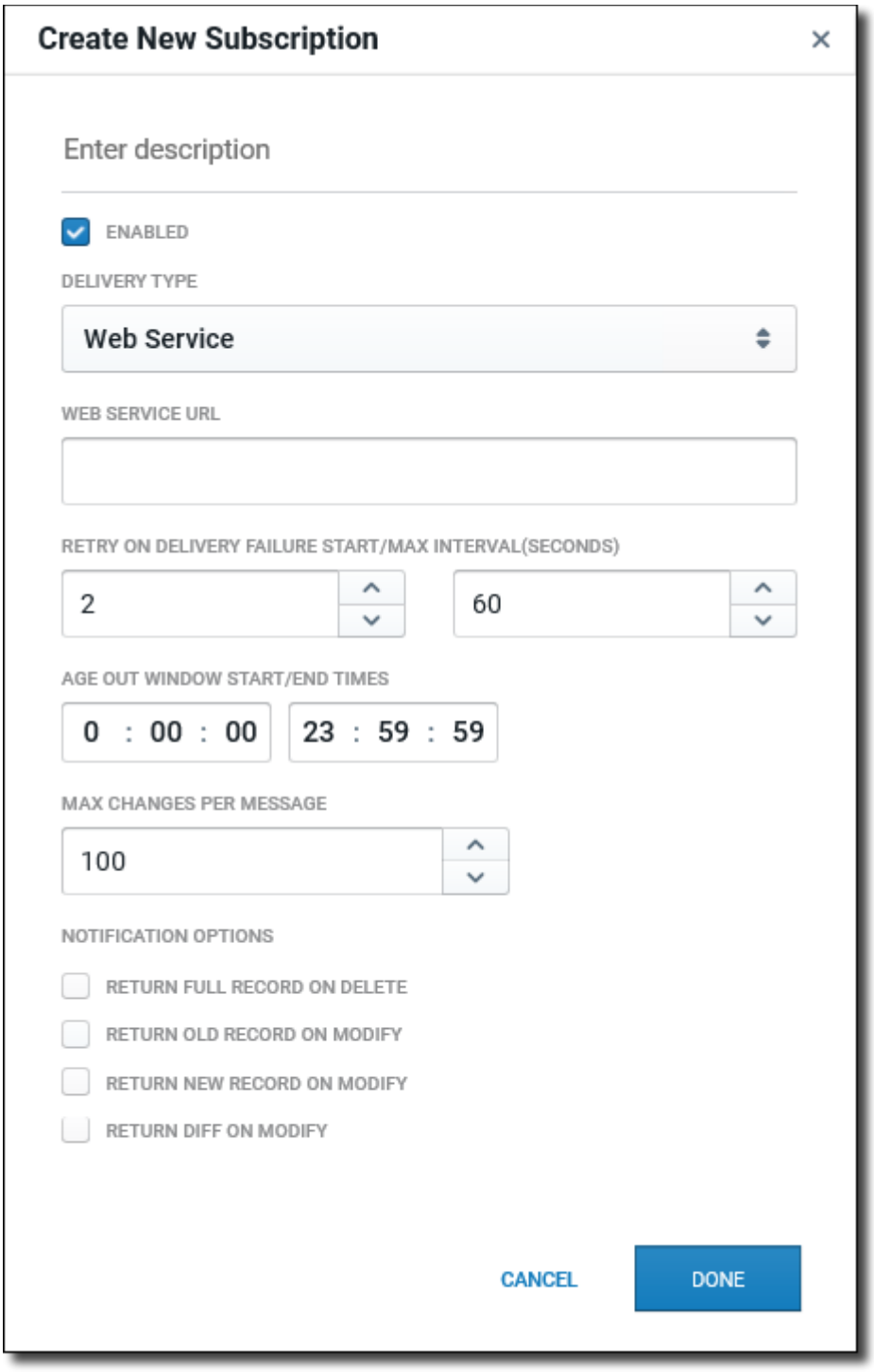
1. On the **Data Hub** > **Views** page, choose a view from the table and click **View**.

To create a subscription for a view, on the Data Hub > Views page choose a view from the table and click View

2. In the **Subscriptions** section, click the **Add** button

The Subscriptions section Add button

3. On the pop up dialog, configure subscription options:



Configuring subscription options

- **Enter a description** (name) for the view.
- Select **Enabled** to enable the view.
- Select the **Delivery type**:

Delivery type	Description
Web Service	Notifications of data changes are sent via the internet.

Delivery type	Description
File	Notifications of data changes are written to timestamped files in CSV-format, with one line per change. The first field in each line is the operation type (C, M or D for Create, Modify or Delete). Create operations list all fields, while Modify and Delete will always have the primary key filled. Other fields may be empty, depending on how **Notification options** are configured.

- Configure delivery options:

Option	Description
Web service URL	If **Delivery type** is **Web Service**, URL that will receive the notifications.
Target folder File pattern	If **Delivery type** is **File**, the folder to receive the notifications file and the file pattern to use in the generated notifications file name. This should be of the form *prefix.suffix*. The generated name will be *target folder* + *prefix* + *timestamp.suffix*. For example, given the target folder `/usr/local/mdmsub` and the file pattern `data.csv`, a typical file will be named `/usr/local/mdmsub/data_2021_01_06_18_10_37.csv`.
Write frequency	If **Delivery type** is **File**, interval at which the notifications file is written.
Write a header row	If **Delivery type** is **File**, optionally write field names to the first line of the notifications file.
Retry on delivery failure	After a notification delivery failure, number of seconds (*x*) to wait before trying to resend a notification.
Start/max interval	Number of seconds (y) to keep trying to resend a notification. Taken together, this means after a notification failure, wait x seconds before trying to resend the notification, then keep trying to resend the notification for y seconds. If this time period expires, wait x seconds before trying again. Continue this cycle indefinitely until a notification is successfully sent.
Age out window	Some aggregation rules have a filter such as $AgeDays(datefield, N) (which means "select everything up to N days old"). Of course, this selection changes daily and these record choices must be recalculated. The Age-out selector lets you specify a daily (24-hour) window as to when MDM is allowed to do this potentially computationally-expensive process. Values are 24-hour time in the form HH:MM:SS. If no values are entered, the form defaults to start="23:00:00" and end="23:59:59". If start > end the calculation happens between PM and AM (for example, 23:00 - 01:00). If start < end the calculation happens during a contiguous chunk during the same day (for example, 00:00 - 04:00).
Max changes per message	Maximum number of record changes to include in a notification message. If the number of record changes at a given time exceeds this value, multiple messages will be sent. The max value is 1000.
Notification options	By default, a notification simply tells you what records were deleted or modified. Select one or more of these options to include whole records (or what changed in each record) in the notification message.

4. Click **Done** to finish.

6.5.3. QUERYING MDM VIEW DATA WITH A DM CONNECTOR

Inside a Redpoint Data Management project, you can use the MDM Query View tool to pull all of a view's output in batch. This tool allows you to:

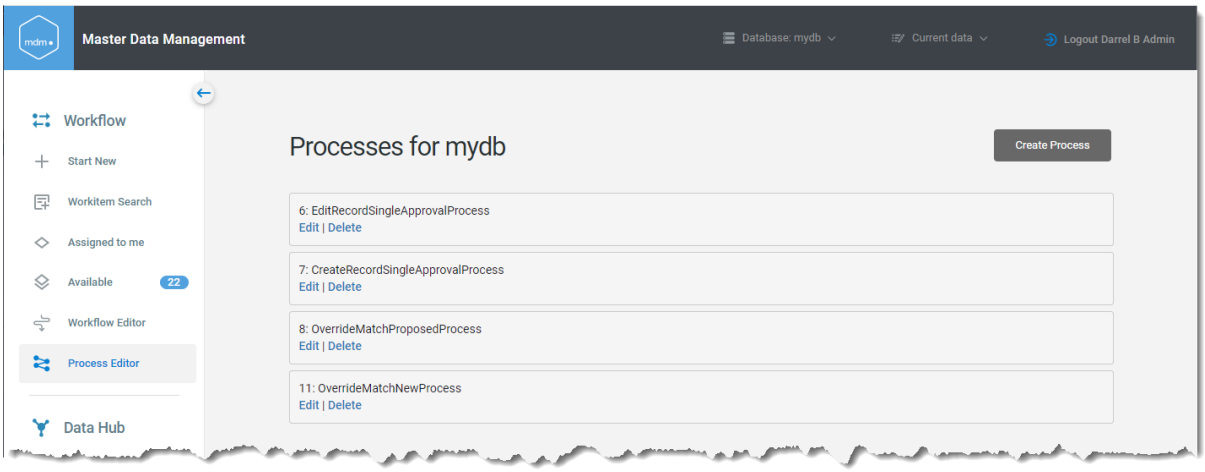
- Export the view data to some other format/database.
- Validate the results by hand.
- Validate the view using processing steps in the Redpoint Data Management project.

6.6. ADVANCED WORKFLOW

In a *workflow*, users perform a series of actions (a *process*) on a specific database and table. For example, in one business user workflow, a business user adds a new customer to table *x* in database *y*"). Processes are defined using Business Process Model and Notation (BPMN) and publish as BPMN 2.0 XML files.

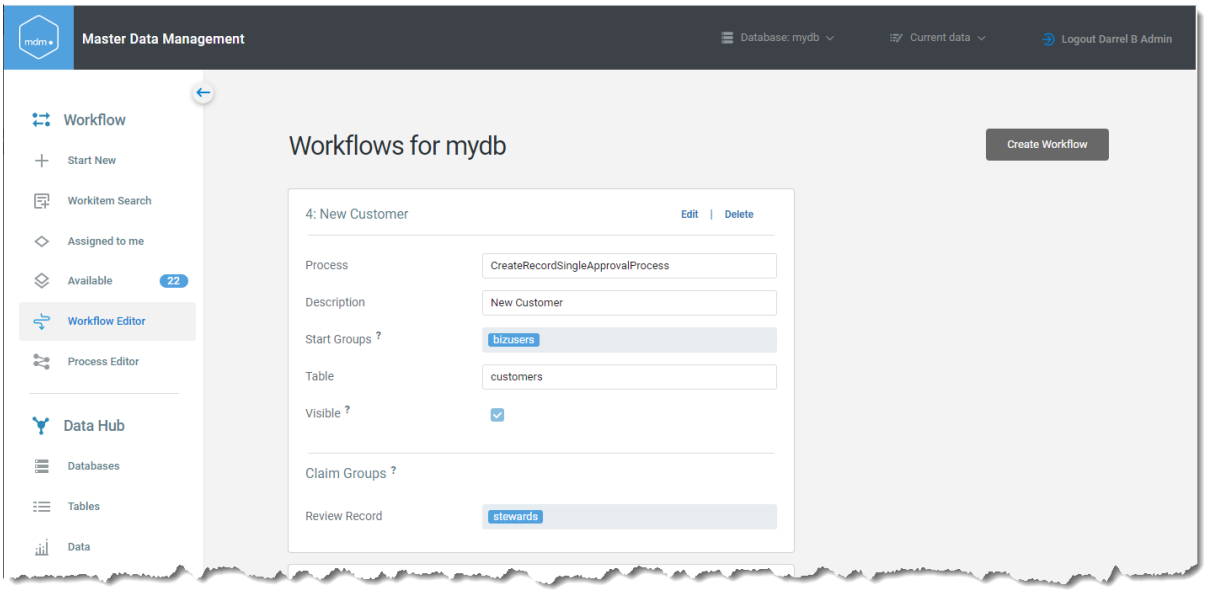
Users in the **system** or **dbas** groups can use the **Process Editor** to create, edit, and delete BPMN processes, and the **Workflow Editor** to create, edit, and delete workflows that implement these processes.

Process Editor (system and dba users only)



Create, view, edit, and delete processes in editor

Workflow Editor (system and dba users only)



Create, view, edit, and delete workflows in editor

6.6.1. STANDARDIZED WORKFLOWS

Organizations typically use a small set of standard workflows. Two common workflows are described below.

Data steward review of business user changes

In these workflows, a business user can create new records or change existing ones. The new or modified record is forwarded to a data steward, who can approve, reject, or send the record back to the originating business user to make changes (the data steward can include comments for the business user explaining desired changes). Once the data steward approves the record, the record's enclosing workitem is committed to the database.

See Enter a new record and Edit an existing record for detailed descriptions of these workflows.

Data steward review of marginal matches

In the automated match review workflow, if a Redpoint Global Inc. Redpoint Data Management ETL process has automatically identified potential duplicate records, a data steward can review these match groups and edit, accept, or reject them. For example, you could review individual match level for new customers. See Review automated matches for a detailed description of this workflow.

In the manual match review workflow, a data steward can manage duplicate records in a table by querying the table to identify likely duplicate records and manually grouping them into match groups. For example, you could review and change current customer matches at individual match level. See Review manual matches for a detailed description of this workflow.

6.6.2. WORKFLOW ROADMAP

All MDM workflows follow a similar series of steps:

- **Validation rules:** After creation, MDM validates an initial workitem before it is placed in the workitem queue. For example, if a Redpoint Data Management ETL process has automatically identified potential duplicate records and forwarded the information to MDM, MDM will validate

this Redpoint Data Management data before it places the resulting workitem in the MDM workitem queue.

- **Dashboard notification:** Once the initial workitem is created and added to the workitem queue, the MDM dashboard will indicate to users in the appropriate group or groups that a workitem is ready to be claimed. For example, when a business user adds a new customer, the workitem is added to the queue and MDM notifies users in the **data steward** group that a new workitem is available to be claimed by them.
- **Escalation:** After a workitem is claimed and processed by an appropriate user, the workitem is escalated to the next level. For example, if a data steward reviews a new customer record and rejects it, MDM routes the record back to the originating business user so that user can make requested changes.
- **Quarantine:** Once a workitem has traveled through a workflow, it remains in the workitem queue until it is explicitly committed. Only after a workitem is committed are the changes are made to the table. For example, once a new customer record has been reviewed by a data steward and approved, MDM commits the changeset. (The changeset is committed by "system" because all data changesets are committed by MDM on behalf of the user).

6.7. GDPR AND PRIVACY SUPPORT

The increase of data breaches and data-related threats has led to the enactment of numerous statutes related to data privacy. The GDPR (General Data Protection Regulation) is the most significant of these regulations, and has served as a model for laws in other jurisdictions.

Implemented in May 2018, the GDPR significantly increased the data privacy rights of consumers and the requirements on companies that solicit and retain customer identities. It requires that any organization that manages or stores the personal data of EU citizens offer individuals the right to request and obtain any Personally Identifiable Information (PII), and in certain circumstances also have their data erased. It also levies hefty financial penalties for non-compliance.

6.7.1. DATA VISIBILITY

In MDM, data visibility is controlled via options set in the table schema:

- The **Display** option must be selected for a field to be visible in any view of the table. Fields that are not display-enabled cannot be queried via the web interface.
- The **PII** option can be selected and a masking hash applied to remove sensitive field data. In a masked record, all columns marked with PII are erased. When viewing masked records, PII fields are indicated by using asterisks (*) as placeholders for the removed values. Those values have been removed from the database.

6.7.2. MASKING ERASES SENSITIVE DATA FIELDS

Data disclosure rules tend to be more relaxed for *pseudonymized* data (information that has been secured via data masking, encryption, and hashing). This masking process lets you retain data while “de-identifying” it. In MDM, you can select columns containing PII and apply a masking hash. If a request is made to mask an individual’s data, you can quickly locate the requested record and mask it. The record will remain in the system, but columns marked as PII cannot be queried, and their field values will be displayed as asterisks (*).

6.7.3. ERASING (PURGING) PERMANENTLY DELETES RECORDS

The **Erasure** operation permanently deletes entire records from a table, throughout the entire history of the table.

The **Verify Erasure** operation lets you confirm whether a record was masked or purged by querying the value of a hashed PII field.

6.7.4. MASKING AND PURGING CREATE HASH ENTRIES

When you mask or purge a record, MDM creates a unique hash value for each mask hash defined for the table. When you choose fields for your mask hashes, ensure that the fields used to generate the mask hash are, in combination, unique for the record or set of records you want masked or erased.

For example, in table **customers** you have defined three mask hashes:

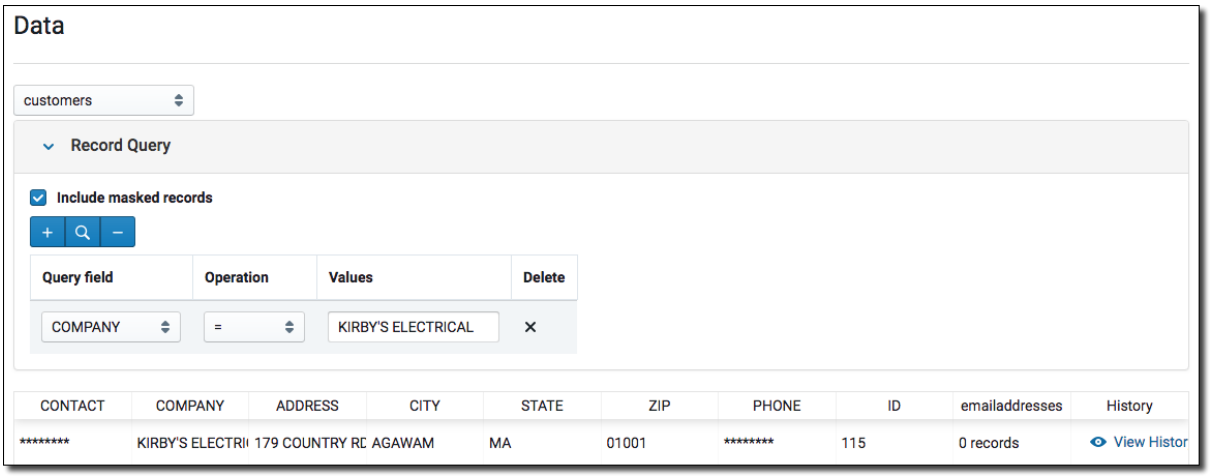
1. **name** (hashes the **CONTACT** field)
2. **address** (hashes the **ADDRESS** field)
3. **email+phone** (hashes both the **EMAIL** and **PHONE** fields)

When you mask or purge a record in table **customers**, MDM creates three hash values for the record (one for each of the defined mask hashes). Mask hashes must be comprised of a unique set of fields. A hash cannot be a subset or superset of another hash. For example, if you have a mask for **name** and one for **address** you cannot have one for **name+address**.

6.7.5. CHOOSE MASKING OR PURGING BASED ON CUSTOMER NEEDS

If a customer requests that their PII be secured, mask their record. If a customer requests that you delete all their information, purge their record.

6.7.6. MASKING AND PURGING



Data record with masked fields

In general, record masking is done if a customer asks for their private data to be masked, or if a company's policy requires it.

Record masking is usually available to users with low-level permissions (such as a business user).

Record purging is usually available only to users with high-level permissions (such as a data steward).

Some additional notes on masking and purging records:

- A masked record does not appear at all in Data Hub queries unless you define a query on a field that is *not* configured as PII and select the **Include masked records** box. If you do so, masked records are displayed in the data grid with asterisks (*) instead of data in PII fields.
- You cannot query or view masked records from a workflow workitem.
- You can either mask or purge a record, but not both.

6.7.6.1. IDENTIFYING PII IN TABLE SCHEMAS

The masking and purging functions are available only for tables that have been configured with one or more PII columns with associated masking hashes. Field hash values are needed so that MDM can verify whether a record has been masked or purged without having to retain the PII data in MDM.

PII is defined as any data that could potentially identify a specific individual. Any information that can be used to distinguish one person from another and can be used for de-anonymizing anonymous data is considered PII. Under the GDPR, PII may include information like transaction histories and IP addresses in addition to sensitive personal and financial data. You should consult with your organization's Privacy Officer to ensure that you identify PII in your data assets and comply with applicable laws, regulations, and policies.

You should know the following about your data:

- **Which fields contain PII**? Typical fields to mask are:
  - name
  - phone number
  - address
  - email address
  - account number
  - Social Security number
  - gender
  - birth date
- **Which PII fields do I want to use for querying masked or erased records?** Typical mask hash query fields are:
  - account number
  - the individual's full name
  - primary email address
  - phone number

  If your organization looks people up using a special code, make a hash of that code.

6.7.6.2. HASHING

Hashing supports future verification that a record has been masked or purged.

When a record is masked or purged, the record's query fields are hashed to create a verifiable record of the masking or purge, and the original PII fields are erased.

Multi-field vs. single-field hashes

Multi-field hashes are not as common as single-field hashes, but may be necessary if you need more than one field to correctly identify a person (for example, full name is expressed as **First Name** and **Last Name** fields). A multi-field hash narrows the possible matches so you can make sure you've identified the right person. Do not use insufficiently unique fields (such as **First Name**) as a mask.

Single vs. multiple hashes

This depends on your data and how your organization looks up people. Hashes on email, name, account number, and phone number fields are typical, though some organizations may need only account number or a name-and-address multi-field hash. Remember that each hash should correspond to one person/contact.

6.7.6.3. RECORD HISTORY FOR MASKING AND PURGING

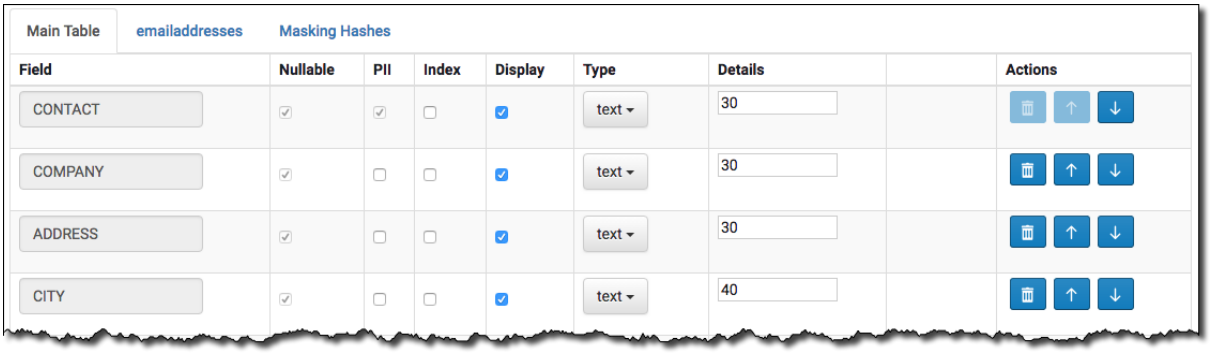
When a record is masked, PII is erased from all versions of the record (current and historical).

When a record is purged, all versions of the record (current and historical) are permanently deleted from the table. You cannot query purged records, except to verify that they have been erased.

6.7.6.4. CONFIGURE A TABLE FOR MASKING AND PURGING

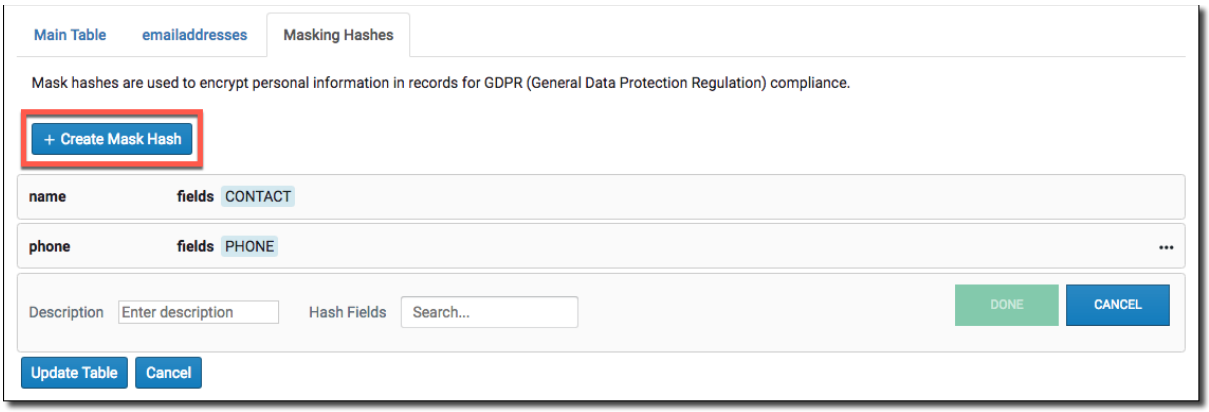
To configure a table for masking and purging

1. Go to the **Data Hub** > **Tables** page.
2. Locate the desired table and click its **Schema** button.
3. On the **Edit Table** page, locate each **Field** you want to designate as **PII** and select its **PII** box.



To designate a record field as PII, select the field's PII box on the Edit Table page

4. Click the **Masking Hashes** tab.
5. Click **Create Mask Hash**.



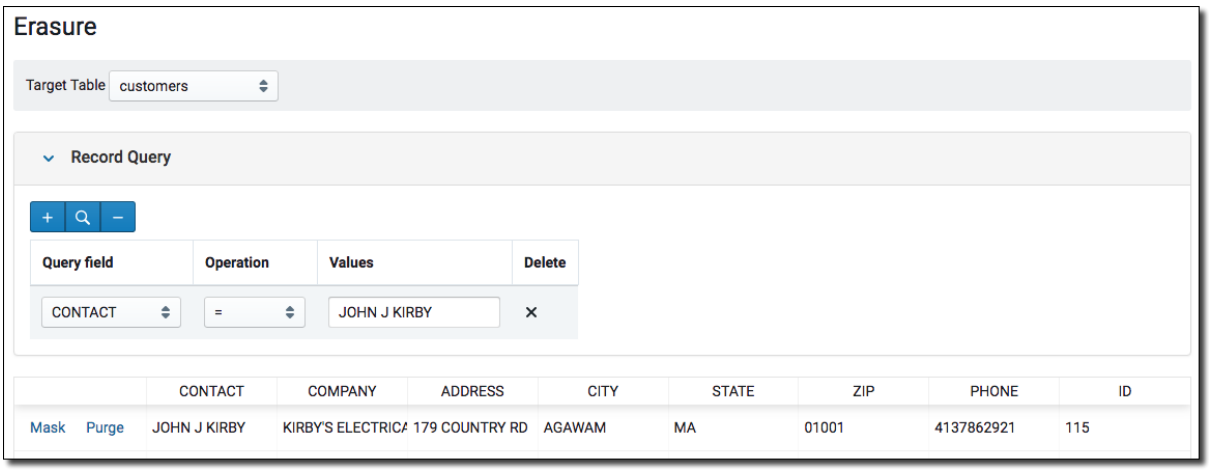
To create a mask hash for a table, on the table's Edit Table page, click the Masking Hashes tab and click Create Mask Hash

6. For each mask hash you want to create:
   a. Enter a **Description** for this hash.
   b. Click inside the **Hash Fields** box and select one or more PII-enabled fields.
   c. Click **Done**.
7. Click **Update Table**.

6.7.6.5. MASK OR PURGE A RECORD

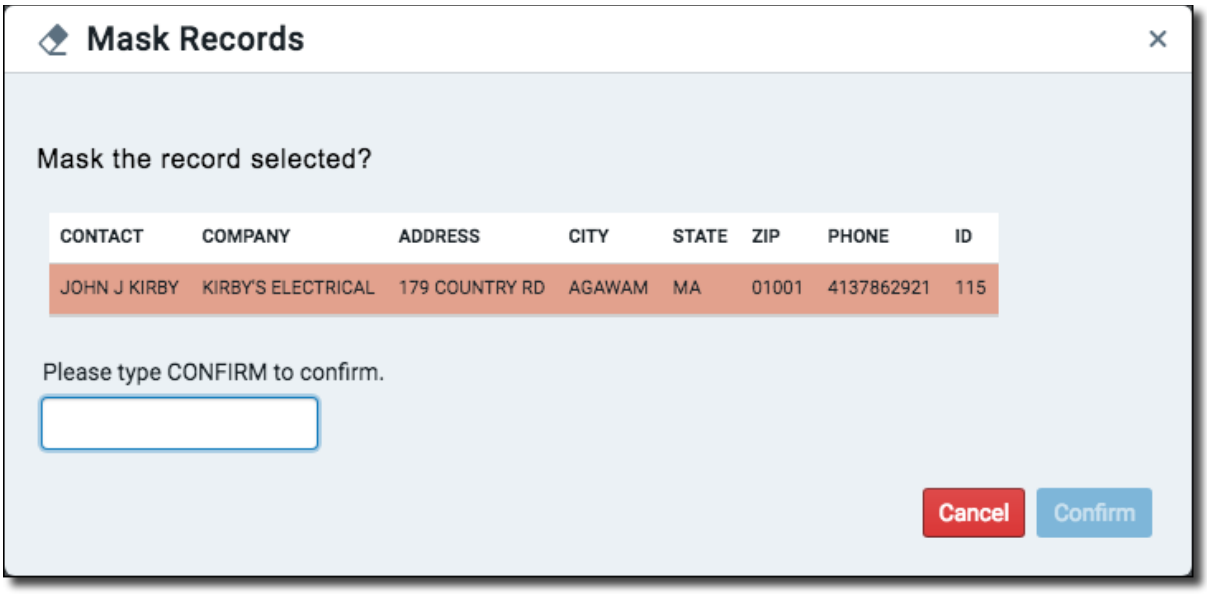
To mask or purge a record

1. On the **Compliance** > **Erasure** page, select a **Target Table**.
2. Execute a query to locate the customer record.

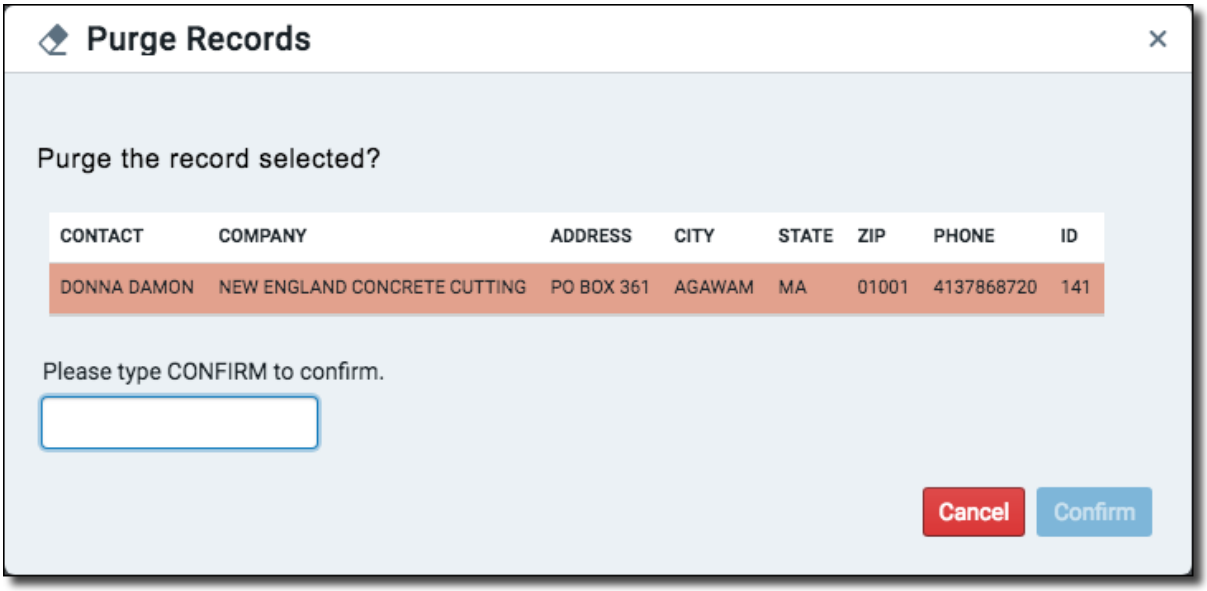


To mask or purge a record, first execute a query to locate the record

3. Find the record that you want to mask or purge and click its **Mask** or **Purge** link.
4. Examine the **Mask/Purge Records** pop up to confirm that the selected record is correct. Enter "confirm" and click **Confirm**. *Warning! When a record is purged, all versions of the record (current and historical) are permanently deleted from the table.*



Mask Records confirmation dialog



Purge Records confirmation dialog

6.7.6.6. VERIFY A RECORD WAS MASKED OR PURGED

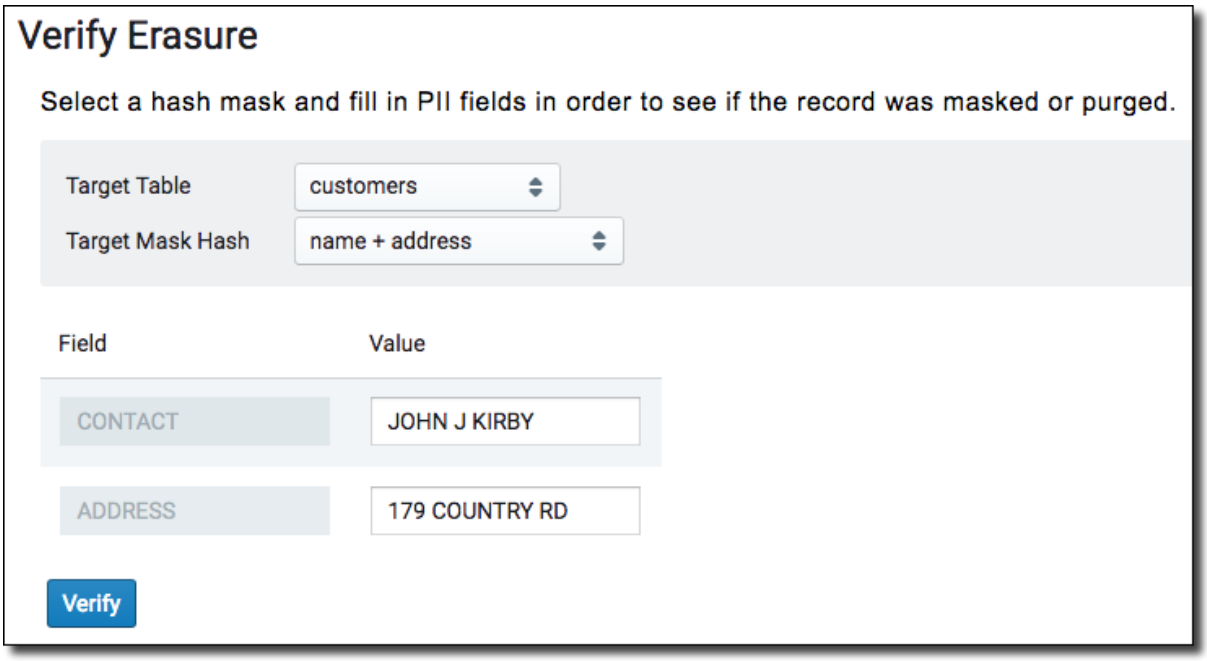
The **Verify Erasure** page lets you query tables with a generated mask hash for one or more PII fields and confirm whether a record has been masked or purged. The query contains one or more values that are checked against a table containing hashes for records that have been masked or purged. MDM disassociates the masked record from the hash code, so the query returns not the masked record, but the hash information showing what action was taken (masking or erasure) and when the action occurred.

The verify masked/purged record operation is usually available to users with low-level permissions (such as a business user).

The verify masked/purged record operation only confirms if a particular record was masked or purged. The record itself cannot be located this way.

To verify that a record was masked or purged

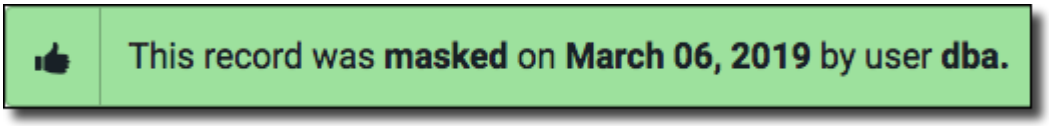
1. On the **Compliance** > **Verify Erasure** page, select a **Target Table**.



Compliance > Verify Erasure page

2. Select a **Target Mask Hash**. (You must have already defined at least one mask hash.)
3. For the selected mask hash, enter the field value(s) of the target record.
4. Click **Verify**.

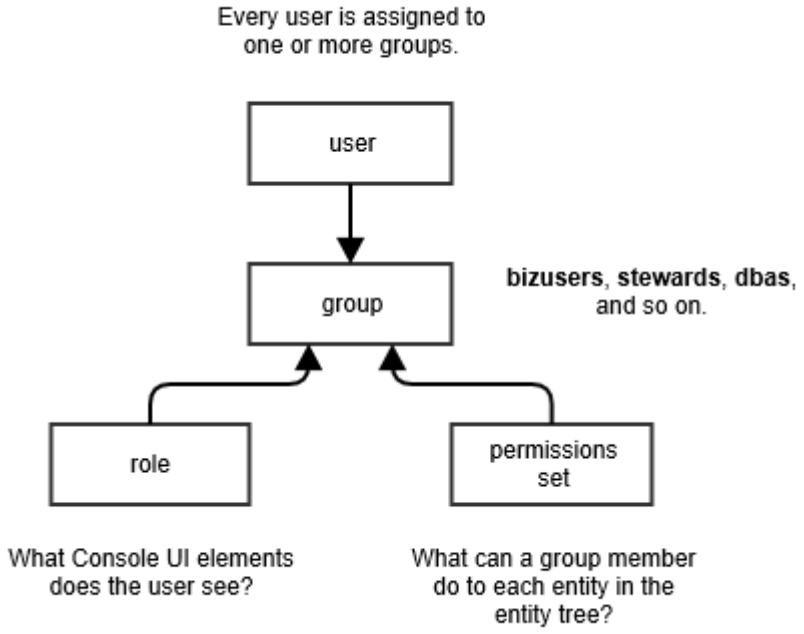
   If a matching hash value is found, a success message is displayed:



   If no matching hash value is found, a failure message is displayed:

6.8. SECURITY

MDM security is constructed using the following elements:

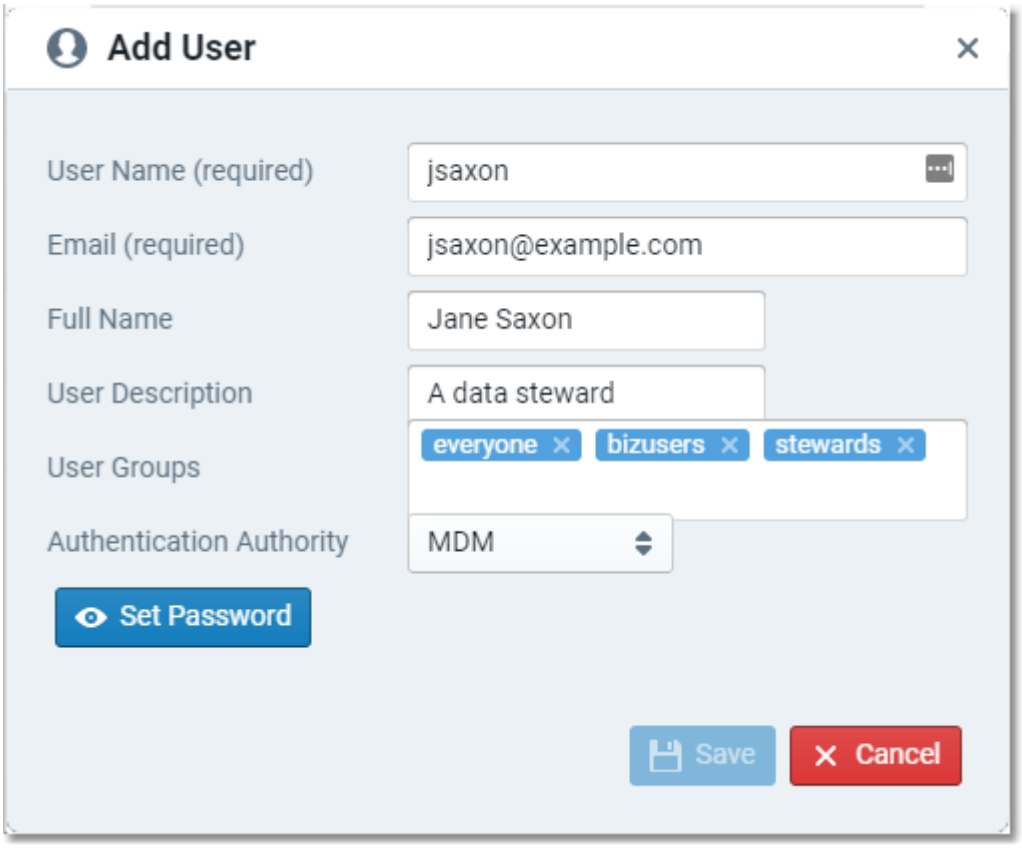


MDM security elements

6.8.1. USERS AND GROUPS

Each person who logs on to MDM is an individual *user*. Each user is assigned to one or more groups. The system administrator defines the users and groups that are appropriate for the site.

For example, this user belongs to the **everyone**, **bizusers**, and **stewards** groups:



Adding a user

The following groups are the default configuration when MDM is initialized with the demo database:

- **bizusers**: Low-privilege group designed for call centers and field representatives.
- **stewards:** Medium-privilege users with authority to make data decisions.
- **dbas:** High-privilege users that can make data model changes and bulk data changes.
- **officers**: Not assigned to any role in particular; reserved for future high-privilege users.

For example, one could have a business user "srogers" who belongs to both the **everyone** and **bizusers** groups, and a data steward "nromanoff" who belongs to the **everyone** and **stewards** groups.

MDM security features can be modified by users in the **dbas** and **system** groups.

6.8.1.1. SPECIAL USERS AND GROUPS

The following users and groups are built into MDM and cannot be modified or deleted:

Everyone group

Users are automatically assigned to the **everyone** group. A user cannot be removed from the **everyone** group, and the **everyone** group cannot be deleted. The **everyone** group is typically granted read-only data privileges.

System group

A user assigned to the **system** group has permission to do anything in MDM. This group cannot be deleted.

System user

The **system** user always belongs to the **system** group and cannot be deleted or removed from the **system** group.

6.8.1.2. USER OPERATIONS

The following operations can be performed on users:

- Browse users
- Create a user
- Edit a user
- Delete a user

6.8.1.2.1. BROWSE USERS

To browse users

1. In the **Administration** section, click **Users**.
2. The **User Management** page displays a list of Redpoint MDM users.

List of configured users

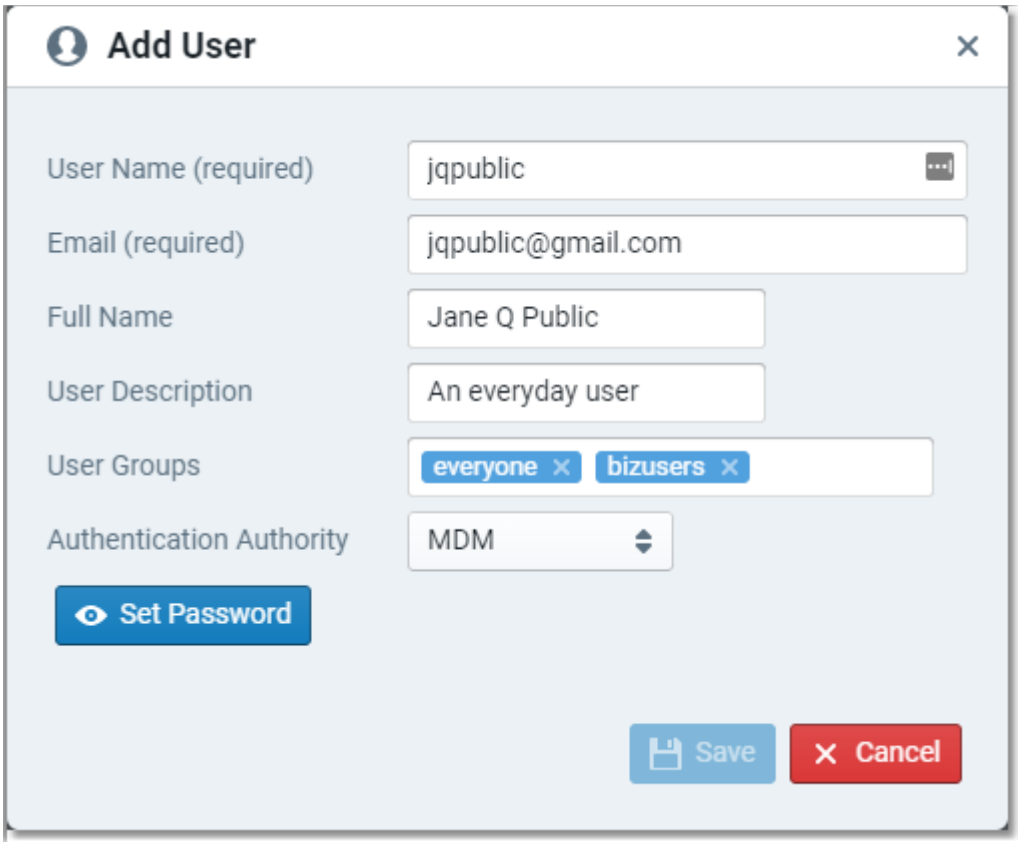
6.8.1.2.2. CREATE A USER

To create a user

1. In the **Administration** section, click **Users**.
2. On the **User Management** page, click **Create User**.

The User Management page allows you to create a new user

3. Enter the user information, deleting or adding **User Groups** as necessary.



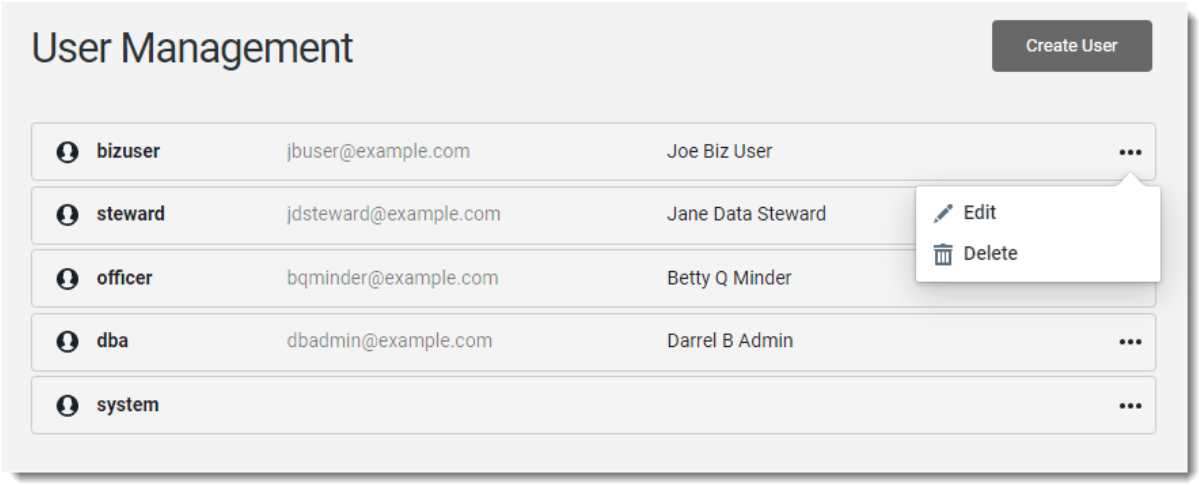
Adding a new user

4. Click **Authentication Authority** and select the authentication service used to validate the user:
   - Choose **MDM** to use Redpoint MDM's built-in user validation.
   - Choose **Active Directory** to validate the user with Active Directory authentication services. Redpoint MDM must be configured to use Active Directory, and **User Name** must match a defined Active Directory user. See <link to AD integration topic>.
5. If **Authentication Authority** is **MDM**, click **Set Password** and define the user password.
6. Click **Save** to finish.

6.8.1.2.3. EDIT A USER

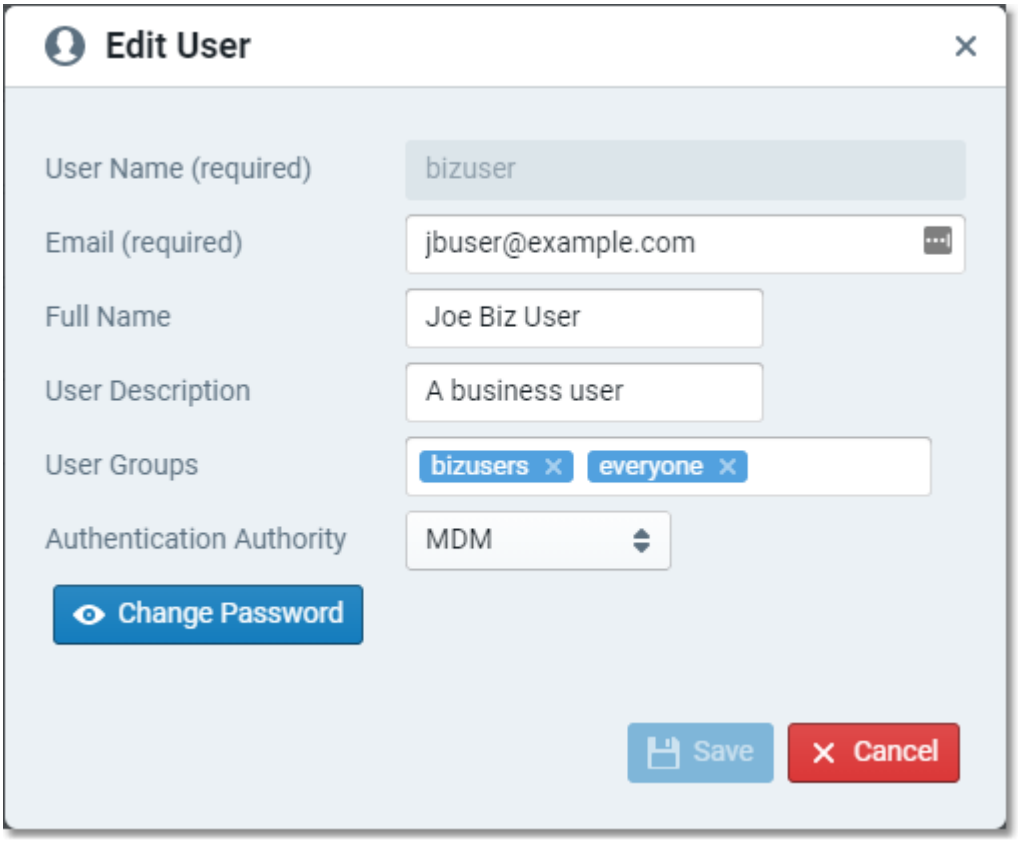
To edit a user

1. In the **Administration** section, click **Users**.
2. On the **User Management** page, click the menu icon ••• next to the desired user and select **Edit**.



Selecting a user to edit

3. Edit the user information, deleting or adding **User Groups** as necessary.



Editing a user's information

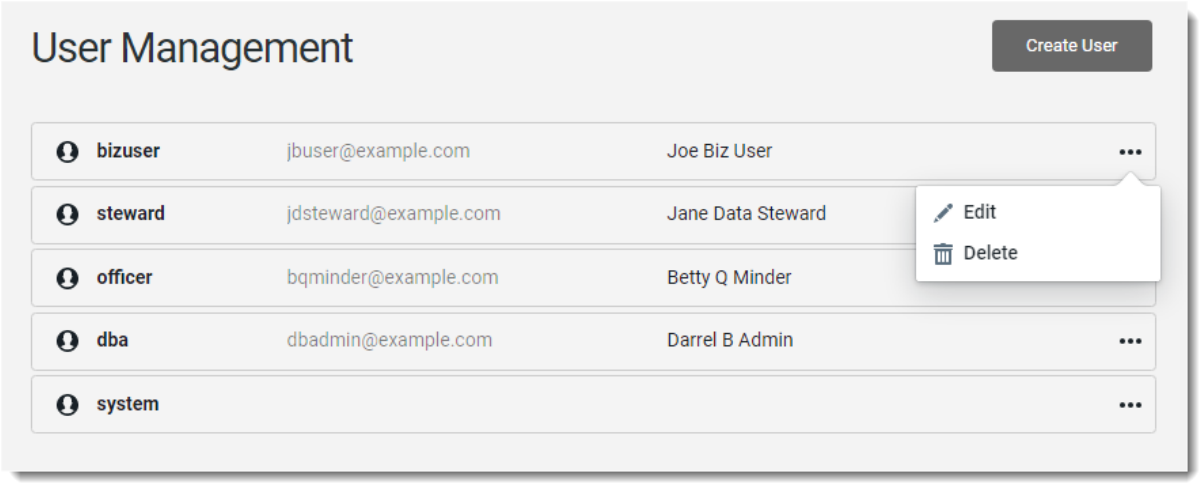
4. Optionally, click **Change Password**.
5. Click **Save** to finish.

6.8.1.2.4. DELETE A USER

To delete a user

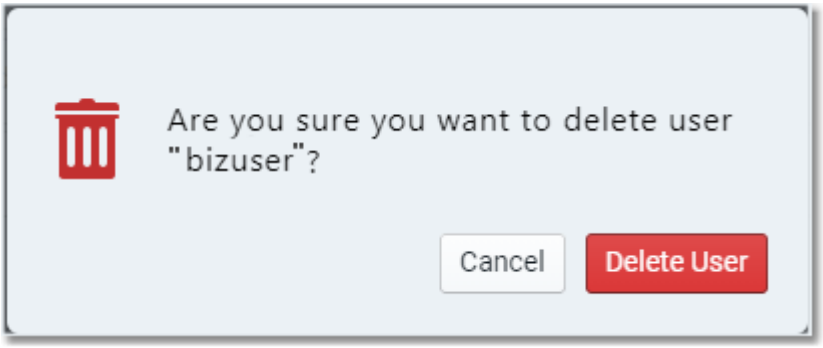
1. In the **Administration** section, click **Users**.

2. On the **User Management** page, click the menu icon ••• next to the desired user and select **Delete**.



Deleting a user

3. Click **Delete User**.



Confirming before deleting an existing user

6.8.1.3. GROUP OPERATIONS

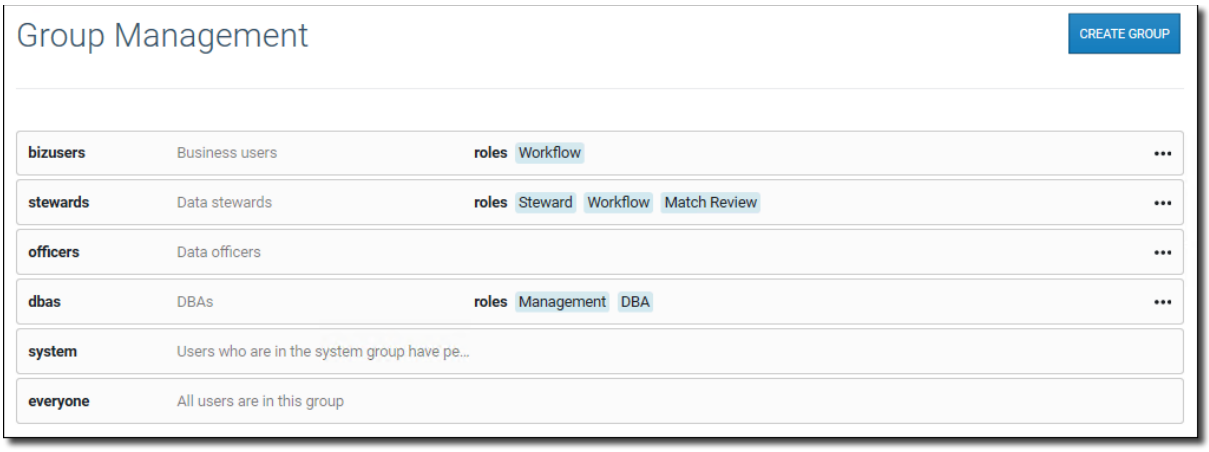
The following operations can be performed on groups:

- Browse groups
- Create a Group
- Edit a group
- Delete a group

6.8.1.3.1. BROWSE GROUPS

To browse groups

1. In the **Administration** section, click **Groups**.
2. The **Group Management** page displays a list of groups.

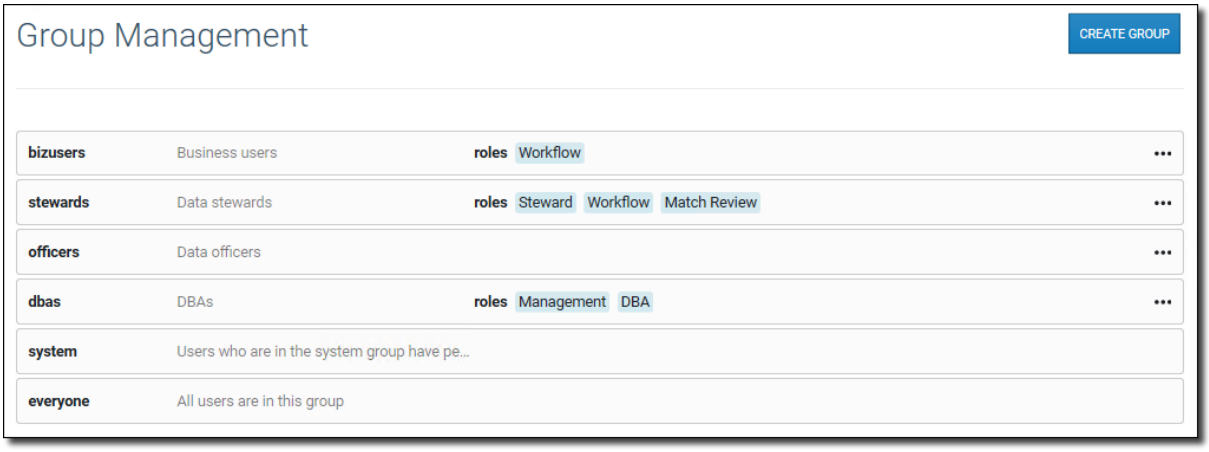


Displaying a list of all groups

6.8.1.3.2. CREATE A GROUP

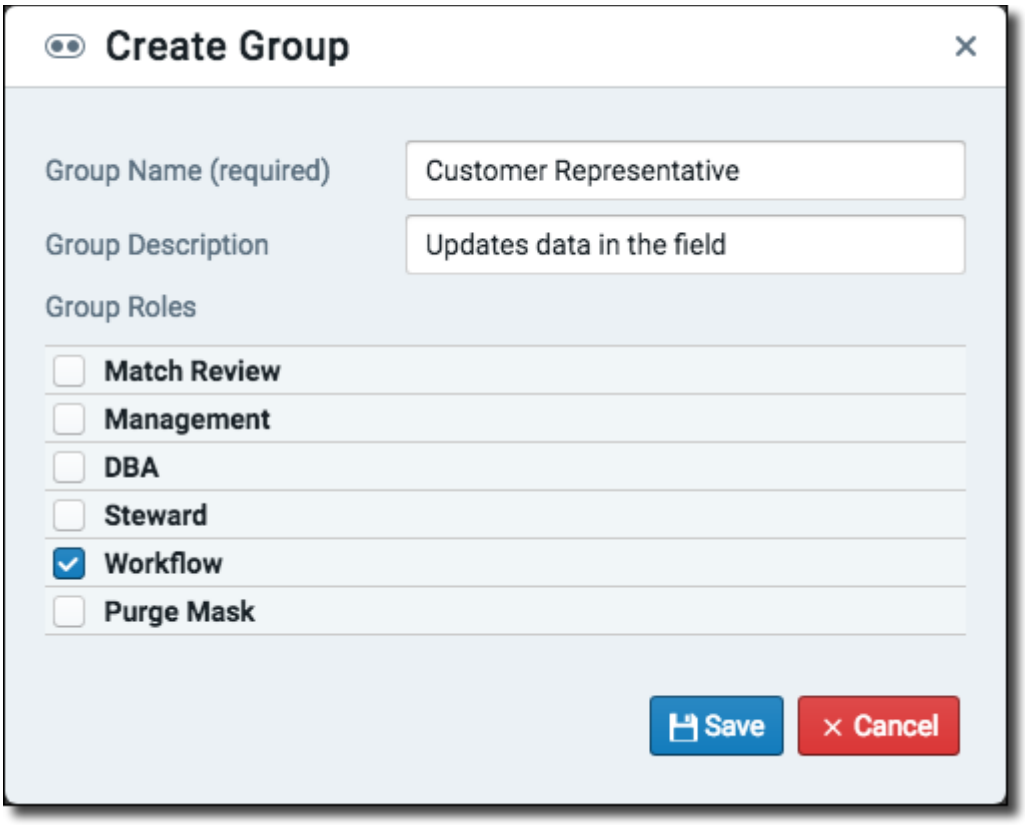
To create a group

1. In the **Administration** section, click **Groups**.
2. On the **Group Management** page, click **Create Group**.



Creating a new group

3. Enter the **Group Name** and (optionally) a **Group Description**, and then select the **Group Roles**.



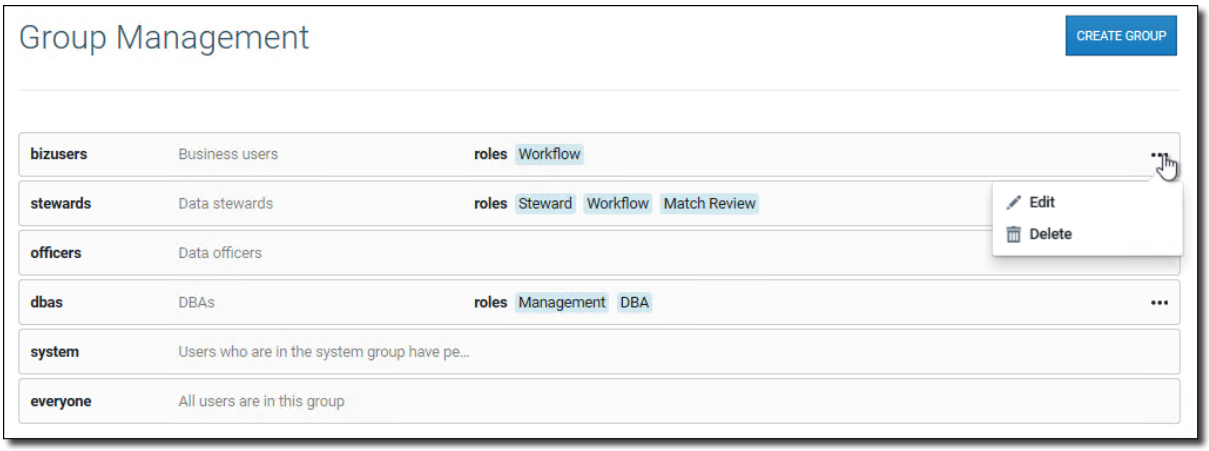
Defining a new group and choosing group roles

4. Click **Save** to finish.

6.8.1.3.3. EDIT A GROUP

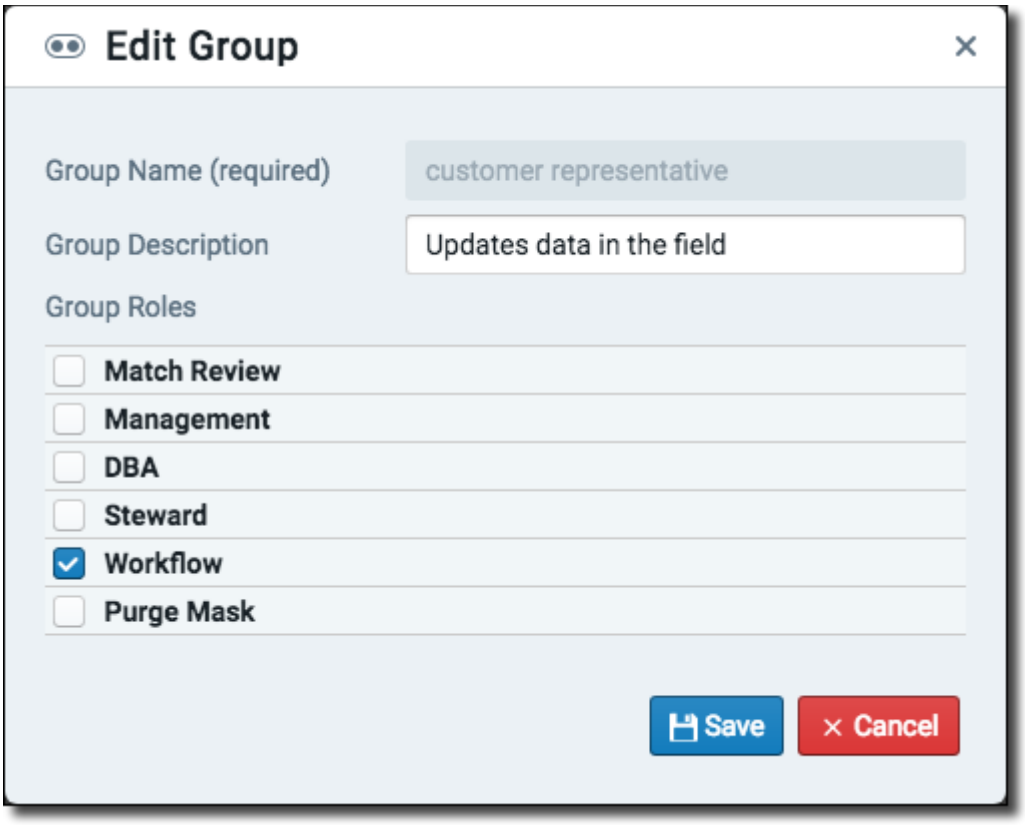
To edit a group

1. In the **Administration** section, click **Groups**.
2. On the **Group Management** page, click the menu icon ••• next to the desired group and select **Edit**.



Selecting a group to edit

3. Edit the **Group Description** and **Group Roles** as necessary.



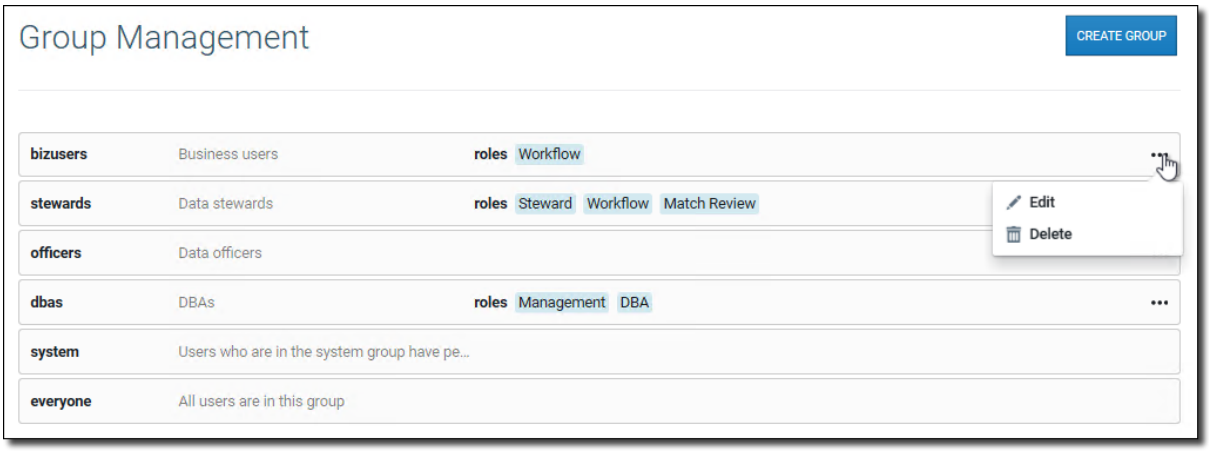
Editing an existing group

4. Click **Save** to finish.

6.8.1.3.4. DELETE A GROUP

To delete a group

1. In the **Administration** section, click **Groups**.
2. On the **Group Management** page, click the menu icon ••• next to the desired user and select **Delete**.



Deleting an existing group

6.8.2. GROUP ROLES

Group roles control what each user can access in the MDM Web App. Roles are attached to groups, and each role defines a set of user interface elements that are displayed for members of a given group. Groups can have multiple roles. The **stewards** group shown below is assigned the **Steward**, **Workflow**, and **Match Review** roles.

Roles are attached to groups, and determine which user interface elements are displayed to group members

As an example, user "tstark" is a member of the **dbas** group (which is assigned the **DBA** role). User "bbanner" is a member of the **bizusers** group (which is assigned the **workflow** role). For "tstark", the Web App displays both the **Create Database** and **Refresh** buttons, but "bbanner" only sees a database **Refresh** button.

The following table lists the user interface elements available to each role:

Role	Permissions
Management	Manage (add/remove/update) these entities: • users • groups • processes • workflows • permissions • id generators • exports
DBA	Manage metadata (data model) editing: • create database • delete database • create table • modify table • delete table • specify associations and linkages
Steward	Ad-hoc data Create, Read, Update, and Delete (CRUD) and changeset management:

Role	Permissions
	• create record • delete record • modify record • create/select pending changeset • commit pending changeset • review and manage/undo pending operation • conflict view and resolution
Workflow	Workflow menu is available to: • Start a new workflow • View and claim available workflows • Continue work on assigned workflow • View workflows user has started
Match Review	Enable ad-hoc (no workflow) interfaces related to record matching: • suspect review • force match • break apart match
Purge Mask	Allow PII (Personally Identifiable Information) actions. These actions address requirements of the GDPR (General Data Protection Regulation). These actions include: • Masking PII fields in records. • Purging (removing) PII records. • Determining if a record with PII data has been masked or purged, and if so, by whom and when. For more information on GDPR, refer to *Right of access by the data subject* and *Right to erasure* (articles 15 and 17 of the GDPR code). This source provides a concise reference, and the complete regulation can be found here.

A user implicitly has the roles of all groups to which the user belongs. A user whose groups are not assigned any roles can see only the common Web App elements available to everyone.

6.8.3. PERMISSIONS AND ENTITIES

Permissions define what operations a member of a group can do to each entity in the system.

The MDM entities are:

- System
- Database
- Table
- View
- Process
- Workflow
- Subscription

In the MDM permissions model:

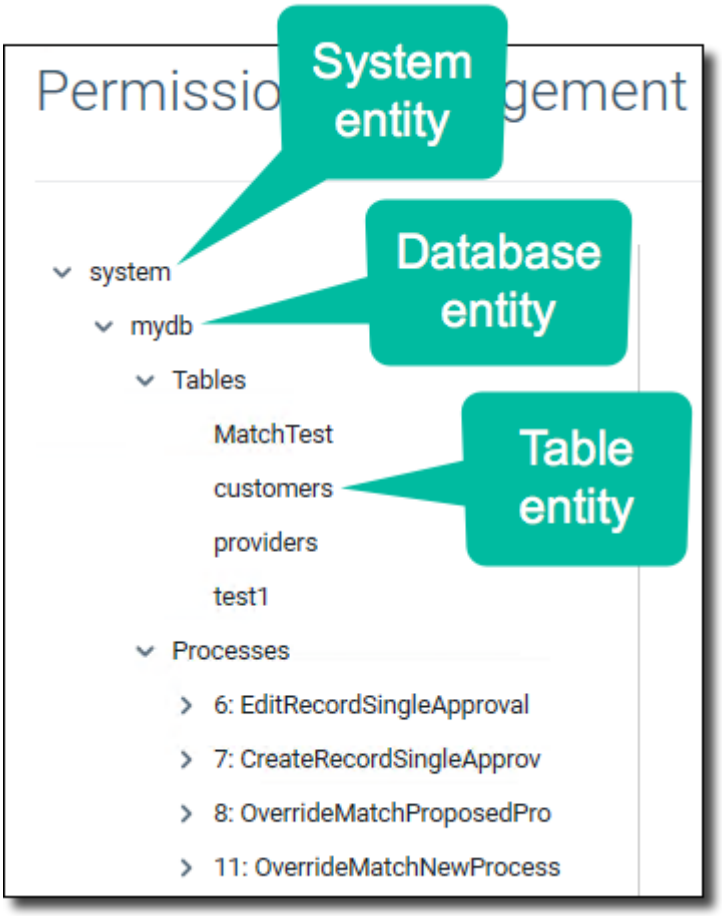
- Permissions (to operate on entities) are assigned to groups.
- A user is allowed to do anything that any of her groups is allowed to do.
- For a given group, entity permissions are inherited by the entity's children, but may be explicitly overridden.

Note that ***permissions*** *are not* ***roles***! Group roles control which user interface a user sees in the Web App, while permissions control operations on entities. Permissions and roles should align to avoid confusion.

The **system** group is assigned all permissions. A user in the **system** group can do anything.

6.8.3.1. PERMISSIONS INHERITANCE

The root entity of the permissions hierarchy is **system**. All other entities are children of **system**. In the example below, **mydb** is a child entity of **system**, and **customers** is a child of both **system** and **mydb**.



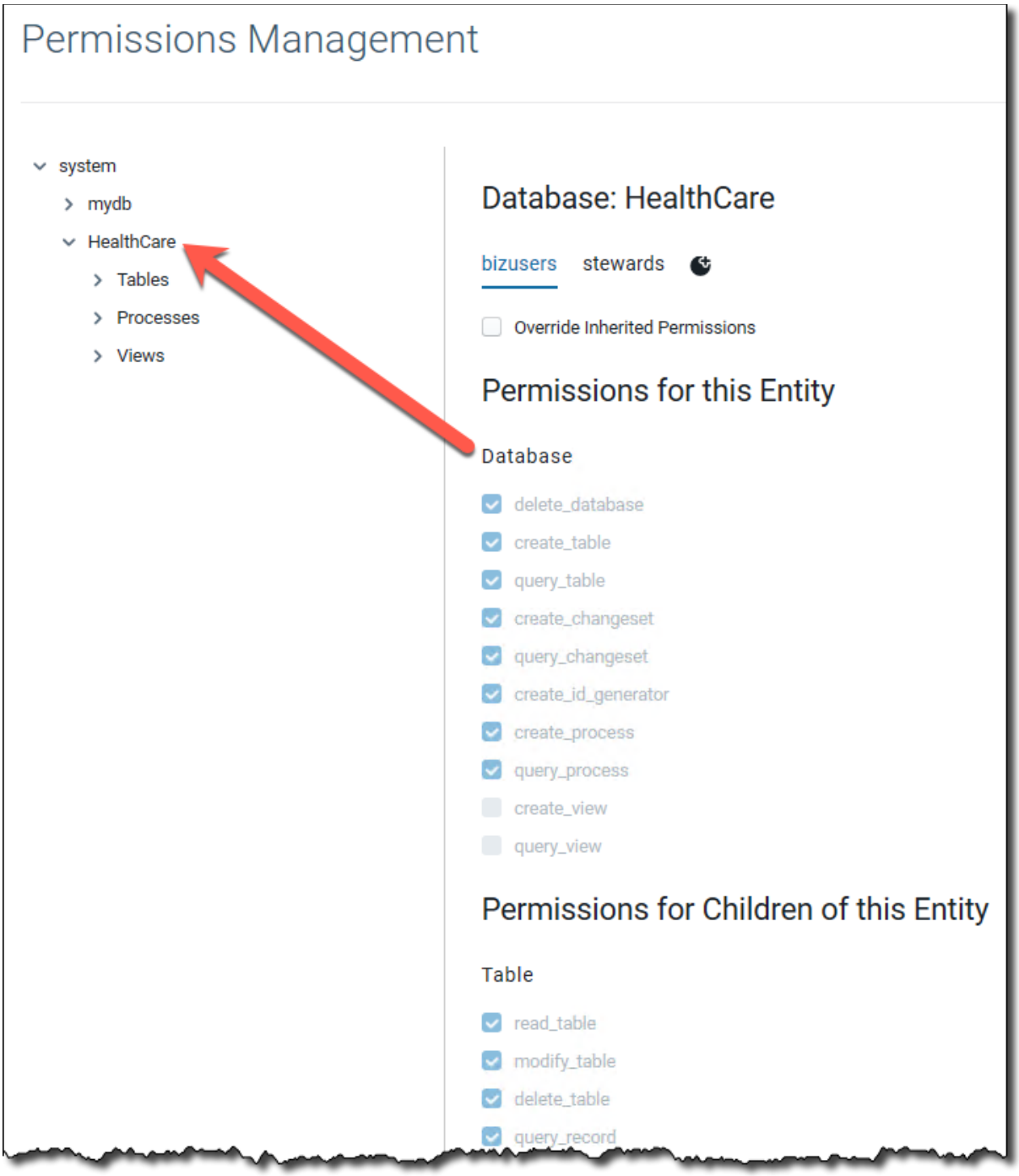
Entity parent-child relationships

Permissions assigned to a group for an entity are inherited by children of the entity, but these inherited permissions may be overridden.

Unless overridden, group permissions assigned for an entity are inherited for the entity's children according to this hierarchy:

- system
  - database
    - table
      - column (*future enhancement*)
    - process
      - workflow
    - view
      - subscription

In the example below, the permissions assigned to the **bizusers** group for the **Database** entity are displayed, as well as the permissions inherited for the child entities.



Bizusers permissions for the HealthCare database

Note that you can add and remove permissions for groups at any level of a parent-child entity tree.

Remember that permissions are assigned on a group basis. For example, let's say there are two groups assigned permissions to work with tables: **bizusers** and **stewards**. We see that the **bizusers** group has the following table permissions:

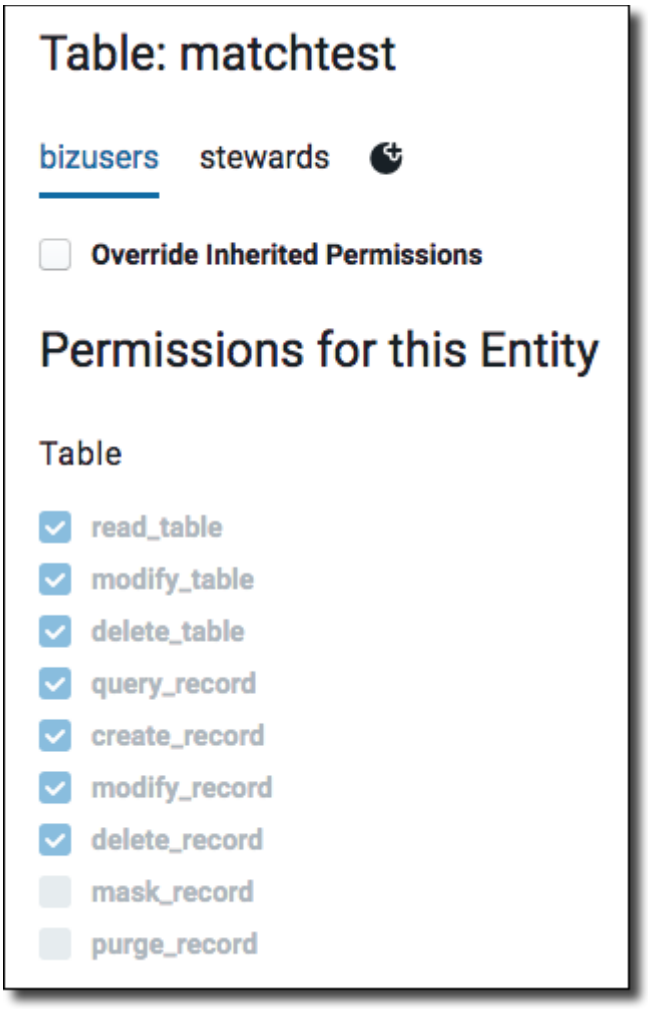


Table permissions for the bizusers group

The **stewards** group has a slightly different permissions list:

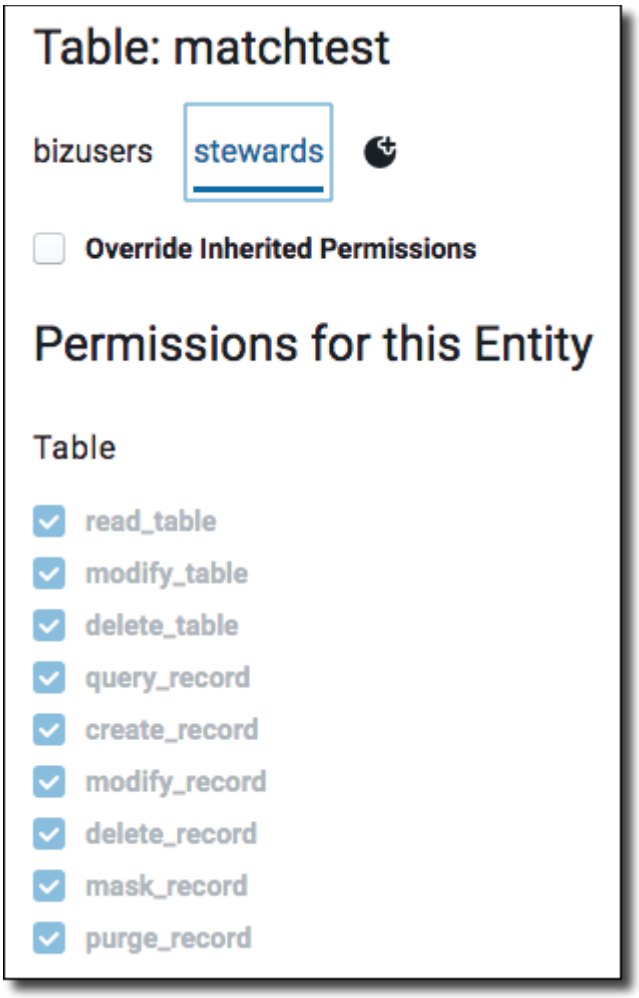


Table permissions for the stewards group

6.8.3.2. HOW PERMISSIONS ARE INTERPRETED FOR EACH ENTITY

Permissions are interpreted for each entity as follows:

Entity Type	Permission	Action allowed
system	create_database	Create a new database.
database	delete_database	Delete a database.
database	create_table	Create a new table.
database	query_table	Query the list of tables in the database.
database	create_changeset	Create a changeset on this database.
database	query_changeset	Query changesets on this database.
database	create_export	Create an export to external RDBMS.

Entity Type	Permission	Action allowed
database	create_id_generator	Create a new ID generator.
database	create_view	Create a new computed view.
database	query_export	Query list of exports.
database	create_process	Create a new process in the database.
database	query_process	Query the list of defined processes.
database	query_view	Query the list of defined computed views.
table	read_table	Read table schema and other details.
table	modify_table	Change the table's schema.
table	delete_table	Delete this table.
table	query_record	Query records in this table. Also grants the right to determine if a record has had its PII data masked or purged.
table	create_record	Create new record in this table.
table	modify_record	Modify records in this table. This combined with create_records allows upsert.
table	delete_record	Delete records from this table.
table	purge_record	Permanently expunge records or record versions from the database.
column	query_column	Include this column in query output (future enhancement).
column	modify_column	Change the value of this column (future enhancement).
database	query_operation	Query the operations performed in a changeset.
database	undo_operation	Undo operations of a changeset.
database	commit_changeset	Commit a changeset.

Entity Type	Permission	Action allowed
database	discard_changeset	Discard a changeset.
database	manage_conflict	Query and resolve conflicts.
process	read_process	Read a process definition.
process	modify_process	Replace an existing process.
process	delete_process	Delete a process.
process	create_workflow	Create a workflow based on this process.
process	query_workflow	Query which workflows are available to start, which workitems are assigned, and so on.
workflow	read_workflow	Read a workflow definition.
workflow	modify_workflow	Modify a workflow definition.
workflow	delete_workflow	Delete the workflow.
view	read_view	Read details of a view.
view	modify_view	Modify a view.
view	delete_view	Delete a view.
view	create_subscription	Create a new subscription to a computed view.
view	query_subscription	Query the subscriptions for a computed view.
subscription	read_subscription	Read details of a subscription.
subscription	modify_subscription	Modify a subscription.
subscription	delete_subscription	Delete a subscription.
database	mask_record	Clear PII fields in a record and note who masks the record and when.
database	purge_record	Remove a record with PII data and note who purged the record and when.

6.8.3.3. SYSTEM-ONLY PERMISSIONS

There are some things that can only be done by users in the **system** group, so there is no explicit permission type for them:

- Manage users
- Change permissions of the **system** object

6.8.3.4. SYNCHRONIZATION AND VERSIONS

There is no versioning of permissions—they apply to all versions of data and metadata simultaneously. When a database is deleted, everything is deleted (including all history) and permissions for the database (and all child entities of the database) are removed with it.

6.8.3.5. PERMISSIONS AND GROUPS

The following topics relate to permissions and groups:

6.8.3.5.1. PERMISSION ENTRIES

A permission entry is a set of permissions for an entity and all inheriting descendants. Because of this, it is useful to think of setting permissions on an entity as actually setting the default permissions for an entire *entity tree*. An entity tree consists of the entity and all of its current and future descendants. The root of every entity tree is the **system** entity. Some things to note about permission entries:

- Each permission entry is for a single group.
- Each permission entry applies the given permissions to that entity and all descendants.
- Each permission entry must specify all desired permissions for the given entity and its child entities. Because of this requirement, permission overrides are an all-or-nothing proposition: if any permission is changed from its inherited default, all permissions must be specified.
- Permissions that aren't applicable to the given entity type are ignored (but may be passed on to child entities). For example, when setting permissions on a database, you will also want to specify the default permissions for every table, process, workflow, view, and subscription.
- Inherited permissions are taken from the closest ancestor in the permissions tree.

Typically, the **system** entity has a set of permission entries (one for each group) to establish the default permissions for each group. You can gain finer-grained control by overriding permissions for a particular group and entity combination. Note that deleting a permission entry does *not* remove any descendant permission entries.

In another example, suppose that database **hospitalRecords** has a permission entry containing **modify_record** for group **stewards**. Given that setting, all table records in that database can be modified by users in the **stewards** group.
However, if the administrator wants to limit access to certain tables (for example, **hospitalRecords.no_touch**), she may override permissions for the **no_touch** table, and remove the **modify_record** permission, thus preventing stewards from being able to modify that specific table.

6.8.3.5.2. MOST-PERMISSIVE LOGIC

There may be multiple permission entries for an entity that apply to the user attempting to perform an operation because a user may belong to multiple groups. If this is the case, the *effective* set of permissions for a user on an entity consists of the union of the permissions on that entity for all of the user's groups.

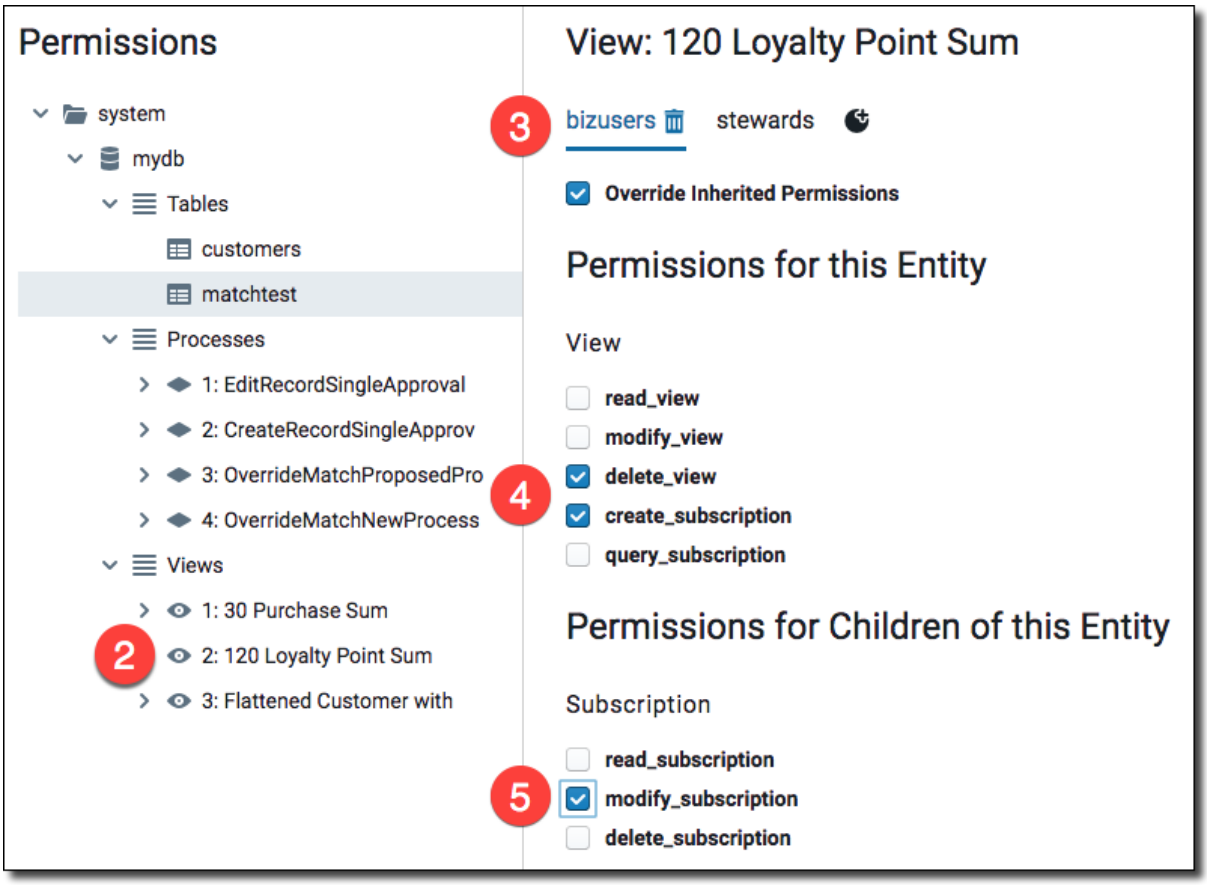
6.8.3.6. CHANGE ENTITY PERMISSIONS

By default, child entities inherit the permissions of their parent entity. For example, **system** entity permissions are inherited by all database entities. Database permissions are inherited by their table entities, and so on down the hierarchy.

If you don't want to use inherited permissions for a portion of the permissions tree, you can override an entity's permissions.

- Children of the overridden entity will inherit the new permissions unless they also override permissions.
- Overrides are done per-group. Therefore, some groups may see overridden permissions and others may not.
- A user that does not belong to a group with sufficient privileges to modify permissions will still be able to view permissions but not change or override them.

To change permissions on an entity



The process of changing permissions for an entity (steps 2 through 5 below)

1. In the **Administration** section, click **Permissions**.
2. Select the entity on the tree for which you want to change inherited permissions.
3. Select a group.
4. Select or clear entity permissions. (If the permissions are not selectable, select **Override Inherited Permissions**.)

WARNING! Once you override inherited permissions on an entity, you cannot switch back to inherited permissions. Your only option at this point is to destroy the entire group for that entity.

5. Select or clear permissions on the entity's children.

6.8.3.7. ADDING AND DELETING PERMISSION ENTRY GROUPS

Adding or deleting a permission entry group for an entity means that you are granting or revoking a group's permissions on an entity (and possibly its children).

For example, if you grant permissions for the **officers** group to a database entity, you are explicitly granting any user who is a member of the **officers** group officer-level permissions for the database entity and its children (because of inheritance). If you decide that **officers** should be granted different permissions for certain tables of that database, you could go to a child table of that database entity and delete the **officers** group permissions entry. This would revoke officer-specific permissions from the table.

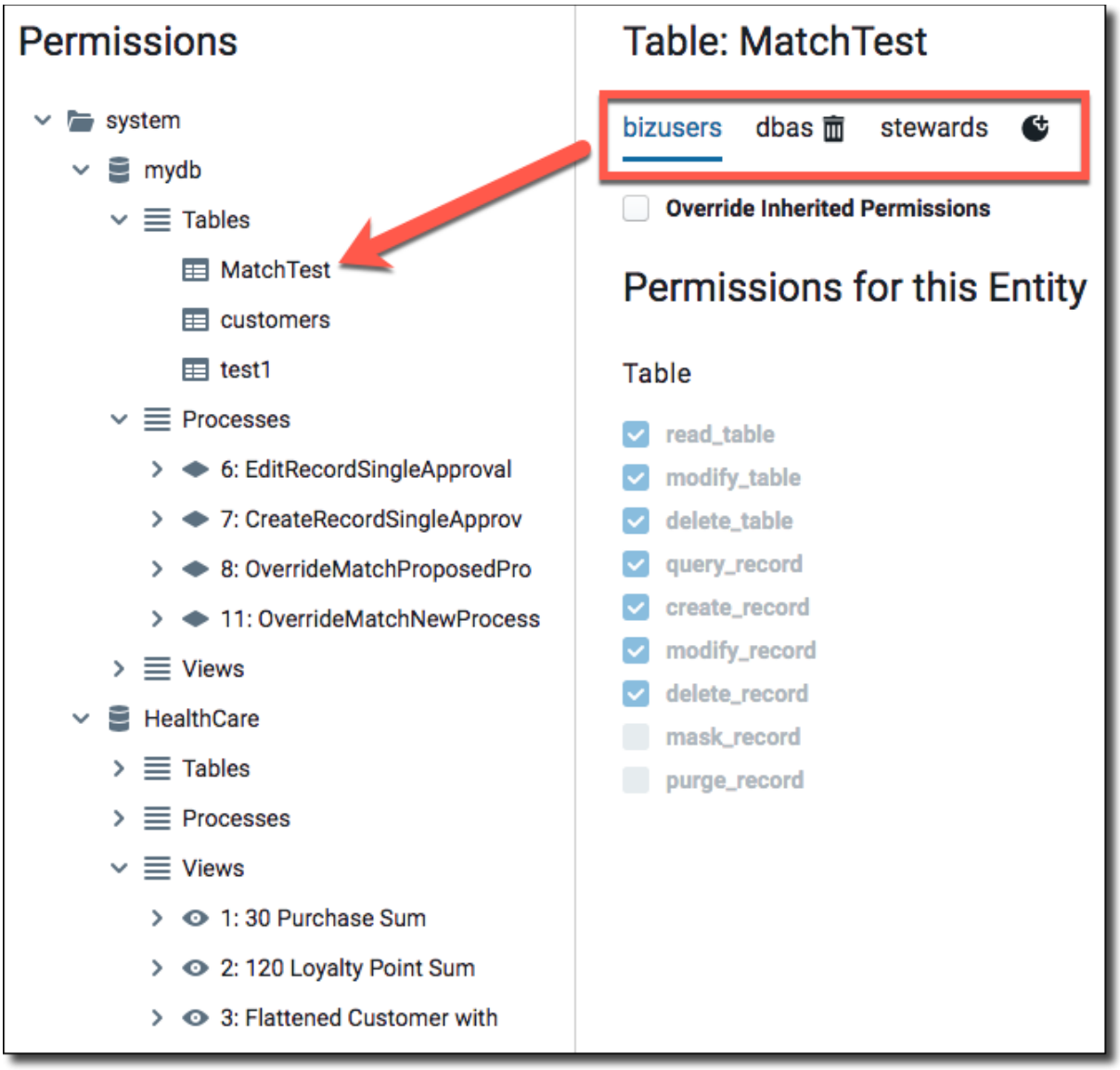
To add a permission entry for a group

Note that you can only add groups that have been previously defined for your MDM site workflows.

1. In the **Administration** section, click **Permissions**.
2. Select an entity in the tree and click the group add button  .
3. On the **Create Permission Group** pop up, select a group name and click **Create**.

To delete a permission entry for a group

1. In the **Administration** section, click **Permissions**.
2. Select an entity in the tree:



Deleting the bizuser's permission entry for the MatchTest table

3. On the row of groups to the right of the entity tree, click the group's associated delete icon. (If the group doesn't have an associated delete icon, select **Override Inherited Permissions**.
4. On the **Delete Group** pop up, click **Delete**.

6.9. MATCHING SUB-DOCUMENT

Tables created with **matching** set in the schema will create documents with an optional **rpmdm_match** sub-document. This sub-document contains the entirety of the matching state for a record, including:

- Automatic matching performed inside MDM
- Matching performed externally by Redpoint Data Management
- Match overrides performed by stewards
- Match groups

6.9.1. THE SUB-DOCUMENT

The matching sub-document takes the following form:

```
"rpmdm_match":{

   type:{
      "type":"string",
      "group":{"id":long, "score":float},
      "auto":[
         {"other":"value", "score":float}
         ...
      ],
      "always":["value",...],
      "never":["value",...]
   },
   ...
}
```

Each sub-document in the **rpmdm_match** document represents a separate *match type*.

The tags inside each element are:

Tag	Meaning
type	The match type. This is used as the key into the map and is repeated within the match sub-doc. The default value is "default" (if nothing else is specified).
group	The match group to which this record has been assigned, if any: • **id** is the 8-byte integer match group ID. Match group IDs are automatically generated and have no relation to record primary keys. • **score** is a number between 0.0 and 1.0 (inclusive) and indicates the quality of the match group. This score is synthesized from a combination of the inter-record match scores and a measurement of the match group inter-connection strength.
auto	An array of matches to other records that were made through automated matching. Each entry contains: • **other**: The primary key of the other record. This should not contain any primary keys contained in the always or never arrays. • **score**: The match score produced by the matching system. The value is between 0.0 and 1.0, and indicates the quality of the match. This score has no direct interpretation, although it is generally related to the edit distance between this record and other

Tag	Meaning
	records of the group, and higher scores are better matches.
always	An array of primary keys indicating "always match" to other records. These are the result of manual force-match operations. This record will be grouped with all other records in the **always** list unless: • Doing so would place this record in a group with one or more records from the **never** list, or • The resulting group would break limits on group size or topology (see the specification of grouping rules for details).
never	An array of primary keys indicating "never match" to other records. These are the result of manual break/split operations. This record will never be added to a group containing another record in this list. **Note**: **never** takes priority over **always**.

Note that **auto**, **always**, and **never** are symmetric (both records in the pairs should contain the same information).

6.10. AGGREGATION LANGUAGE

MDM's views support *aggregation projections* that can perform complex calculations on groups of records, summarizing each group down to a single output record. You specify the aggregation rules using a JSON document in which each computation is an expression loosely based on Java. This aggregation language is not simple, but it is extremely powerful.

6.10.1. EXPRESSION SYNTAX

Aggregating expressions are used within the aggregation projections of computed views. An aggregating expression is a declaration of the ultimate value of an output variable, *not* code that performs iterative operations as input variables are processed. An aggregating expression may reference input variables only through *aggregators*. It may also reference constants and Java methods (for example, Math.abs) and operations (for example, + , -, *, /).

In addition to aggregating expressions, there are expressions used as the terms for aggregators. These are simple expressions over input and output variables, without the aggregators.

6.10.2. SPECIAL IDENTIFIERS

A MDM aggregating expression is really a Java expression. We evaluate these expressions by transforming them into Java classes, compiling them, and running the compiled code. MDM uses a simple

technique to distinguish between the things that MDM cares about, and the rest of the expression that is passed through to Java. To accomplish this, the aggregating language prefixes MDM identifiers with `$`:

- Input record field names
- Output record field names
- Temporary variable names
- Computed field names
- Selectors and aggregators
- MDM-provided functions

For example, in the following expression:

```
$Sum($Filter($TYPE==3), Math.abs($PURCH_QTY) / 2)
```

MDM recognizes these elements:

- `$Sum` is an aggregator.
- `$Filter` is a selector.
- `$TYPE` is an input field.
- `$PURCH_QTY` is an input field.

By contrast, `Math` and `abs` are not recognized or processed by MDM, but are passed through to Java. The aggregation library allows only a small documented set of Java imports such as Math and Joda types. Internally, MDM generates Java code and compiles it into a class that carries out the computations.

All `$` identifiers are case-insensitive and treated as lower case regardless of how they are input. For example, the following entities are treated as identical:

```
$FILTER
$filter
$FiLtEr
```

6.10.3. BACKQUOTED STRINGS

To make it easier for users to use double quotes in expressions, backquotes are treated as double quotes inside expression strings. For example, within an expression string, `` `hello` `` is the same as `\"hello\"`. Because we parse these expressions, to include a backquote in a backquoted string, you need to escape the backquote that will still be converted into a double quote. To create the backquote character, use the unicode escape sequence `\u0060` instead.

6.10.4. EXPRESSION TYPES

MDM's aggregation language uses two expression types: *aggregating expressions*, and *scalar expressions*.

Aggregating expressions

An aggregating expression is an *expression over aggregators*. It doesn't compute directly on field values, only on aggregated field values. All assignments to output fields or temporary variables are aggregating expressions. Aggregating expressions operate implicitly on "the group of records," but the group of records was defined in the computed view itself (such as "all transactions for a customer ID"). For example:

```
$Sum($Filter($TYPE==3), Math.abs($PURCH_QTY)) / 2.0
```

The above expression means "Find all records in the group where TYPE==3, total absolute value of the PURCH_QTY field, and divide the result by 2.0." Note that this expression has only three elements as far as Java is concerned:

- `$Sum(...)` is an aggregator.
- `/` is a Java divide operator.
- `2.0` is a Java floating-point constant.

Of course, *inside* the `$Sum(...)` aggregator are other items that are not themselves aggregators, as discussed below.

Scalar expressions

A scalar expression computes over record variables. It can be a Boolean condition such as:

```
$PURCH_QTY > 0
```

It can be a scalar value such as:

```
$PURCH_AMOUNT / 100
```

It can involve Java language elements, such as:

```
Math.sqrt(Math.abs($PURCH_AMT)) + $FIRSTNAME.length()
```

It can be arbitrarily complex. However, it can only access the fields of one record at a time. Which record it is accessing is controlled by the aggregating machinery and is not the concern of the expression. Scalar expressions are used in several places:

- The *expression* argument to `Sum()`, `Min()`, and `Max()`.
- The *condition* argument to `Count()`.
- The *condition* in `Filter()`.

6.10.4.1. SCALAR VERSUS AGGREGATING USAGE

The two expression types (*aggregating expressions*, and *scalar expressions*) must be used in the appropriate context:

- Selectors consume scalar expressions.
- Aggregators consume scalar expressions.
- An output field must be assigned an aggregating expression.
- A temporary variable must be assigned an aggregating expression.
- A computed field must be assigned a scalar expression.
- An input field referenced as $FIELD can only be used in scalar expressions.

- A computed field referenced as $FIELD can only be used in scalar expressions.
- A temporary variable referenced as $VARIABLE can be used in scalar or aggregating expressions.

For example, if you assign a temporary variable:

```
$NumRecordsInGroup = $Count($All())
```

You can later say:

```
$Filter(condition).$Sum($PURCHASE) / $NumRecordsInGroup
```

or:

```
$Filter(condition).$Sum($PURCHASE / $NumRecordsInGroup)
```

Similarly, if you assign a computed field:

```
$Rank = $SOURCE.equals("A") ? 1 : $SOURCE.equals("Z") ? 2 : 3
```

You can later say:

```
$First($Filter(condition).$Sort($Rank, descending), $VALUE)
```

But you cannot say:

```
$Filter(condition).$Sum($PURCHASE) / $Rank
```

6.10.5. AGGREGATORS

An aggregator is a function that takes an *input set* and an expression:

Function(selector, expression)

The selector is represented as a chain of *selectors*, which taken together specify an ordered set of input records to process. The expression can be any Java expression over input fields and computed fields. For example, this expression counts the number of records where PROGRAM_ID is equal to 3:

```
$Count($Filter($PROGRAM_ID==3))
```

This expression computes the average value of TRANSACTION over records where TYPE is equal to 10:

```
$Average($Filter($TYPE==10),$TRANSACTION)
```

The following aggregators are supported:

- $Sum with optional type
- $Min and $Max and $Min and $Max with type
- $Count
- $Average
- $First and $First with type
- $Nth and $Nth with type

6.10.5.1. $SUM WITH OPTIONAL TYPE

The sum of the given *scalar-expression* over all records in *selector*.

Syntax

$Sum(*selector*, *scalar-expression* [, *type*])

Remarks

MDM supports BigDecimal, long, double, and String types. Summing dates and times is not currently supported. If no type is specified, $Sum() returns a value of type double.

The sum of a string is the concatenation of the all of the string values. For example, $Sum(*..., expression returning* ["a", "b", "c"]) produces the string `abc`.

The sum of an empty set is zero for numbers and the empty string ("") for strings.

6.10.5.2. $MIN AND $MAX

The minimum or maximum value of the given *scalar-expression* over all records in *selector.*

Syntax

$Min(*selector*, *scalar-expression*)

$Max(*selector*, *scalar-expression*)

Remarks

Returns a value of double. The minimum or maximum value of an empty set is zero (`0.0`).

6.10.5.3. $MIN AND $MAX WITH TYPE

The minimum or maximum value of the given *scalar-expression* over all records in *selector.*

Syntax

$Min(*selector*, *scalar-expression, type*)

$Max(*selector*, *scalar-expression, type*)

Remarks

Returns a value whose type is determined by the *type* argument. Choose one of the following keywords:

- BigDecimal
- Boolean
- Double
- Long
- LocalDate
- LocalDateTime
- LocalTime
- String

The *type* keywords are case-insensitive but are replaced with the case-sensitive version listed above. All of the date types are Joda types. The minimum or maximum value of an empty set depends on the type:

Type	Default value
BigDecimal	0
Boolean	false
Double	0.0
Long	0L
LocalDate	1600-01-01
LocalDateTime	1600-01-01 00:00:00
LocalTime	00:00:00 (Midnight)
String	""

Why have a type?

There are times you will be working with a data type other than double. Because these functions are used in arbitrary scalar expressions and our compiler is not part of the Java compiler, we cannot determine the correct type at compile time and instead require a hint from the user. This only matters if a default value is required. For the case of $Min and $Max, this is for the empty set.

6.10.5.4. $COUNT

The number of records in selector, returning a value of type long.

Syntax

$Count(*selector*)

6.10.5.5. $AVERAGE

The average of the given *scalar-expression* over all records in *selector*.

Syntax

$Average(*selector*, *scalar-expression*)

Remarks

Returns a value of type double. The average of the empty set is zero.

6.10.5.6. $FIRST

Calculates *scalar-expression* for the first record in *selector*.

Syntax

$First(*selector*, *scalar-expression*)

Remarks

This is typically done using $Sort() as the selector to choose the "best value of the group" based on the ranking established in $Sort.

Do not confuse this with the $Top(1) selector. While they seem similar, they have completely different purposes:

- $First: This aggregator evaluates its expression for the first record in the group, and returns it.
- $Top(1): This selector reduces the record group to a single record and makes it available to aggregators.

If the selector returns the empty set, $First will return the double value 0.0.

6.10.5.7. $FIRST WITH TYPE

Calculates *scalar-expression* for the first record in *selector*.

Syntax

$First(*selector*, *scalar-expression*, *type*)

Use this version of **$First** if your scalar expression will return a value with a type other than double. Returns a value whose type is determined by the *type* argument. Choose one of the following keywords:

- BigDecimal
- Boolean
- Double
- Long
- LocalDate
- LocalDateTime
- LocalTime
- String

The *type* keywords are case-insensitive but are replaced with the case-sensitive version listed above. All of the date types are Joda types. If the *selector* returns the empty set and a default value is needed, the *type* argument determines that default value as follows.

Type	Default value
BigDecimal	0
Boolean	false
Double	0.0
Long	0L
LocalDate	1600-01-01
LocalDateTime	1600-01-01 00:00:00
LocalTime	00:00:00 (Midnight)
String	""

6.10.5.8. $NTH

Calculates *scalar-expression* for the nth record in *selector*.

Syntax

$Nth(*selector*, *offset*, *scalar-expression*)

Remarks

The offset argument must be a constant integer value. Zero is not a valid offset.

If *offset* is positive, it is the one-based offset of element starting at the front of the record set. For example, 1 gets the first element, 4 gets the 4th element, and so on.

If *offset* is negative it is the one-based offset of element starting at the back of the record set. For example, -1 gets the last element, -2 gets the first from the last element, -3 the second from the last element, and so on.

The default return value is of type double.

6.10.5.9. $NTH WITH TYPE

Calculates *scalar-expression* for the nth record in *selector*.

Syntax

$Nth(*selector*, *offset*, *scalar-expression, type*)

Use this version of **$Nth** if your scalar expression will return a value with a type other than double. Returns a value whose type is determined by the *type* argument. Choose one of the following keywords:

- BigDecimal
- Boolean
- Double
- Long
- LocalDate
- LocalDateTime
- LocalTime
- String

The *type* keywords are case-insensitive but are replaced with the case-sensitive version listed above. All of the date types are Joda types. If the *selector* returns the empty set and a default value is needed, the *type* argument determines that default value as follows.

Type	Default value
BigDecimal	0
Boolean	false
Double	0.0

Type	Default value
Long	0L
LocalDate	1600-01-01
LocalDateTime	1600-01-01 00:00:00
LocalTime	00:00:00 (Midnight)
String	""

6.10.6. SELECTORS

Selectors are pseudo-functions that can be chained together to specify the *selector* upon which the aggregators are evaluated in an expression.

Selectors are chained together with a period character `.` and applied left-to-right. For example:

```
$Filter($TYPE==4).$Sort($DATE,descending).$Top(5)
```

selects the most recent five records where TYPE==4.

Selectors are automatically cached and shared within a set of aggregation rules. If the same chain of selectors is used repeatedly, code is generated automatically to avoid computing the set more than once per group.

The following selectors are supported:

- $All
- $Filter
- $AgeDays
- $Sort
- $Top
- $Unique
- $MostCommon

6.10.6.1. $ALL

Designates all input records of a record group.

Syntax

$All

6.10.6.2. $FILTER

Filters the record set, retaining only records for which *condition-expression* evaluates to true.

Syntax

$Filter(*condition-expression*)

Remarks

The *condition-expression* must evaluate to true/false.

6.10.6.3. $AGEDAYS

Selects records whose *$FIELD* is between *low* and *high* days old.

Syntax

**$AgeDays**(*$FIELD*, *low*, *high*)

Remarks

The $AgeDays selector is a special kind of filter, selecting records whose *$FIELD* (which must be Date or DateTime type) is between *low* and *high* days old (inclusive) relative to the current date. The low and high values are literal integers. Use of this filter triggers automatic re-computation of affected aggregations during a specified "middle of the night" window as records age in and out of the day range. For example:

```
$AgeDays($PURCHASE_DATE,0,30)
```

selects records whose PURCHASE_DATE is between 0 and 30 days old.

To express "up to 30 days old," you should use `$AgeDays(*$FIELD*,0,30)`, not `$AgeDays(*$FIELD*,1,30)`. One day old means "yesterday" whereas zero days old means "today."

The time is never considered when performing the age calculation; it is stripped from both the current date/time and the field value. Only the date portion is considered. For example:

- If the current time is `2020-01-01T23:59:59` and the date field value is `2020-01-02T00:00:00`, the field is "one day old."
- If the current time is `2020-01-01T00:00:00` and the date field value is `2020-01-01T23:59:59`, the field is "zero days old."

6.10.6.4. $SORT

Sorts the record set by a list of input field names and key orders.

Syntax

$Sort(*field1*, *order1* [, *field2*, *order2…*])

Remarks

Each *order* is the keyword either *ascending* or *descending*.

6.10.6.5. $TOP

Retains only the first N records of a record set.

Syntax

$Top(*N*)

Remarks

N must be a constant integer >= 1. For example:

```
$Sort($DATE,descending).$Top(3)
```

selects the three newest records.

6.10.6.6. $UNIQUE

Reduces the record set by discarding records that are duplicated across the input variables *uniqueVar1 … uniqueVarN*.

Syntax

$Unique(*uniqueVar1* [, *uniqueVar2…*], [*tieBreakField1*, *tieBreakOrder1,…*])

Remarks

If additional consideration must be given to which records survive deduplication, a list of "tie breakers" may also be specified, each of which contains an input variable and an order. Note that ascending order keeps the lowest-valued records, and descending order keeps the highest-valued records. For example:

```
$Unique($SOURCE,$DATEOFBIRTH,descending)
```

Will select a group of records that are unique by the SOURCE field, breaking ties by choosing the most recent DATEOFBIRTH.

6.10.6.7. $MOSTCOMMON

Reduces the record set by selecting only records that have the most common value or values.

Syntax

$MostCommon(*field* [, *N*])

Remarks

The $MostCommon selector is limited to using the first `${rpmdm.subscription.max_delta_size}` records. If an aggregation has more than this number of records, only the first delta size (default 100,000) records are considered.

If *N* is not specified, it defaults to one. If *N* is specified using a positive integer value, records which have the *N* most common values are selected. *N* should be small, but is not constrained. The selector creates a histogram of the field value of all input records and then chooses the most commonly-occurring value(s) of that field.

- In case of ties, the record order is used as a tie breaker.
- The order of the input record is not preserved. The result is ordered from most common to least common. For a given value of the field, the records are left in the order found. For example, given values `a, b, c, d, c, d, b, c` the records are reordered to `c, c, c, b, b, d, d, a`. If *N* is less than 4, the result is `ccc`, `cccbb`, or `cccbbdd`, corresponding to values of N=1,2,3 respectively.
- If *N* is larger than the number of distinct values for the field, the input is reproduced but sorted as described above.
- The algorithm runs in O(*n*) or O(*n*) + O(*log n*) if *n* is small, where *n* is the number of input records.

For example, given this input:

IndividualID	XactID	Qty	Amount	ProductID
100	1	1	17.45	535
100	2	2	1.56	432
100	3	1	463.00	535
100	4	4	34.17	145
100	5	1	100.00	532

running the selector

```
$MostCommon($ProductID)
```

records are selected containing `ProductID=535` (occurs twice vs all the other values) which:

IndividualID	XactID	Qty	Amount	ProductID
100	1	1	17.45	535
100	3	1	463.00	535

`$MostCommon($ProductID, 2)` would select the following records in the following order:

IndividualID	XactID	Qty	Amount	ProductID
100	1	1	17.45	535
100	3	1	463.00	535
100	2	2	1.56	432

6.10.7. TEMPORARY VARIABLES

A *temporary variable* is a scalar value that you can compute once and reuse later. This is helpful if you find yourself typing the same sub-expression multiple times. Some rules of thumb to remember:

- Temporary variables (like output fields) are an aggregation result. They are not computed per-record (see "computed fields" below for per-record computations).
- Temporary variable calculations are done before any output field aggregations.
- Temporary variable calculations are done in order. If there are any dependencies, put them in the right order!

Examples

```
$Last30Count = $Count( $AgeDays($PURCH_DATE, 0,30) )$outputField = $Last30Count >
12$anotherOutput = $Last30Count <
```

6.10.8. COMPUTED FIELDS

A *computed field* acts like another field that is appended to each input record, except that its value is derived from the other input fields using a scalar expression. For example, to compute a custom Rank

field on each record that you can use later in $Sort, you might try a conditional expression chain in Java:

```
$Rank = $SOURCE.equals("A") ? 1 : $SOURCE.equals("B") && $PURCHASE > 1000 ? 2 : 3
```

Refer to computed fields in later scalar expressions just like you would any other input field (using $VARIABLE).

6.10.9. JAVA LIBRARY SUPPORT

Only the following Java and Joda objects are supported:

- java.lang.Math
- java.lang.StrictMath
- java.math.BigDecimal
- java.util
    - java.util.ArrayList
    - java.util.HashSet (although these classes are needed by the aggregation compiler and would be unusual to use in an expression)
- org.joda.time
    - LocalDate
    - LocalDateTime
    - LocalTime

6.10.10. SUPPORT FUNCTIONS FOR EXPRESSIONS

MDM's aggregation language offers the following built-in functions that simplify some Java constructs to aid in string comparison and the coercion of scalar expressions to a particular type:

- $Compare(v1, v2)
- $Equals(v1, v2)
- $StrCompare(s1, s2)
- $StrEquals(s1, s2)
- $DaysDiff(dt1, dt2)
- $ToBigDecimal
- $ToBoolean
- $ToDate
- $ToDateTime
- $ToDouble
- $ToFloat
- $ToInteger

- $ToLong
- $ToString
- $ToTime

6.10.10.1. $COMPARE

Compares the expressions *v1* and *v2*.

Syntax

$Compare(*v1*, *v2*)

Remarks

Compares the expressions *v1* and *v2*, promoting variables to the "biggest type" (for example, in int versus long, the int is cast to long) and returns:

- a negative integer if *v1* < *v2*
- 0 if *v1* == *v2*
- a positive integer if *v1* > *v2*

Strings use a case-sensitive comparison. See $StrCompare.

6.10.10.2. $EQUALS

Compares two variables for equal value.

Syntax

$Equals(*v1*, *v2*)

Remarks

Compares two variables and returns *true* if they are equal and *false* otherwise. Strings use a case-sensitive comparison. See $StrCompare for case-insensitive comparison. When used on values with differing types the variables with "smaller" types are promoted to larger types. For example, int becomes long, Date compared to DateTime will be promoted to DateTime, and so on.

6.10.10.3. $STRCOMPARE

Compares the string expressions *s1* and *s2*.

Syntax

$StrCompare(*s1*, *s2*)

Compares the string expressions *s1* and *s2* (ignoring case) and returns:

- a negative integer if *s1* lexically precedes *s2*
- 0 if *s1* ==*s2*
- a positive integer if *s1* lexically follows *s2*

For example, `$StrCompare("FooBIE!", "foobie!")` returns 0.

If *s1* or *s2* is not a string, it is converted to string before comparing.

6.10.10.4. $STREQUALS

Compares two string expressions.

Syntax

$Equals(*v1*, *v2*)

Remarks

$StrEquals(*s1*, *s2*)

Compares the string expressions *s1* and *s2* (ignoring case) and returns *true* if they are lexically equal and *false* otherwise. For example, `$StrEquals("FooBIE!", "foobie!")` returns *true*. If *s1* or *s2* is not a string, it is converted to string before comparing.

6.10.10.5. $DAYSDIFF

Returns the difference in days between *dt1* and *dt2*.

Syntax

$DaysDiff(*dt1*, *dt2*)

Remarks

The arguments *dt1* and *dt2* must be LocalDate or LocalDateTime instances. The returned value is of type integer. This number is positive if *dt2* is later than *dt1*, and negative if *dt2* is earlier than *dt1*.

For example, if *dt1* is 2000-01-01 and *dt2* is 2000-01-02:

```
$DayDiff( $JAN_01_2000, $JAN_02_2000)
```

returns 1.

6.10.10.6. $TOBIGDECIMAL

Converts a scalar value to type BigDecimal.

Syntax

$ToBigDecimal( )

Remarks

For numeric types, this is a one-to-one map. For non-numeric types, the value is first converted to a string and then to a BigDecimal. A run-time exception may occur if the input is not a number or its $ToString value is not a number.

6.10.10.7. $TOBOOLEAN

Converts a scalar value to Boolean.

Syntax

$ToBoolean( )

Remarks

Numeric values of `0` (zero) return false. Numeric values other than zero return true. Non-empty strings return true (for example, `$toBoolean("false")` is true). Non-numeric, non-string instances return *false* if their $ToString value is an empty string and *true* otherwise.

6.10.10.8. $TODATE

Convert a LocalDate, LocalDateTime, or String instance to a LocalDate instance.

Syntax

$ToDate( )

Remarks

Converting strings with invalid date formats or types other than LocalDate and LocalDateTime will cause run-time errors.

6.10.10.9. $TODATETIME

Convert a LocalDate, LocalDateTime, or String instance to a LocalDateTime instance.

Syntax

$ToDate( )

Remarks

Converting strings with invalid date formats or types other than LocalDate and LocalDateTime will cause run-time errors.

6.10.10.10. $TODOUBLE

Convert a Numeric, Boolean, or String value to a Double.

Syntax

$ToDouble( )

Remarks

Numeric values are cast to double. Boolean becomes 1.0/0.0 for true/false. String values are parsed. Invalid strings are treated as 0.0.

All other classes (for example, Dates) cause run-time errors.

6.10.10.11. $TOFLOAT

Convert a Numeric, Boolean, or String value to a Float.

Syntax

$ToFloat( )

Remarks

Numeric values are cast to 32-bit float. Boolean becomes 1.0/0.0 for true/false. String values are parsed. Invalid strings are treated as 0.0.

All other classes (for example, Dates) cause run-time errors.

6.10.10.12. $TOINTEGER

Convert a Numeric, Float, Boolean, or String value to an Integer.

Syntax

$ToInteger( )

Remarks

Numeric values are cast to integer. Floating point values are rounded as per java.lang.Math.round(). (For example, `$ToInteger(0.5)` is 1.) Boolean becomes 1 / 0 for true / false. String values are parsed. Invalid strings are treated as zero.

All other classes (for example, Dates) cause run-time errors.

6.10.10.13. $TOLONG

Convert a Numeric, Float, Boolean, or String value to a Long.

Syntax

$ToLong( )

Remarks

Numeric values are cast to long integer. Floating point values are rounded as per java.lang.Math.round(). For example, `$ToInteger(0.5)` is 1. Boolean becomes 1 / 0 for true / false. String values are parsed. Invalid strings are treated as zero.

All other classes (for example, Dates) cause run-time errors.

6.10.10.14. $TOSTRING

Converts the value to a String.

Syntax

$ToString( )

Remarks

Runs the .toString() method on the instance.

6.10.10.15. $TOTIME

Converts a LocalTime, LocalDate, LocalDateTime, or String instance to a LocalTime instance.

Syntax

$ToTime( )

Remarks

Convert a LocalTime, LocalDate, LocalDateTime, or String instance to a LocalTime instance as per the Joda cast operation. Converting values of other types causes a run-time exception.

6.10.11. AGGREGATION EXPRESSION NOTES

The following topics offer more in-depth information on aggregation expressions:

- Rules and conventions
- Java language considerations

- MDM type to Java type mappings
- Null values and Date/DateTime fields
- Selecting multiple fields as a group
- Multiple aggregators in an expression
- Making decisions
- Complex ranking

6.10.11.1. RULES AND CONVENTIONS

When thinking about expressions, observe the following rules and conventions:

- Expressions have a passive role. They do not "loop" over records; they are handed records one at a time, or handed groups of records for aggregation.
- Expressions are declarative. Thus, they should not have any side effects (opening files, writing to the Web App, changing global variables, and so on). MDM restricts access to most things that could have side effects.
- Numeric variables that are null on input are given the value zero (`0`).
- Date fields that are null on input are given start of epoch default value 1600-01-01.
- Time fields that are null on input are given the default value 00:00:00.000.
- DateTime fields that are null on input are given the default value 1600-01-01T00:00:00.000.
- String fields that are null on input are given the value "" (empty string).
- Because nulls are mapped to default values, special variables of the form $isXXXValid (where XXX is the input field name) are automatically generated so that rules can test for null explicitly.
- Java comma expressions (statement, statement) are not supported.

6.10.11.2. JAVA LANGUAGE CONSIDERATIONS

Because expressions are essentially Java with placeholders for the special aggregating functionality, you can use most language constructs that you can use in Java itself. In addition, fields in records become data fields in Java classes that have correct semantics and rich types. Specifically:

- Mathematical operators: `+ - * / %`
- Bitwise operators: `<< >> >>> & | ~ ^`
- Logical operators: `&& || ! == != < > >= <=`
- The conditional expression: `condition ? expr1 : expr2`
- Parenthetical grouping: `( )`
- Bracket operators: `[ ]`
- The functions of the Math and StrictMath packages
- Methods of LocalDate, LocalTime, and LocalDateTime fields. Note, for example, that `LocalDate.now()` returns the current date.
- Do not compare strings using `=` or `==`. Use $Equals or $StrEquals.

- Do not compare strings using `<` or `>`. Use $Compare or $StrCompare.
- Aggregation language does not support comma expressions. For example, `while( x >>>= 1, x == 0)` is unsupported.
- A backquote is treated as a double quote. Use `\u0060` to create the ` character in a string.

6.10.11.3. MDM TYPE TO JAVA TYPE MAPPINGS

MDM type	Java type	Description
all integer types (8, 16, 32, 64, signed and unsigned)	long (64 bits)	64 bit unsigned is not allowed. All integer types are coerced to long
all floating point types (32, 64)	double	
text (string)	java.lang.String	
boolean	boolean	true, false (no null)
date	org.joda.time.LocalDate	date with no timezone
time	org.joda.time.LocalTime	
datetime	org.joda.time.LocalDateTime	date/time with no timezone
binary	byte[]	This is not useful for aggrega-tions, in general
money	BigDecimal	Roughly equivalent to a decimal with precision=20 and scale=4

6.10.11.4. NULL VALUES AND DATE/DATETIME FIELDS

Newest by Date/DateTime

Because null Date/DateTime fields are replaced by "epoch" dates at 1600-01-01, finding the newest records generally works as expected without special consideration for null. For example:

```
$Sort($DATEOFBIRTH,descending).$Top(3)
```

selects the three newest records.

Oldest by Date/DateTime

Because null Date/DateTime fields are replaced by "epoch" dates at 1600-01-01, finding the oldest records needs extra filtering if nulls may be present in the data and you don't want to select them. For example:

```
$Filter(!$isDATEOFBIRTHNull).Sort($DATEOFBIRTH,ascending).$Top(3)
```

selects the three oldest records.

6.10.11.5. SELECTING MULTIPLE FIELDS AS A GROUP

This is useful when you have one ranking criterion that governs multiple fields as a group. These aggregation rules often look like "choose the best address record using these ranking criteria, then

choose STNAME, STNUM, UNIT, UNITNBR, CITY, STATE, and ZIP all at once." It looks more difficult than it actually is.

The first thing is to establish the ranking criteria, which is typically some combination of source priority and recency, such as:

```
$Filter(!$isADDRESSNull && !$isCITYNull && !$isZIPNull).$Sort($DATE,descending,$SOURCEPRIO,descending)
```

Then you repeat that same ranking criteria for each field:

```
STNUM : $First($Filter(!$isADDRESSNull && !$isCITYNull && !$isZIPNull).$Sort($DATE,descending,$SOURCEPRIO,descending), $STNUM)STNAME : $First($Filter(!$isADDRESSNull && !$isCITYNull && !$isZIPNull).$Sort($DATE,descending,$SOURCEPRIO,descending), $STNAME)...
ZIP : $First($Filter(!$isADDRESSNull && !$isCITYNull && !$isZIPNull).$Sort($DATE,descending,$SOURCEPRIO,descending), $ZIP)
```

This looks inefficient, but it is not because the `$Filter().$Sort()` selector chain is cached and reused for all of these expressions.

6.10.11.6. MULTIPLE AGGREGATORS IN AN EXPRESSION

Recall that the aggregating expression syntax allows for more than one aggregator to be embedded in a single expression. This is useful when you need to express different aggregations over the same (or different) record sets.

Example: Find ratio of offers to responses

```
$Count($Filter($EVENT_TYPE.equals("offer")) / $Count(All(), $EVENT_TYPE.equals("response"))
```

Example: Find difference between highest and lowest purchase

```
Math.abs($Max($Filter($PURCHASE > 0), $PURCHASE) - $Min($Filter($PURCHASE > 0), $PURCHASE)
```

Example: Find ratio of current and past YTD sales

This is somewhat complex, because the Year To Date concept is tricky to express using the LocalDate interface.

Filter for this year:

```
$Filter($DATE.getYear() == LocalDate.now().getYear())
```

Filter for last year YTD:

```
$Filter($DATE.getYear() == LocalDate.now().getYear() - 1 && $DATE.getDayOfYear() <= LocalDate.now().getDayOfYear())
```

Combined expression:

```
$Sum($Filter($DATE.getYear() == LocalDate.now().getYear()), $PURCHASE) / $Sum($Filter($DATE.getYear() == LocalDate.now().getYear() - 1 && $DATE.getDayOfYear() <= LocalDate.now().getDayOfYear()), $PURCHASE)
```

Note: This construct will not trigger nightly re-computation of affected aggregations.

6.10.11.7. MAKING DECISIONS

You can create expressions that make decisions based on aggregations using a Java *conditional expression.*

For example, "if total purchases this year exceed 10000, then A else B":

```
$Sum($Filter($DATE.getYear() == LocalDate.now().getYear()) > 10000 ? "A" : "B"
```

6.10.11.8. COMPLEX RANKING

Sometimes your ranking criteria for the $Sort selector is more than a simple calculation over fields. In this case, use a chain of *conditional expressions* to map conditions to ranks:

```
condition1 ? 1 : condition2 : 2 : condition3 ? 3 ... : N
```

For example:

```
$Sort($SOURCE.equals("A") ? 1 : $SOURCE.equals("B") && $CODE == 3 ? 2 :
$SOURCE.equals("B") && $CODE == 9 ? 3 : $ALTBIZ > 10 ? 4 : 5, descending)
```

To use this ranking in multiple places, you may find it easier to assign it to a computed field:

```
$RANK = $SOURCE.equals("A") ? 1 : $SOURCE.equals("B") && $CODE == 3 ? 2 :
$SOURCE.equals("B") && $CODE == 9 ? 3 : $ALTBIZ > 10 ? 4 : 5
```

Then later you can use the RANK field wherever you need it:

```
$Sort($RANK, descending)
```

6.11. CONFIGURABLE PROPERTIES

In MDM, configurable properties are available in:

- MDM core services
- MDM node properties
- MDM UI configuration file

6.11.1. MDM CORE SERVICES

MDM core services are configured via:

- MDM core services options
- MDM core services properties file

6.11.1.1. MDM CORE SERVICES OPTIONS

These are options that can be set on the MDM core services command-line or via environment variables. These are different than the settings found in the MDM core services properties file. Certain settings are available only via command-line options and environment variables, because the loading of properties files happens too late during startup.

Command-line options

Option	Meaning
-t	Set "trace" level logging. This is the most detailed logging level.

Option	Meaning
--traceLogging	Set "trace" level logging. This is the most detailed logging level.
-d	Set "debug" level logging. This is less detailed than "trace" but more than the default.
--debugLogging	Set "debug" level logging. This is less detailed than "trace" but more than the default.

Environment variables

Option	Meaning
MDM_MONGO_URI	The MongoDB connection string URI. The connection string must contain the username and password for connecting to MongoDB, and should be of the form: `MDM_MONGO_URI="mongodb://username:mongopassword@host.docker.internal:27017"`
MDM_JAVA_TRACE_LOGGING	Set "trace" level logging. This is the most detailed logging level. For example on Linux systems: `setenv MDM_JAVA_TRACE_LOGGING.` If there are conflicting log-level options, the highest level will be used.
MDM_JAVA_DEBUG_LOGGING	Set "debug" level logging. This is less detailed than "trace" but more than the default.

6.11.1.2. MDM CORE SERVICES PROPERTIES FILE

The following properties are recognized by the MDM core services. These can be overridden on the command-line (if running using `mdm-quickstart` application), or by supplying the properties override file `/usr/local/share/redpointmdm/redpointmdm.properties`.

These are text files, with each line containing property=value.

Property	Description	Default value
rpmdm.mongodb.uri	The MongoDB connection string URI.	mongodb://admin:redpoint@localhost:27017
rpmdm.authentication.url	URL to the authentication service. Must include protocol, host, port and path.	http://localhost:9901/mdm
rpmdm.usergroup.url	URL to the user/group service. Must include protocol, host, port and path.	http://localhost:9903/mdm

Property	Description	Default value
rpmdm.permission.url	URL to the permissions service. Must include protocol, host, port and path.	http://localhost:9908/mdm
rpmdm.mongodb.flavor	What flavor of MongoDB is this? One of: • mongodb • atlas	mongodb
rpmdm.mongodb.query_limit_in_values	Maximum number of values that can be processed in a single `$in` term for MongoDB or ATLAS	10000
rpmdm.mongodb.network_wait_seconds	Initial wait time in seconds after a network exception occurs talking to MongoDB.	1
rpmdm.mongodb.network_max_wait_seconds	Maximum wait time between tries to contact MongoDB.	5
rpmdm.mongodb.network_total_wait_seconds	Maximum total time we try to contact MongoDB from a network error before we give up.	160
rpmdm.mongodb.max_write_size	The maximum number of records which can be written in a single bulkWrite style operation.	10000
rpmdm.mongodb.max_awaits	The maximum number of times we can await for an database operation can fail.	100
rpmdm.committer.threads	Maximum number of threads to use when processing a commit.	4
rpmdm.committer.parallel_threshold	Number of ops in single changeset to trigger parallel commit algorithm	50000
rpmdm.mongodb.max_partitions	Maximum number of partitions to update in parallel during commit	30
rpmdm.mongodb.min_partition_size	Smallest partition size to allow during parallel commit	1000
rpmdm.db.max_unique_db_attempts	Maximum number of attempts to create a unique database name.	10

Property	Description	Default value
rpmdm.authentication.system-secret	Secret token to pass to `get_system_token` authentication call.	secret
rpmdm.authentication.system-secret.lifetime_ms	Duration in milliseconds that a system token is cached. Default is 10 minutes (10 * 60 * 1000)	600000
rpmdm.cache.block_cache_size	Maximum size of result blocks to cache, in bytes.	10485760
rpmdm.cache.cursor_cache_size	Maximum number of open query cursors to cache.	20
rpmdm.cache.max_age	Retention period, in seconds, for cached blocks and cursors.	300
rpmdm.cache.disable	Prevent the server from caching user requests regardless of the request "nocache" field value. Used for debugging.	false
rpmdm.spill_to_disk_threshold_bytes	The maximum amount of memory to use for sorting, buffering and related operations before spilling to disk.	10000000
rpmdm.merge_factor	When sorting large data, the maximum number of temp files that will be opened at once during a block-merge.	100
rpmdm.temp_folder	If specified, sets the folder to use for temporary files, instead of the default	
rpmdm.tempfile_compression	If true, compress temp files used during sorting and buffering	true
rpmdm.views.no_load	For testing only: Setting to true will prevent views from being loaded on startup.	false
rpmdm.views.clear	For testing only: Clear views from database upon startup.	false
rpmdm.views.join.threads	Number of threads to use for joins in computed views	4
rpmdm.workflow.badge_update_rate_seconds	Number of seconds between workflow badge updates	5

Property	Description	Default value
rpmdm.workflow.clear_peristent_store	For testing only: Clear all processes, workflows, and workitems before starting.	false
rpmdm.aggregation.source-directory	Root folder where auto generated aggregation Java files are saved. If empty, no files are saved.	./generated-source
rpmdm.subscription.max_call_rate	The most times a subscription will call a web service in a minute. The choice of calls per minute is to let this be an integer value and also to allow for significant throttling rates. The subscription delivery will wait at least (max_call_rate / 60 * 1000) milliseconds before making another call	6000
rpmdm.subscription.max_changeset_span	Used primarily for testing and field diagnosis, this limits the number of changesets that a subscription will process in one round	30000000
rpmdm.service.host	Host (bound interface) of published MDM service. If this is "localhost", only a local clients be able to connect. If this is "0.0.0.0", publish on all IP addresses.	0.0.0.0
rpmdm.service.port	Port of published service.	12345
rpmdm.service.secure	True to use https.	false
rpmdm.service.ssl_jks_file	The relative or absolute path to the certificate file. JKS format. Only used if secure is true. Do not enclose filenames in quotes.	
rpmdm.service.ssl_jks_password	The password used to encrypt the JKS certificate file. Only used if secure is true.	
rpmdm.service.logging	True to log requests and responses to MDM core services.	false

Property	Description	Default value
rpmdm.service.logging.size_limit	Limits the size of logged request and response packets	10000
rpmdm.proxy.logging	True to log requests and responses that MDM core services make to other services.	false
rpmdm.service.cors_allow_origin	Controls CORS requests for security. Set as Access-Control-Allow-Origin header in responses.	*
rpmdm.mongodb.allow_connect_fail	Normally MDM will attempt to connect (or reconnect) indefinitely. If this value is set to true, an error will occur if the connection can't be established after the timeout period.	false
rpmdm.mongodb.connection_timeout_ms	Override the connection timeout in the MongoDB URI.	30000
rpmdm.mongodb.socket_timeout_ms	Override the socket timeout in the MongoDB URI.	30000
rpmdm.mongodb.server_select_timeout_ms	Override the server-selection timeout in the MongoDB URI.	30000

The following properties are used only for testing:

Property	Description	Default
rpmdm.testing.mock_users	Create fake users and groups for testing.	false
rpmdm.testing.mongodb_proxy.enable	Enable the connection proxy.	false
rpmdm.testing.mongodb_proxy.host	Network-fail testing proxy will publish proxy to this host.	none
rpmdm.testing.mongodb_proxy.port	Network-fail testing proxy will publish proxy on this port. If replica sets or shards are in use, this will be the first port of several.	none
rpmdm.testing.mongodb_proxy.connection_timeout_ms	Override the connection timeout in the MongoDB URI.	none

Property	Description	Default
rpmdm.testing.mongodb_proxy.socket_timeout_ms	Override the socket timeout in the MongoDB URI.	none
rpmdm.testing.mongodb_proxy.init_delay_ms	Delay in milliseconds before connection proxy opens at initialization time	0
rpmdm.logging.iterator_interval	When logging record/key output, log every iterator_interval records.	100000
rpmdm.logging.iterator_length_limit	When logging records, clip the debug information to this many characters. Prevent very large records from overwhelming log file.	200

To override a property on the command-line of `mdm-quickstart`, use `--property=value`, for example:

```
--rpmdm.mongodb.uri=mongodb://redpoint:admin@dbserver.example.com:27017
```

6.11.1.2.1. SAMPLE FILE

You can start with `redpointmdm.properties` as a template:

```
#Uncomment lines that you want to override

#rpmdm.service.host=0.0.0.0
#rpmdm.service.port=9902
#rpmdm.service.logging=true
#rpmdm.service.logging.size_limit=10000
#rpmdm.service.cors_allow_origin=*

#rpmdm.authentication.url=http://rp-mdm-ui:9901/mdm
#rpmdm.usergroup.url=http://rp-mdm-ui:9903/mdm
#rpmdm.permission.url=http://rp-mdm-ui:9908/mdm

#rpmdm.proxy.logging=false

#rpmdm.authentication.systemsecret=secret
#rpmdm.authentication.systemsecret.lifetime_ms=60000
#rpmdm.cache.max_age=300
#rpmdm.cache.cursor_cache_size=20
#rpmdm.cache.disable=false
#rpmdm.mongodb.max_awaits=100


#rpmdm.mongodb.flavor=mongodb
#rpmdm.mongodb.network_wait_seconds=1
#rpmdm.mongodb.network_max_wait_seconds=5
#rpmdm.mongodb.network_total_wait_seconds=160
#rpmdm.mongodb.allow_connect_fail=false
#rpmdm.mongodb.query_limit_in_values=50000
#rpmdm.mongodb.max_write_size=10000
```

```
#rpmdm.committer.threads=4
#rpmdm.mongodb.max_partitions=30
#rpmdm.mongodb.min_partition_size=1000

#rpmdm.spill_to_disk_threshold_bytes=30000000
#rpmdm.merge_factor=100
#rpmdm.temp_folder=
#rpmdm.tempfile_compression=true

#rpmdm.logging.iterator_interval=100000
#rpmdm.logging.iterator_length_limit=200

#rpmdm.testing.mongodb_proxy.enable=false
#rpmdm.testing.mongodb_proxy.host=
#rpmdm.testing.mongodb_proxy.port=0
#rpmdm.testing.mongodb_proxy.connection_timeout_ms=0
#rpmdm.testing.mongodb_proxy.socket_timeout_ms=0
#rpmdm.testing.mongodb_proxy.server_select_timeout_ms=0
#rpmdm.testing.mongodb_proxy.init_delay_ms=0
#rpmdm.testing.mock_users=false
#rpmdm.testing.clear_committing_changesets=false

#rpmdm.workflow.clear_persistent_store=false

#rpmdm.aggregation.source-directory=
#rpmdm.aggregation.template-name=/net/redpoint/mdm/aggregation/template/MDMAggre-
gatorTemplate.txt
#rpmdm.aggregation.extra-classpath=

#rpmdm.views.no_load=false
#rpmdm.views.clear=false
#rpmdm.views.join.threads=4
#rpmdm.subscription.max_call_rate=6000
#rpmdm.subscription.max_changeset_span=100000000
```

6.11.2. MDM NODE PROPERTIES

The following properties are recognized by the MDM node services. These can be overridden by supplying the properties override file `/usr/local/share/redpointmdm/redpointjs.properties`.

Section	Property	Description	Default
	run_auth_service	Set to true to run the authentication service. This is at the top level and not in a section.	true
	run_permissions_service	Set to true to run the permissions service. This is at the top level and not in a section.	true
	run_user_service	Set to true to run the user service. This is at the top level and not in a section.	true

Section	Property	Description	Default
authentication	authority	The default authority to verify user credentials, either `mdm` or `activedir`.	mdm
authentication	host	External name of the authentication service.	rp-mdm-node
authentication	port	Authentication service port number.	9901
authentication	protocol	HTML protocol to use for the calling authentication service, either `http` or `https`. Can be set for calling the authentication service, but do not change to `https` on the host running the service.	http
authentication	system_user	Name of the system user. Overriding this is untested.	system
authentication	system_password	Default password that is put into the database if no system user exists.	system
authentication	system_group	Name of the system group. Overriding this is untested.	system
authentication	system_group_description	System group description used in creating the system group record if the node service creates the system group.	Users who are in the system group have permission to do anything.
authentication	everyone_group	Name of the everyone group. Overriding this is untested.	everyone
authentication	everyone_group_description	System group description used in creating the everyone group record if the node service creates the system group.	All users are in this group.
authentication	USER_EXPIRE_MILLISECONDS	Maximum duration of user session.	24 hours in milliseconds (86,400,000).

Section	Property	Description	Default
authentication	SYSTEM_EXPIRE_MILLISECONDS	Maximum duration of system token generated by getSystemToken.	5 minutes in milliseconds (300,000).
authentication.active_dir	url	Address and optionally the port of the LDAP (Lightweight Directory Access Protocol) service, for example: `ldap://myldapserver.mydomain.com`	
authentication.active_dir	domain	X.500 Directory Specification parts. Ask your system administrator for the correct value. CN (Common Name), OU (Organizational Unit) and DC (Domain Component) descriptions such as 'DC=datalever,DC=com'	
authentication.active_dir	bind_dn	LDAP Bind DN attribute. See LDAP documentation for details. Ask your system administrator for the correct value.	
authentication.active_dir	bind_credentials	LDAP Bind bind credentials. See LDAP documentation for details. Ask your system administrator for the correct value.	
authentication.active_dir	search_base	LDAP search base attribute. See LDAP documentation for details. Ask your system administrator for the correct value.	
authentication.active_dir	search_attributes	LDAP search attribute. Please See LDAP documentation for details. Ask your system administrator for the correct value.	

Section	Property	Description	Default
authentication.active_dir	tls_options	LDAP TLS options. Please see See LDAP documentation for details. Ask your system administrator for the correct value.	
core	host	Core service hostname	$JAVA_API_SERVICE environment variable if set, otherwise rp-mdm-core.
core	port	Core service port number	9902
core	protocol	HTML protocol for calling core service. `http` or `https`.	http
mongodb	uri	URI of the Mongo service. This is only provided for historic reasons. Use the $MDM_MONGO_URI environment variable.	mongodb://admin:redpoint@localhost:27017/admin
permission	host	External name of the permission service.	rp-mdm-node
permission	port	Permission service port number.	9908
permission	protocol	HTML protocol for calling permission service, `http` or `https`. Can be set for calling the permission service, but do not change to `https` on the host running the service.	http
permission	base_url	Base path used for requests serviced by the permission service. Part of every url used by the service. *{protocol}://{host}:{port}/{base_url}/{activity}*. Example: `http://rp-mdm-`	/mdm/permission/

Section	Property	Description	Default
		`node:9908/mdm/permission/validate`	
permission	MAX_PERMISSIONS	Maximum number of permissions returned when querying permissions.	10,000
user	host	User service hostname.	rp-mdm-node
user	port	User service port number.	9903
user	protocol	HTML protocol for calling user service, `http` or `https`. Can be set for calling the user service, but do not change to `https` on the host running the service.	http

6.11.3. MDM UI PROPERTIES FILE

In the MDM user interface (UI), configuration is handled via the `newConfig_js` file, located in the base `mdm-ui` directory (in the same place as `app.js`). It can be overridden and modified. The configuration is stored as name=value pairs, and is described in this properties list.

6.11.3.1. PROPERTIES LIST

Properties should be localized by adding name=value pairs to the file: `/usr/local/share/redpointmdm/redpoint_js.properties`.

If `$NODE_ENV` is set to `test`, property overrides are read from `redpoint_js.test.properties` instead.

SSL is supported via proxy only. Internally, MDM services run using HTTP and not HTTPS.

Property	Description	Default
trace_messages	Boolean value. If `true`, log contents of ajax messages as they pass through the middleware.	false
badge_update	Boolean value. If `true`, update badges in the web UI to indicate changes in workflow items. Normally set to `false` when debugging other issues to limit the messages sent to the services.	true

Property	Description	Default
version	Web service version number.	Injected from the `package.json` version
mdm.host	Web service hostname.	localhost
mdm.port	Web service port number.	9090
mdm.protocol	HTML protocol to use for web service, either `http` or `https`.	http
user.host	User service hostname.	localhost
user.port	User service port number.	9903
user.protocol	HTML protocol to use for user service, either `http` or `https`.	http
permission.host	Permission service hostname.	localhost
permission.port	Permission service port number.	9908
permission.protocol	HTML protocol to use for permission service, either `http` or `https`.	http
core.host	MDM Core service ("rpmdm") hostname.	Environment variable JAVA_API_SERVICE or localhost
core.port	Core service port number.	9902
core.protocol	HTML protocol to use for core service, either `http` or `https`.	http
authentication.host	Authentication service hostname.	localhost
authentication.port	Authentication service port number.	9901
authentication.protocol	HTML protocol to use for authentication service, either `http` or `https`.	http

KEYWORD INDEX

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W