Informatica® PowerExchange for MongoDB 10.4.0

User Guide

Informatica PowerExchange for MongoDB User Guide
10.4.0
December 2019

Publication Date: 2019-12-12

Table of Contents

Preface

Use the *Informatica® PowerExchange® for MongoDB User Guide* to learn how to read from and write to MongoDB by using the Developer tool. Learn to create a connection, develop mappings, and run sessions in an Informatica domain.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Introduction to PowerExchange for MongoDB

This chapter includes the following topics:

PowerExchange for MongoDB Overview

PowerExchange for MongoDB provides connectivity between Informatica and MongoDB. Use PowerExchange for MongoDB to extract and load MongoDB documents through the Data Integration Service.

You can use PowerExchange for MongoDB to integrate and migrate data from diverse data sources that are incompatible with MongoDB architecture.

You can use PowerExchange for MongoDB for the following data integration scenarios:

- Create a MongoDB data warehouse. You can aggregate data from MongoDB and other source systems, transform the data, and write the data to MongoDB.
- Migrate data from a relational database or other data sources to MongoDB. For example, you want to migrate data from a relational database to MongoDB. You can write data from multiple relational database tables with different schemas to the same MongoDB collection. A MongoDB collection contains the data in a MongoDB database.
- Move data between operational data stores to synchronize data. For example, an online marketplace uses a relational database as the operational data store. You want to use MongoDB instead of the relational database. However, you want to maintain the relational database along with MongoDB for a period of time. You can use PowerExchange for MongoDB to synchronize data between the relational data store and the MongoDB data store.
- Migrate data from MongoDB to a data warehouse for reporting. For example, your organization uses a business intelligence tool that does not support MongoDB. You must migrate the data from MongoDB to a data warehouse so that the business intelligence tool can use the data to generate reports.

Introduction to MongoDB

MongoDB is an open source, document based, NoSQL database that maintains dynamic schema. You can maintain more than one database on a MongoDB server.

A MongoDB database contains a set of collections. A collection is a set of documents and is similar to a table in a relational database. MongoDB stores records as documents that are similar to rows in a relational database. A document contains fields that are similar to columns in a relational database. A document can have a dynamic schema. A document in a collection does not need to have the same set of fields or structure as another document in the same collection. A document can also contain nested documents.

The following schema provides a sample MongoDB document from the collection called Product:

```
{
  sku: "111445GB3",
  title: "CM Phone",
  description: "The best in the world.",

  manufacture_details: {
    model_number: "CMP",
    release_date: new ISODate("2011-07-17T22:14:15.656Z")
  },

  shipping_details: {
    weight: 350,
    width: 10,
    height: 10,
    depth: 1
  },

  quantity: 99,

  pricing: [
    {region: "North America",
                    cost_price: 1000,
                sale_price:    1200},
               {region: "Europe",
                    cost_price: 1200,
                sale_price:    1500}
        ]

}
```

In the example, sku, title, description, quantity, manufacture_details, shipping_details, and pricing are fields. The fields manufacture_details and shipping_details are nested document type fields and pricing is an array type field.

PowerExchange for MongoDB Implementation

To extract and load MongoDB data, create a MongoDB data object in the Developer tool. You can include the data object as a source or target in a mapping. You can run the mapping or add the mapping to a workflow to process the data.

PowerExchange for MongoDB includes the Informatica MongoDB ODBC driver that connects to the MongoDB server. PowerExchange for MongoDB supports the MMAPv1 storage engine in MongoDB. You can create an ODBC connection to extract data from or load data to a MongoDB database. You can also configure the replica sets for the MongoDB server so that the Data Integration Service can access the secondary servers if the primary server is not available.

The Developer tool uses the schema of a collection, or you can define the schema for the collection before you import a data object. The Developer tool flattens the schema if there is any hierarchical element in the collection and retains the original schema of the collection when you import it.

The Developer tool imports a document based on the schema that you set for the collection. If a document contains hierarchical elements like arrays or nested documents, the Developer tool imports them as columns at the same level as other columns.

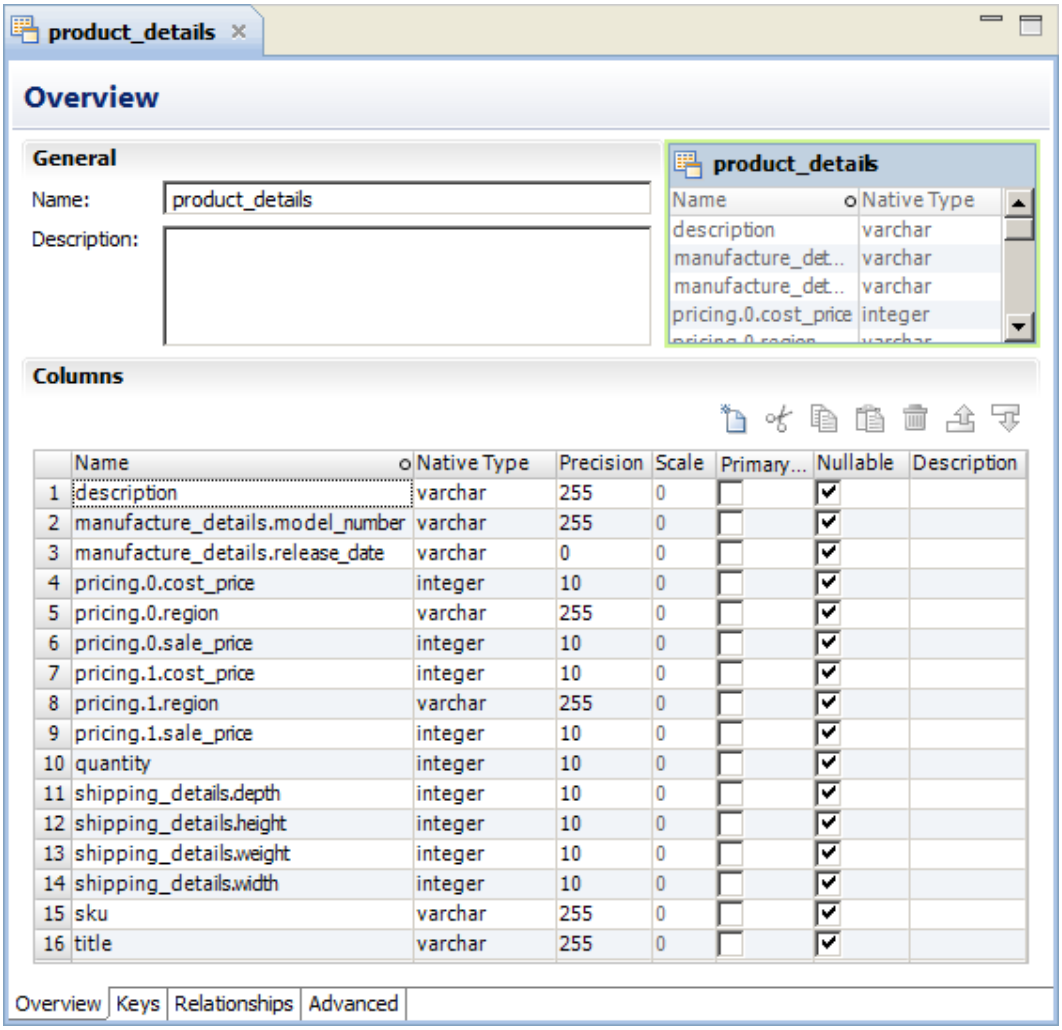
For example, you need to import the collection product_details with the following schema:

```
{
  sku: "sku_name",
  title: "product_name",
  description: "description",

  manufacture_details: {
    model_number: "model_number",
    release_date: new ISODate("date")
  },

  shipping_details: {
    weight: <value>,
    width: <value>,
    height: <value>,
    depth: <value>
  },

  quantity: <value>,

  pricing: [
    {region: "North America",
                  cost_price: 1000,
                sale_price:    1200},
               {region: "Europe",
                  cost_price: 1200,
                sale_price:    1500}
        ]
}
```

The Developer tool imports the collection schema into a tabular format. You can identify arrays and nested documents with the naming convention of the column. The naming convention of a nested document is <top level element name>.<nested document name>.<nested document element name>. The naming convention of an array is <array name>.<element number>.

If a column name in a document or a key name in an array element exceed 128 characters, the Developer tool truncates the column name or key name to 128 characters. When multiple column names exceed 128 characters each, multiple columns are truncated. As a result, duplicate column might be created, and data preview failure might occur. To prevent failure of data preview, ensure that duplicate column names do not occur.

The following figure shows the source definition when you import the collection into the Informatica Developer and the delimiter is a period (.):



When you run a mapping, the Data Integration Service uses the MongoDB ODBC data source name in the machine that runs the Data Integration Service to extract data from or load data to a MongoDB database.

CHAPTER 2

PowerExchange for MongoDB Configuration

This chapter includes the following topics:

PowerExchange for MongoDB Configuration Overview

You can use PowerExchange for MongoDB on Windows or Linux. You must configure PowerExchange for MongoDB before you can extract data from or load data to MongoDB database.

Prerequisites

You must complete the prerequisites before you can use PowerExchange for MongoDB.

Complete the following prerequisites:

- Install or upgrade Informatica.
- Ensure that you have the PowerExchange for MongoDB license file. You do not require a separate ODBC license to use PowerExchange for MongoDB.
- On Windows, download and install the Microsoft Visual C++ 2010 Redistributable Package in server and client machines from the Microsoft website. For example, download the vc_redist_x86.exe file.

For more information about product requirements and supported platforms, see the Product Availability Matrix on Informatica Network:
https://network.informatica.com/community/informatica-network/product-availability-matrices

PowerExchange for MongoDB Upgrade

Before you upgrade to Informatica 10.4.0, back up the `odbc.ini` file.

After you upgrade to Informatica 10.4.0, perform the following steps:

- For the Simba MongoDB ODBC 64-bit driver running on a Linux or SUSE machine, replace the `odbc.ini` file with the backup copy of the `odbc.ini` file, and verify if the MongoDB driver name in the `odbc.ini` file is `libinformaticamongodbodbc64.so`.
- To use the Schema Editor for the Simba MongoDB ODBC driver, running on a Windows machine, apply the Informatica EBF-13871.
  For more information about applying the EBF-13871, contact Informatica Global Customer Service.

Informatica MongoDB ODBC Driver Configuration

The Informatica MongoDB ODBC driver is installed on the machines where you install Informatica services and clients. Configure the Informatica MongoDB ODBC driver on those machines.

The Developer tool uses the Informatica MongoDB ODBC driver to import MongoDB collections as source or target definitions. The Data Integration Service uses the driver to extract data from or load data to the MongoDB database. Create ODBC data source names to connect to the MongoDB database.

Configuring the Informatica MongoDB ODBC Driver on Linux

You must configure the Informatica MongoDB ODBC driver with details of the MongoDB database and ODBC driver manager before you can run MongoDB mappings.

Edit the `odbc.ini` file to configure the driver in the following location: `<INFA_HOME>/tools/mongodb/Setup`

1. Enter the correct ODBCInstLib for the ODBC Driver Manager in all the .ini files.
2. Replace <INSTALL_DIR> with the path to the Informatica services installation directory in all the .ini files.
3. Add the following information to the LD_LIBRARY_PATH environment variable:
   - <INFA_HOME>/tools/mongodb/lib
   - 64-bit library directory of the ODBC Driver Manager
4. Add the path of the odbc.ini file to the ODBCINI environment variable.
5. Add entries for all the MongoDB data sources in the `odbc.ini` file.

   The following section shows a sample entry in the `odbc.ini` file:

```
[Sample Informatica MongoDB DSN]
Description=Informatica MongoDB ODBC Driver DSN
Driver=]<INFA_HOME>/tools/mongodbodbc/lib/libinformaticamongodbodbc64.so
Host=[Host]
Port=[Port]
Database=[Database]
ReadPreference=primary
ReplicaSetName=""
SecondaryServers=""
UseReplicaSet=0
VirtualTableDetection=0
VTAnyMatchColumnsDetection=0
VTAnyMatchString=ANY
VTAnyMatchTableNameSuffix=any
VTArrayCountPrefix=Number of
```

```
VTHideRealTables=0
VTIndexColSuffix=index
VTInsertUpdateSafeMode=0
VTKeyColumnSeparator=.
VTMainTableNameSeparator=main
VTMainTableShowArrayCounts=0
VTTableNameSeparator=_vt_
DefaultBinaryColumnLength=32767
DefaultContainerColumnLength=511
DefaultJSONColumnLength=1023
DefaultStringColumnLength=255
CheckGetLastError=1
OmitColumns=1
TruncateDocument=0
UpdateMultipleRows=1
NestedColumnSeparator=__
SchemaDetectRealColumnsMax=10000
SchemaDetectSampleSize=100
SchemaDetectSampleStrategy=End
SchemaDetectShowContainerColumns=0
ArrayColumnMax=5
AuthenticationDatabase=
CacheMetadata=1
ExportMetadataToFile=
ExportSchemaMapTo=
ExtendedJSON=0
ImportSchemaMapFrom=
LocalMetadataFile=
ResetFileFromMetadata=
RowsFetchedPerBlock=4096
UpsertOnUpdate=0
UseJsonColumn=0
```

Data Source Name Configuration on Windows

Configure the connection properties, advanced properties, and schema when you configure a data source name.

You must create a data source name in the ODBC datasource administrator to extract data from and load data to a MongoDB database. The connection properties provide information for the MongoDB server and the database. The advanced properties are read and write operations. You can also define a schema after you create a database.

You can find the ODBC datasource administrator in the Control Panel on Windows. Configure the ODBC data source name in the 64-bit ODBC datasource administrator in the client and the machines where you install the Informatica services. You can access the 64-bit ODBC datasource administrator, odbcad32.exe, in 64-bit Windows from the following location: `C:\Windows\System32`

MongoDB ODBC Connection Properties

You must configure a MongoDB ODBC data source before you can import MongoDB data sources.

The following table describes the MongoDB ODBC connection properties:

Property	Description
Data Source Name	Name of the data source name.
Description	Description to identify a data source name.
Host	Host name of the MongoDB server.
Port	Port from which you can access MongoDB.
Database	MongoDB database in the server that you want to access.
Username	Optional. MongoDB user name.
Replica Set Name	Optional. Name of the replica set of the database.
Additional Servers	Optional. Host names of the secondary MongoDB servers.

Advanced Properties

Configure the advanced properties when you create a data source name.

The following table describes the advanced properties in the Informatica MongoDB ODBC driver:

Property	Description
Documents fetched per block	The maximum number of documents that the Data Integration Service reads for every call to the MongoDB database. Default is 4096.
Nested column separator	Separator character for arrays and nested documents. The nested column separator must be consistent across connections used in a mapping. For example, if one connection uses the period (.) as the nested column separator and another connection in the same mapping uses the underscore (_) as the separator, then the mapping fails. You can use either the underscore (_) or the period (.) as the nested column separator. Default is period (.).
Maximum number of columns to flatten	The maximum number of array elements that the ODBC driver flattens into multiple nested columns. Default is 5.

Property	Description
Read preference	Server that you prefer to read data from if you configure replica sets. You can select one of the following server options: - Primary. The Data Integration Service reads data from the primary server. If the primary server is offline, the session fails. - Primary Preferred. The Data Integration Service reads data from the primary server if the primary server is available. If the primary server is offline, the Data Integration Service reads data from the secondary server. - Secondary. The Data Integration Service reads data from the secondary server. If the secondary server is offline, the session fails. - Secondary Preferred. The Data Integration Service reads data from the secondary server if the secondary server is available. If the secondary server is offline, the Data Integration Service reads data from the primary server. - Nearest. The Data Integration Service reads data from the nearest available server. Default is primary.
Sampling strategy	Number of rows to scan in the schema definition. You can select one of the following sampling strategies: - Start. Scans the specified number of rows from the start. - End. Scans the specified number of rows from the end. - Random. Scans the specified number of rows in random order. Default is End.
Documents to sample (0 to sample all documents)	Number of documents to scan. Default is 100.
String Columns Lengths	The string column length to use for the fields. You can select one of the following string column lengths: - Standard. The string column length to use for the standard fields. Default is 255. - Container. The string column length to use for the container fields. Default is 511. - DocumentAsJSON. The string column length to use for the documentAsJSON fields. Default is 1023.
Use SQL_WVVARCHAR for String datatype	The Data Integration Service maps the String datatype to SQL_WVARCHAR ODBC instead of SQL_VARCHAR. Default is disabled.
Enable reading/writing as JSON document.	Read or write data as a JSON document. If enabled, the driver reports a special column named documentAsJSON that retrieves or stores whole documents as JSON formatted strings. Default is disabled. **Note:** For a MongoDB connection, if you toggle between enabling and disabling this option, the metadata cache might lose its integrity. Instead of changing the Enable reading/writing as JSON document property for a MongoDB connection, create separate connections with this property.
Show container columns when generating metadata	Show the container columns when the Integration Service generates the metadata. Default is disabled.
Enable SSL	Not Applicable.
Check GetLastError on writes	Calls the MongoDB CheckGetLastError() function to check for failures after a write operation. Default is enabled.

Property	Description
Enable Updating Multiple Rows	The Informatica MongoDB ODBC driver updates multiple rows for each Data Integration Service write call. If enabled, the driver updates all rows that match the filter condition. If disabled, the driver updates only the first row that matches the filter condition. Default is disabled.
Omit default NULL column on insert	The Data Integration Service does not write columns with NULL value to a MongoDB target. Default is enabled.
Truncate documents larger than 16 MB	Truncate the document size to 16 MB when you load data to MongoDB. Default is disabled.
Active Metadata Location	Read metadata changes from the MongoDB database or from a local file. Required if you choose to store the metadata in a local file. Default is database.

CHAPTER 3

Schema Definition

This chapter includes the following topics:

Schema Definition Overview

You can define the schema for a MongoDB collection that you want to import as a data object in the Developer tool. You can define the schema for multiple collections with the same ODBC data source name.

A collection in MongoDB might contain several fields that you do not want to import. When you define the schema you can limit metadata that you import. The driver dynamically detects the collection schema of a MongoDB database. It flattens the MongoDB schema and displays the keys in the a tabular format with each key as a column in the Schema Editor.

You can export the collection to an external schema definition file and edit the schema definition in the Schema Editor. After you modify the collection properties and column metadata, you can save the modifications in the schema definition file. The driver does not modify the schema of the actual MongoDB collection. You can choose to store the modifications in the MongoDB database or as a file.

If you enable virtual table detection in the Informatica MongoDB ODBC driver, the driver creates virtual tables in the schema if the collection contains arrays. You can import the virtual table as a data object in the Developer tool.

Schema Editor

Use the Schema Editor to view or edit the MongoDB collection schema that you want to import.

You can access the Schema Editor from the ODBC Data Source Administrator when you configure the Informatica MongoDB ODBC Driver DSN. You can also find the Schema Editor in the following location:
`$INFA_HOME/clients/tools/mongodb/Tools`

When you define a schema in the Informatica MongoDB ODBC driver DSN, you must specify a schema definition file. You can use an existing schema definition file or create a new one. After you specify a schema

definition file, you can import the collections in the database to the schema definition file. You can import all the collections in the database or a particular collection. You can use a JSON filter to filter records on a collection. You can also export those collections that are missing in the schema definition file.

When you open the Schema Editor, all the databases and collections in the schema definition file appear. When you select a collection, the collection properties and document properties appear on the right pane. You can modify the properties and save the schema. You can also save the schema changes to a new schema definition file.

Collection Properties

Before you import a collection, you can view or edit the properties associated with the collection in the Schema Editor.

You can view or edit the following collection properties in the Schema Editor:

ODBC Table Name

The name of the collection to use for the schema. Default is the same as source table name. You can modify this value to match the name that you require when you import the ODBC data source in the Developer tool.

ODBC Catalog Name

The name of the catalog that to use for the schema. Default is the same as source catalog name. You can modify this value to match the name that you require when you import the ODBC data source in the Developer tool.

Source Table Name

The name of the collection in the source database. You cannot modify this value.

Source Catalog Name

The name of the source database. You cannot modify this value.

Virtual Type

Indicates whether the collection is a virtual collection or not. Reserved for future use.

Permissions

The permissions assigned to you. Reserved for future use.

Column Metadata

When you select a collection in the Schema Editor, you can view or modify the column metadata of the collection.

The following fields are available in the column metadata:

ODBC Column Name

The name of the column that you want to use in the database schema. Default is the source column name. You can modify this value to match the name that you require when you import the ODBC data source in the Developer tool.

SQL Type

The ODBC data type of the column. The Data Integration Service uses the SQL type when you run the session that uses the ODBC data source. You can modify the datatype based on your requirement.

Source Column Name

The name of the column in the source database. You cannot modify this value.

Source Type

The data type of the column in the source database. The Data Integration Service uses the SQL type when you run the session that uses the ODBC data source. You can modify the datatype based on your requirement.

Hide Column

You can choose to hide the column so that it does not appear in the schema.

Behavior

The behavior field shows whether the column is scalar or a container. Scalar columns contain a single value like an integer or a string. Container columns have multiple values. Arrays and documents are examples of container columns.

Note: Container columns do not support transformations.

Key Type

The key type field shows whether the column is a key column.

The following values are possible for the key type:

- Primary key
- Foriegn key
- Not a key

You cannot modify the key type of a column.

ODBC Type Hint

The ODBC type hit field shows the possible ODBC datatype of the column. You can choose the SQL type of a column based on the hint.

Source Nesting Level

The source nesting level field displays the level at which the column is nested in the document metadata. You can use the MongoDB ODBC driver to read up to five levels of nested columns and write up to three levels of nested columns.

Alternate Source Type

The alternate source type field displays the alternate data type of the column in the source database.

Virtual Tables

You can configure the Informatica MongoDB ODBC driver to create virtual tables in the schema if the collection contains arrays.

Virtual tables depict the normalized view of a MongoDB collection. You can import virtual tables as an ODBC data object and create mappings.

To configure virtual table creation, open the Informatica MongoDB ODBC Driver DSN. In the **Schema Definition** dialog box, click **Virtual Table Options**.

If you enable virtual table creation, the driver creates the following virtual tables:

Main virtual table

The main virtual table contains all the data from the original MongoDB collection except the data in the arrays. The driver replaces the cells that contain arrays with the number of arrays in the cell.

The main virtual table use the following naming convention by default: `<original collection name>_vt_main`

The columns that contain arrays use the following naming convention by default: `Number of <original column name>`

Virtual table for array columns

The driver creates a virtual table for each column that contain arrays.

The virtual table for an array column uses the following naming convention by default:`<original collection name>_vt_<original column name>`

Each virtual table has a key column that references back to the primary key column in the original collection. The key column uses the following naming convention by default: `<original collection name>.<primary key column name>.`

The virtual table has an index column that shows the position of the data within the original array. The index column uses the following naming convention by default: `<original column name>.index`

Other columns in the virtual table represent the elements in the array and are named after the array element. If the array is of scalar type, the data column uses the following naming convention by default:`<original column name>.value`

Note: You cannot use a DD_DELETE strategy in an Update Strategy transformation to delete rows from a virtual table. You also cannot use the MongoDB ODBC driver to add an array element to an existing array index because of a limitation from the C API used by the MongoDB driver.

Virtual Table Options

Configure the virtual table options to create virtual tables for a collection that contains arrays.

The following table describes the virtual table options in the Informatica MongoDB ODBC driver:

Property	Description
Enable Virtual Table Detection	The driver creates virtual tables if the collection contains arrays. Default is disabled.
Virtual Main Table Suffix	The suffix for the main virtual table. Default is main.
Virtual Key Column Separator	The separator for the key columns in a virtual table. The virtual key column separator must be consistent across connections used in a mapping. For example, if one connection uses the period (.) as the virtual key column separator and another connection in the same mapping uses the underscore (_) as the separator, then the mapping fails. You can use either the underscore (_) or the period (.) as the virtual key column separator. Default is period (.).
Virtual Table Name Separator	The separator in the virtual table name. Default is _vt_. **Note:** If tables in the MongoDB database and virtual tables have the same names, metadata import might be corrupted. To avoid importing corrupted metadata, do not use table names that contain the virtual table separator in the MongoDB database.
Virtual Table Index Column Suffix	The suffix for the virtual table index column. Default is index.

Property	Description
Hide Real Table if Virtual Tables Created	Hide the real tables if the corresponding virtual tables are created. Default is disabled.
Show Array Counts In Virtual Main Table	The virtual tables contain columns that show the array count. Default is disabled.
Virtual Table Array Count Prefix	The prefix for the virtual table array count column. Default is Number of.
Enable Any Match Columns Detection	The driver filters the data and selects rows where a value in a top-level array matches a specified expression and then returns the results as columns in a virtual table.
Any Match Table Name Prefix	The prefix for naming the array column in an any match virtual table.
Any Match Column Separator	The separator for naming the columns in an any match virtual table.

Virtual Tables - An Example

The collection CustomerTable contains arrays. You want to create virtual tables from the arrays and import the virtual tables as data objects in the Developer tool.

The following table shows the schema of CustomerTable collection:

id	Customer Name	Invoices	Service Level	Contacts	Ratings
1111	John	[{invoice_id=123,item=toaster, price=456,discount=0.2}, {invoice_id=124,item=oven, price=12345, discount=0.3}]	Silver	[{type=primary,name="John Johnson"}, {type=invoicing,name="Jane Johnson"}]	[7,8]
2222	Jane	[{invoice_id=125,item=blender, price=7456,discount=0.5},	Gold	[{type=primary,name="Jane Johnson"}	[5,6]

If you enable virtual table detection, the driver creates the following virtual tables:

CustomerTable_vt_main

The following table shows the schema of CustomerTable_vt_main virtual table:

id	Customer Name	Number of Invoices	Service Level	Number of Contacts	Number of Ratings
1111	John	2	Silver	2	2
2222	Jane	1	Gold	1	2

CustomerTable_vt_Invoices

The following table shows the schema of CustomerTable_vt_Invoices virtual table:

CustomerTable.id	Invoices_index	invoice_id	item	price	discount
1111	1	123	toaster	456	0.2
1111	2	124	oven	12345	0.3
2222	1	125	blender	7456	0.5

CustomerTable_vt_Contacts

The following table shows the schema of CustomerTable_vt_Contacts virtual table:

CustomerTable.id	Contacts_index	type	name
1111	1	primary	John Johnson
1111	2	invoicing	Jane Johnson
2222	1	primary	Jane Johnson

CustomerTable_vt_Ratings

The following table shows the schema of CustomerTable_vt_Ratings virtual table:

CustomerTable.id	Ratings_index	Ratings_value
1111	1	7
1111	2	8
2222	1	5
2222	2	6

Metadata Caching

The Informatica MongoDB ODBC driver caches the schema in the MongoDB database or a flat file. After you define a schema for the collection, you can store the modifications in the MongoDB database or a file so that the Developer tool uses the modifications each time you import a definition.

You must modify the schema definition if there are updates to the documents that require a change in the definitions that you created in the Developer tool.

If you store the schema modification in a file, ensure that the file is available in the location that you configure in the ODBC data source name when you import a data object. If you store the schema modification in the MongoDB database, PowerExchange for MongoDB stores the schema modification in a collection called Mersenne_Collection_Metadata. If you edit Mersenne_Collection_Metadata, you may lose the schema modifications.

Note: If you clear the metadata cache, you must re-create or re-import the source and target objects with the same metadata that the existing mapping objects use.

Defining the Schema for a Collection

You can modify and define the schema for a collection that you want to import as data object in the Developer tool.

1. Open the ODBC Data Source Administrator.
2. Select the Informatica MongoDB ODBC Driver DSN.
3. Click **Configure**.
4. Click **Schema Definition**.

   The **Schema Definition** dialog box appears.
5. Click **Browse** and select a schema definition file.

   You can also enter a file name in the file selection dialog box to create and use a new schema definition file.
6. Choose one of the following collection export options to the schema definition file:
   - Export all the collections in the MongoDB database.
   - Export the tables that are missing from the schema definition file and available in the MongoDB database.
   - Select a particular collection in the MongoDB database. Optionally, you can enter a JSON filter statement to filter records.
7. Click **Launch Schema** Editor.

   The Schema Editor application appears.
8. Select a collection and define the schema in the Schema Editor according to the requirement.
9. Close the schema editor after you save the changes.

   You can also save the schema changes to a different schema definition file.
10. Select whether to store the metadata in the MongoDB database or in a local file.
11. Click **Import File** to store metadata definition from the schema definition file.

    If you read the metadata from a file instead of the MongoDB database, place the schema definition file in the same folder as the metadata file.

Updating the Schema File

You can update the schema file to reflect metadata changes in the MongoDB database or make changes in the imported metadata.

1. Open the schema definition by using the Informatica MongoDB ODBC Driver DSN.
2. Click **Browse** and select a schema definition file.

   You can also enter a file name in the file selection dialog box to create and use a new schema definition file.

3. Export the metadata to the SSD file.
   a. To export the metadata imported by using the MongoDB ODBC driver, click **Export Existing**.
   b. To export metadata sampled from the MongoDB database, click **Generate All**.
   c. To export any missing tables and add metadata, click **Generate Missing**.
4. From the **Database source table** list, select the table to be updated.
5. Click **Generate Table** to update the schema of the table from the database.
6. Click **Edit Schema File** to open the schema file that you exported.
7. In the **Schema Editor**, make the required modifications in the schema file to reflect the metadata changes.

   Note: When you update metadata, press **Enter** and then click **Save** to ensure that the changes to the metadata are saved.
8. Save the schema file and close the **Schema Editor** dialog box.
9. In the **Schema Definition** dialog box, click **Update Metadata** to replace the metadata with the metadata from the SSD file.

CHAPTER 4

MongoDB Read Operations

This chapter includes the following topics:

MongoDB Read Operations Overview

You can import a MongoDB collection as an ODBC data object in the Developer tool and use it as a source in a mapping.

When you run a MongoDB mapping, the Data Integration Service uses the Informatica MongoDB ODBC data source to extract data from MongoDB.

You can configure the optimization levels at the following locations:

- Mapping runtime configuration in the Developer tool
- Data viewer runtime configuration in the Developer tool
- Mapping deployment in application settings in the Developer tool
- Mapping configuration of an application in the Data Integration Service through the Administrator tool

You can configure advanced reader properties for the Informatica MongoDB ODBC Driver in the ODBC driver properties.

You can configure the following read options in the ODBC driver properties:

Read Preference

MongoDB server that you prefer to read data from if you configure replica sets.

You can select one of the following MongoDB server options:

- Primary. The Data Integration Service reads data from the primary MongoDB server. If the primary MongoDB server is offline, the session fails.
- Primary Preferred. The Data Integration Service reads data from the primary MongoDB server if the primary MongoDB server is available. If the primary MongoDB server is offline, the Data Integration Service reads data from the secondary MongoDB server.
- Secondary. The Data Integration Service reads data from the secondary MongoDB server. If the secondary MongoDB server is offline, the session fails.

- Secondary Preferred. The Data Integration Service reads data from the secondary MongoDB server if the secondary MongoDB server is available. If the secondary MongoDB server is offline, the Data Integration Service reads data from the primary MongoDB server.
- Nearest. The Data Integration Service reads data from the nearest available MongoDB server.

Enable Reading/Writing as JSON

Reads MongoDB datasource as a JSON document. If you select the option, a column with documentAsJSON appears in the collection when you read data from MongoDB from which you can read data as JSON.

Documents fetched per block

The maximum number of documents fetched from the MongoDB server for every read request. If more documents are available for a query, the Data Integration Service makes further read requests to the MongoDB server. Default is 4096.

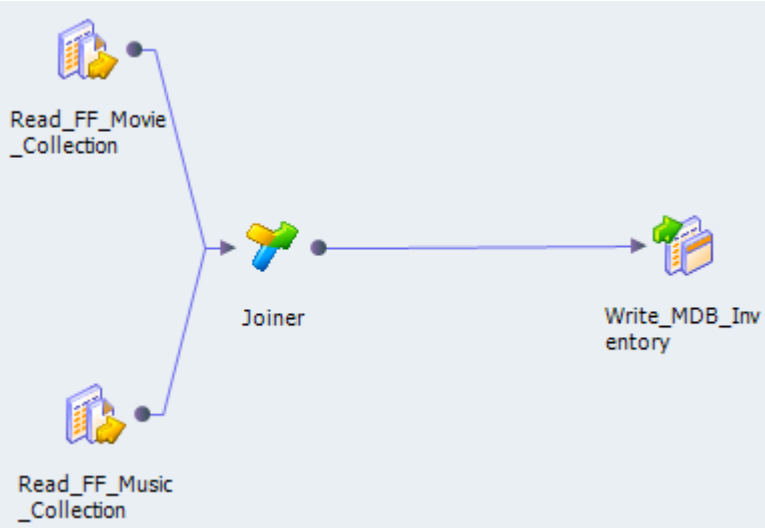
Example: Data Migration to MongoDB

A media store uses flat files with comma-separated values to store details of the store inventory with a unique flat file for each type of media. The file FF_Music_Collection stores the details of audio CDs and FF_Movie_Collection stores the details of movie DVDs and Blu-ray disks.

You want to use a MongoDB database to store all inventory details. Create a mapping to extract data from FF_Music_Collection and FF_Movie_Collection and load it to the MongoDB collection MDB_Inventory.

Create a mapping with two flat file source definitions to read the records from the flat files. Include the MongoDB target definition to write data from the flat files. Use a Join transformation to join the columns before writing to the corresponding MongoDB columns.

The following figure shows the mapping:



FF_Music_Data Source

The following table describes the contents of FF_Music_Collection:

Field	Datatype
Name	String
Artist	String
Units	Integer
Cost Price	Integer
Sale Price	Integer

FF_Movies_Data Source

The following table describes the contents of FF_Movies_Collection:

Field	Datatype
Name	String
Director	String
Artist1	String
Artist2	String
Type	String
Units	Integer
Cost Price	Integer
Sale Price	Integer

MDB_Inventory Target

The collection MDB_Inventory stores audio CD information and movies disks information.

The following sample document shows an audio CD document in the collection:

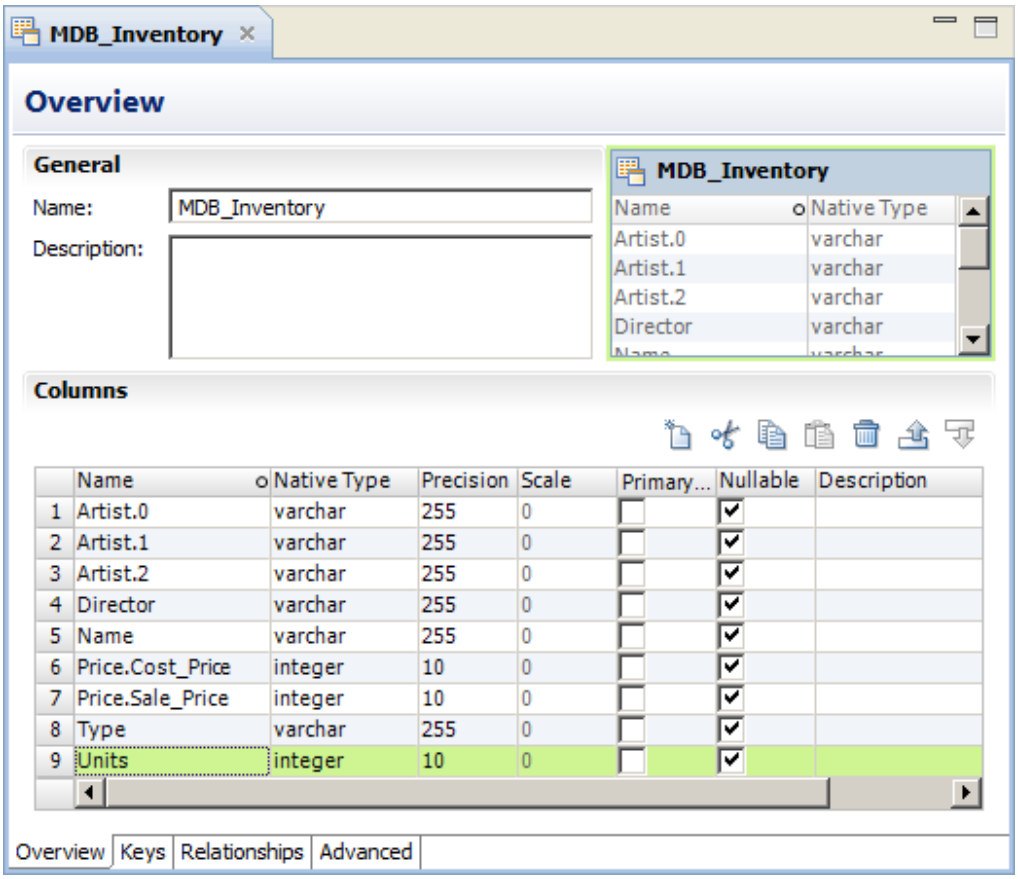
```
{
  "Name" : "Happy Birthday",
  "Artist" : ["Patty Hill", "Mildred J. Hill", "Derek Underhill"],
  "Units" : 1000,
  "Price" : {
    "Cost_Price" : 1,
    "Sale_Price" : 3
  }
}
```

The following sample document shows a movie disk document in the collection:

```
{
  "Name" : "City Lights",
        "Type" : "Blu-ray",
        "Director" : "Charlie Chaplin"
```

```
    "Artist" : ["Charle Chaplin", "Mildred J. Hill", "Derek Underhill"],
    "Units" : 1000,
    "Price" : {
      "Cost_Price" : 10,
      "Sale_Price" : 15
    }
  }
```

The following figure shows the data object that you import in the Developer tool:



CHAPTER 5

MongoDB Write Operations

This chapter includes the following topics:

MongoDB Write Operations Overview

You can import a MongoDB collection as an ODBC data object and create mappings to write data to MongoDB in the Developer tool.

You must configure the ODBC driver and define the MongoDB schema before you import MongoDB collections.

When you run a MongoDB mapping, the Data Integration Service uses the Informatica MongoDB ODBC data source to load data to the MongoDB database.

When you run MongoDB mappings, ensure that the optimization level is set to none. If you do not set the optimization level as none, the mappings may fail.

You can configure the optimization levels at the following locations:

- Mapping runtime configuration in the Developer tool
- Data viewer runtime configuration in the Developer tool
- Mapping deployment in application settings in the Developer tool
- Mapping configuration of an application in the Data Integration Service through the Administrator tool

You can configure advanced write options for the Informatica MongoDB ODBC Driver in the ODBC driver properties.

You can configure the following write options in the ODBC driver properties:

Omit default null columns on insert

Drops columns with null values. Default is enabled.

Truncate documents larger than 16 MB

Truncates a document if the size is more than 16 MB in a writer mapping. MongoDB documents have a size restriction of 16 MB. If enabled, the Data Integration Service truncates the document that exceeds 16 MB when writing to MongoDB. If you disable the option when you run a write session, the Data Integration Service rejects the document that exceeds 16 MB. Default is disabled.

Enable Reading/Writing as JSON

Writes the JSON format of the data to the MongoDB document. If you select the option, a column with the field documentAsJSON appears in the collection when you write data to MongoDB. You cannot write into individual columns if you select this option. Default is disabled.

Enable updating multiple rows

Updates multiple rows in the MongoDB collection for every write operation. If there are multiple documents to update, the Data Integration Service updates multiple documents in the MongoDB collection for every write operation. If you clear this option and multiple documents require update, the Data Integration Service initiates write operation for each document update. Default is disabled.

Check GetLastError on writes

Calls the MongoDB CheckGetLastError() function to check for failures after each insert or update operation. Select this option to include fault tolerance in write operations. Clear this option to speed up the write operation. Default is enabled.

Note: If the source and target data objects in the mapping contains container columns, connect either the container columns or the individual columns. If both the container columns and the corresponding individual columns are connected to the target data object, the Data Integration Service inserts duplicate columns in the MongoDB database.

MongoDB as an Operation Data Store – An Example

A large online music store, Moose, uses MongoDB as the operational data store for the business inventory details.

The business analysts at Moose use a business intelligence tool that does not support reading data from MongoDB. The tool requires the input data to be in a relational database or a flat file.

The data warehouse includes a collection called Music_Contents. The collection Music_Contents contains a catalog of all of the songs in the store. You must move the data in the collection to a flat file to use the data for business analysis. You must also remove those records with zero units to ensure that the data is current.

The following table describes the structure of Music_Contents:

Field	Datatype
Name	String
Type	Array of strings
Artist	Array of strings
Units	Int
Price	Nested document

The following table describes the structure of the nested document, Price:

Field	Datatype
Cost_Price	Int
Sale_Price	Int

The following document is a sample from the collection, Music_Contents:

```
{
  "Name" : "Happy Birthday",
  "type" : ["Folk", "Traditional"],
  "Artist" : ["Patty Hill", "Mildred J. Hill", "Derek Underhill"],
  "Units" : 1000,
  "Price" : {
    "Cost_Price" : 1,
    "Sale_Price" : 3
  }
}
```

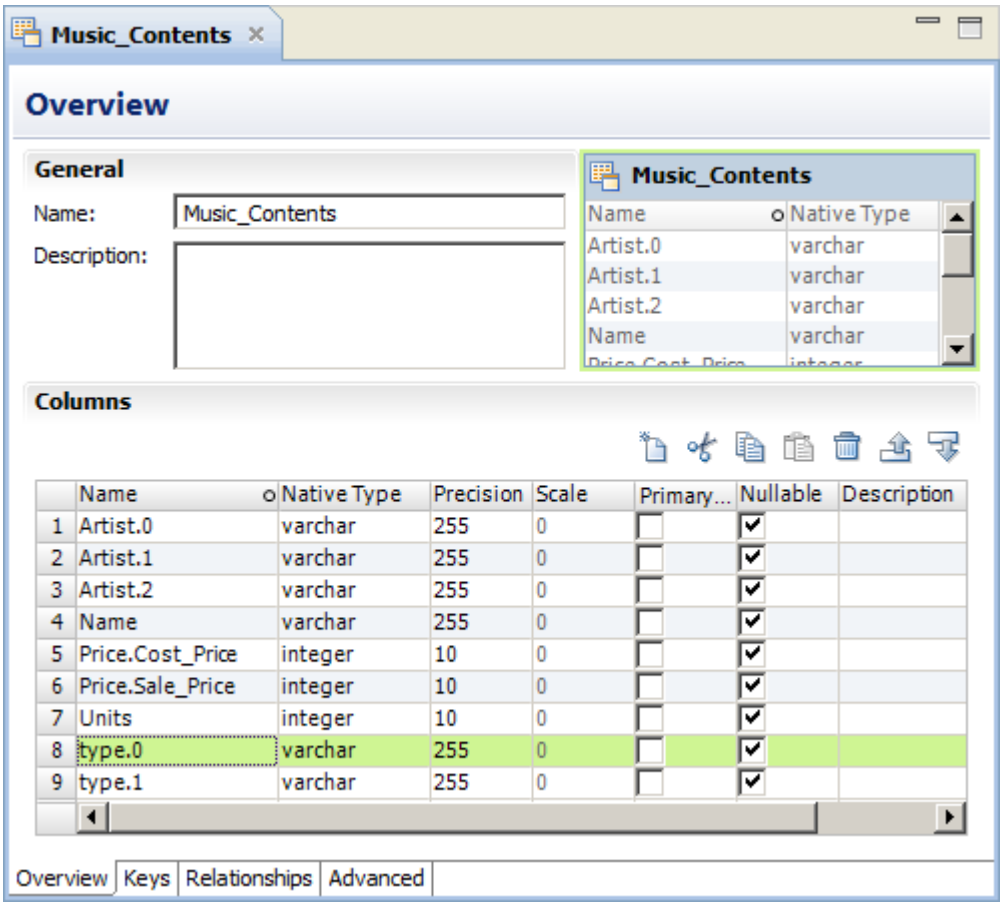
Create a mapping with a MongoDB data object as the read transformation to read the records from the collection. Include a flat file data object as the target in the mapping so that the business intelligence tool can consume the data. Use a Filter transformation to remove the documents that have zero units.

Create a mapping that has a MongoDB data object in read mode, a Filter transformation, and a flat file data object in write mode. The MongoDB reader mapping contains the following components:

MongoDB ODBC Data Object

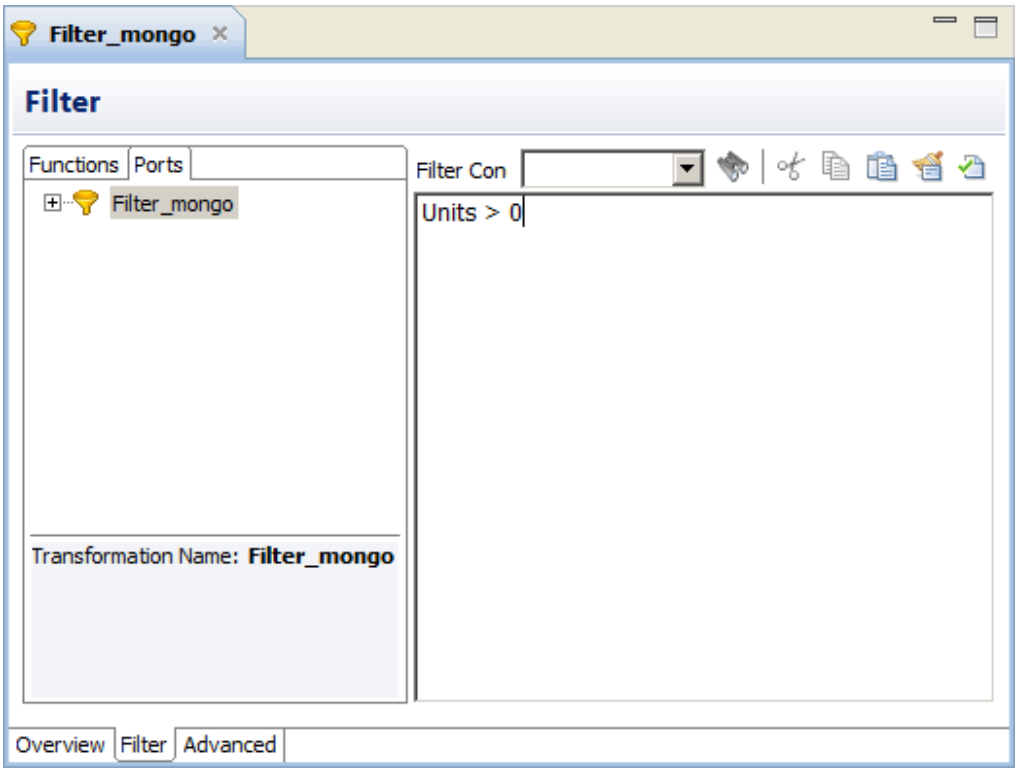
Import the collection Store_Catalog as an ODBC data object.

The following figure shows the data object created from the collection:



Filter transformation

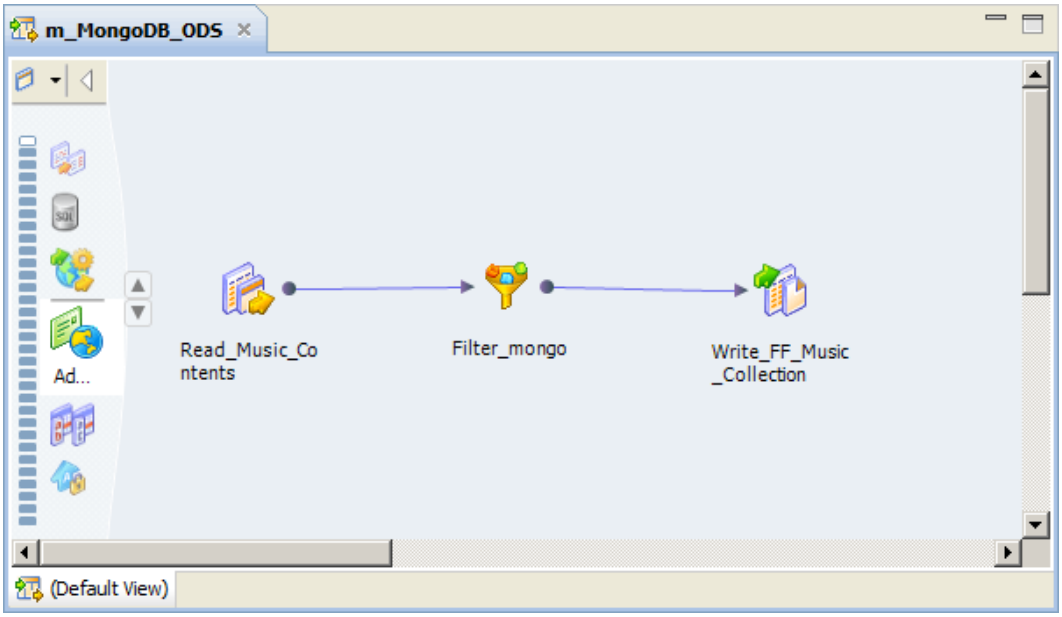
The filter transformation applies a filter on the Units field and writes those records that have one or more units in the Units field.



Flat file data object

The flat file data objec,t ff_Music_Collection, in the write mode, contains the same columns as in the MongoDB ODBC Source Definition.

The following figure shows the mapping:



APPENDIX A

Datatype Reference

This appendix includes the following topic:

MongoDB, ODBC, and Transformation Datatypes

When you define the schema in the Informatica MongoDB ODBC driver, you can view the ODBC datatypes and edit the datatypes. When you import a MongoDB collection as a data object, the transformation datatypes corresponding to the ODBC datatypes appear in the Developer tool .

The Informatica MongoDB ODBC driver reads MongoDB data and converts the MongoDB datatypes to ODBC datatypes. The Data Integration Service converts the ODBC datatypes to transformation datatypes.

The following table lists the MongoDB datatypes and the corresponding ODBC and transformation datatypes:

MongoDB Datatypes	ODBC Datatypes	Transformation Datatypes	Range and Description
String	Varchar	String	1 to 104,857,600 characters
Boolean	Bit	String	Precision of 1
NumberLong	BigInt	Decimal	Precision 1 to 28 digits, scale 0 to 28
NumberInt	Int	Integer	Precision 10, scale 0
NumberDouble	Double	Double	Precision 15
BinData	Binary	Binary	1 to 104,857,600 bytes
Date	Timestamp	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to second)
jstOID	Varchar	String	1 to 104,857,600 characters

INDEX

I

R