‘

**Exporting Master Data from Master Data Services**

SQL Server Technical Article

**Writer:** Jim van de Erve

**Technical Reviewers:** Matthew Roche, Jaime Alva Bravo, Arunjeet Singh

**Published:** July 2012

**Applies to:** SQL Server 2012

**Summary:** This white paper describes how to export master data from Microsoft SQL Server Master Data Services (MDS) using a subscription view, and how to import the master data into an external system using SQL Server Integration Services (SSIS). The white paper provides a step-by-step sample for creating a subscription view and an SSIS package.

Copyright

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious.  No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2012 Microsoft. All rights reserved.

Contents

[Introduction 4](#_Toc328471550)

[Export Master Data into a Subscription View 4](#_Toc328471551)

[Creating a Subscription View 5](#_Toc328471552)

[What’s in a Subscription View 5](#_Toc328471553)

[Versioning a Subscription View 6](#_Toc328471554)

[Customize a Subscription View 6](#_Toc328471555)

[Import Master Data Using an SSIS Package 7](#_Toc328471556)

[Master Data Export and Import Sample 7](#_Toc328471557)

[Set Up the Master Data Export and Import Sample 10](#_Toc328471558)

[Create a Subscription View 10](#_Toc328471559)

[Create the Import Package 11](#_Toc328471560)

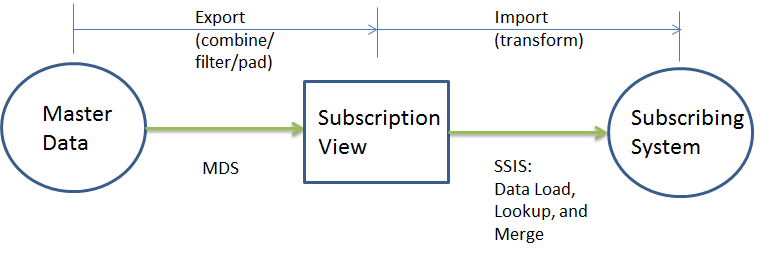
[Testing the Solution 14](#_Toc328471561)

[Conclusion 14](#_Toc328471562)

# Introduction

This document describes how to export master data from Microsoft SQL Server 2012 Master Data Services (MDS) into an operational or analytical system. This enables you to use your master data to meet your operational or analytical requirements in an automated way. This white paper focuses on the following processes to export and then import master data. It also provides a sample that demonstrates their usage.

* Exposing master data as a customizable subscription view created in MDS
* Importing master data from the subscription view into a SQL Server database table using a SQL Server Integration Services (SSIS) package (For more information, see [Import Master Data Using an SSIS Package](#_Import_Master_Data) later in this paper.)



You can find the Microsoft SQL Server 2012 [Evaluation version](http://www.microsoft.com/downloads/details.aspx?FamilyID=A74D1B60-6566-4551-B581-03337853B82B&amp;displaylang=de&displaylang=en) at the Microsoft Download Center.

# Export Master Data into a Subscription View

Subscription views are the primary way to synchronize the data in an external system with the master data in Master Data Services. MDS provides a straightforward way in the **Integration Management** menu of the Master Data Manager Web UI to create a subscription view of the master data. In doing so, you do not have to worry about the data management processes within MDS. For more information about exporting data from MDS, see [Exporting Data (Master Data Services)](http://msdn.microsoft.com/en-us/library/ee633741.aspx) (http://msdn.microsoft.com/library/ee633741.aspx) in SQL Server Books Online.

The advantages of exporting master data into subscription views are the following:

* Subscription views give you an automated means of exporting data from the MDS tables in a readable format. You can **view master data in the MDS Explorer**, but tables in the MDS database are not in a readable format.
* You can **create a view specific to the type of MDS data** that you want to see. A subscription view can be created in either an attribute view format or a hierarchy view format. MDS provides multiple formats for either an attribute view or a hierarchy view.
* You can **access MDS data in views** without being granted permissions to access the underlying base tables directly.

Subscription views are best suited to bulk, batch-oriented exports of master data. For event-based integration, use the change tracking notification feature instead. For more information about how to use change tracking notification, see [Initiate Actions Based on Attribute Value Changes](http://msdn.microsoft.com/library/ff486979.aspx) (http://msdn.microsoft.com/library/ff486979.aspx) in SQL Server Books Online. For programmatic access that enforces user security and application logic, use the MDS Web services API. For more information about working with the API, see [Developer’s Guide](http://msdn.microsoft.com/library/hh230994.aspx) (http://msdn.microsoft.com/library/hh230994.aspx) in SQL Server Books Online.

## Create a Subscription View

You can create a subscription view on any entity or derived hierarchy object within the MDS system. You can generate a subscription view from the Master Data Manager Web User Interface ([Integration Management](http://msdn.microsoft.com/library/ff487013.aspx) page) or by using the [ExportViewCreate](http://msdn.microsoft.com/library/microsoft.masterdataservices.services.servicecontracts.iservice.exportviewcreate.aspx) method in the MDS API. For more information about creating a subscription view by using the MDS samples, see [Create a Subscription View](#_Create__a) later in this paper.

To create a subscription view, you must have permission to access the Integration Management functional area, and you must be a model administrator.

## What’s in a Subscription View

MDS subscription views appear as ordinary views with the MDM schema in the MDS SQL Server database. The view formats include the following:

* The **Attributes view** format lists all leaf, consolidated, or collection members for an entity, and it includes columns for all attributes. Each domain-based attribute in a subscription view is identified by three columns: a name, a code, and an ID.
* The **Hierarchy view** format represents hierarchies as level-based columns, in which each column represents a level, or parent-child relationships, in which each row represents a relationship of one child to one parent. Hierarchy views provide relationship data for all types of relationships in MDS, including explicit and derived hierarchies, and collection members.

Subscription views contain system attributes that are not in the master data. System attributes provide context about the master data: version data, validation data (for attribute views), and data about when the master data was entered and by whom. Subscribing systems can use context data to determine how to update their records with master data. For example, when you import the data into a subscribing system, you can filter the master data by the following:

* The date and time that the data was last changed
* Validation status, to ensure that you are only using valid data
* Only those rows that have a certain type of data filled in, such as price
* Only the data that has been entered by a specific user

Caveats about subscription views include the following:

* Views do not enforce MDS application security permissions. There is no row-level or member-level security on subscription views. As a result, permissions to subscription views should only be granted to those who already have permissions to the MDS data.
* After you create a subscription view for an entity or hierarchy, changes to the associated model objects are not automatically reflected in the view. You will need to regenerate a subscription view in Master Data Manager to reflect changes to model objects. The Changed column on the Export page is updated to TRUE when model objects change. If this value is TRUE, you should regenerate the view.
* You cannot create a subscription view on a recursive derived hierarchy that has one or more nonrecursive levels beneath the recursion.
* There is no direct correlation between the order of columns in the entity of an MDS model and the order of columns in the subscription view.
* Querying the MDS tables is not recommended because schema changes are likely in subsequent releases.

## Versioning a Subscription View

Every time that you open a view, the view is dynamically generated with data from the source, if updated data is available. If **Version** is selected in the Create Subscription View window of the MDM Web UI, the view applies only to the version specified. The view is only updated when it is opened if a value in that version of the master data is updated. If **Version Flag** is selected, the view is updated according to the value of the version flag, as set in the version flag’s drop-down list.

If **Version Flag** is set to **Current**, the view automatically displays new or updated values from the latest version of the master data. When you use the version flag, you do not have to regenerate the view for each new version of the master data. It is automatically regenerated if a new version is available when you open the view.

Use version flags if you expect the published version to change periodically. Use **Current** if you expect more frequent changes.

## Customize a Subscription View

You can customize the standard MDS subscription views to produce different output view formats. The subscription views that you create are located in the Views folder under the MDS database in SQL Server Management Studio. To access subscription views, you need to be logged in as a member of either the db\_datareader role on the MDS database or any role with Select permission on the views. You can customize a view as follows:

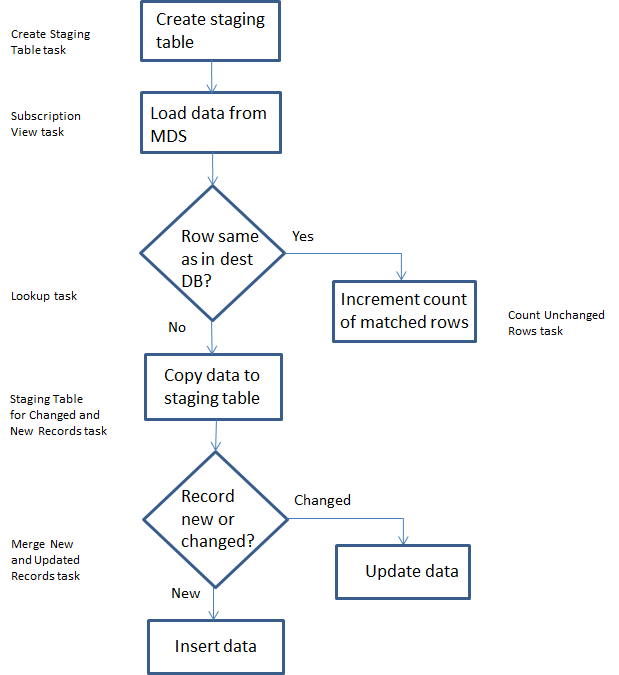
* **Filter a subscription view**. To limit the subscription view to the data that you want and improve query performance, filter the rows and columns of the standard view that is generated by the system. For more information and Transact-SQL code, see the Master Data Services blog post [Subscription Views Part 2: Combining and Customizing Views](http://blogs.msdn.com/b/mds/archive/2010/03/03/subscription-views-part-2-combining-and-customizing-views.aspx) (http://blogs.msdn.com/b/mds/archive/2010/03/03/subscription-views-part-2-combining-and-customizing-views.aspx) in the MSDN blogs.
* **Combine subscription views**. You can combine attribute and hierarchy data in one subscription view by building a custom view. Do so by using a JOIN on two standard subscription views. For more information and Transact-SQL code, see the Master Data Services blog post [Subscription Views—Part 2: Combining and Customizing Views](http://blogs.msdn.com/b/mds/archive/2010/03/03/subscription-views-part-2-combining-and-customizing-views.aspx) (http://blogs.msdn.com/b/mds/archive/2010/03/03/subscription-views-part-2-combining-and-customizing-views.aspx) in the MSDN blogs.
* **Create a padded-level subscription view**. You can fill in NULL columns in a level-based view of a ragged hierarchy. This type of ability is important because some subscribing systems require a fixed-level representation of a hierarchy. You can replace a NULL value at a level beneath the leaf level with the leaf-level member code and name by using the ISNULL function. For more information and Transact-SQL code, see the Master Data Services MSDN blog post [Subscription Views—Part 3: Padding Level-Based Views](http://blogs.msdn.com/b/mds/archive/2010/03/03/subscription-views-part-3-padding-level-based-views.aspx) (http://blogs.msdn.com/b/mds/archive/2010/03/03/subscription-views-part-3-padding-level-based-views.aspx) in the MSDN blogs.

# Import Master Data Using an SSIS Package

After you export master data from MDS into a subscription view, you can import the data into a subscribing system by using tools and techniques that access a standard SQL Server view. You can transform data from subscription views by using SQL queries in an ETL process. One of the most effective ways to do so is by using SQL Server Integration Services, which you can use to build enterprise-level data integration and data transformation solutions. The following sample shows you one way to do so.

## Master Data Export and Import Sample

This sample demonstrates a way to import master data from a subscription view into a SQL Server database table using an Integration Services package. This sample compares selected data in each row of the subscription view to the data in the destination database. It inserts data in the row if it is new, updates it if it is changed, and increments a counter if the data is the same. The workflow in the Master Data Export and Import sample includes creation of the subscription view and the following SSIS package.

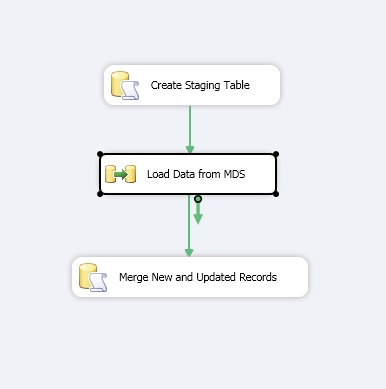


**Figure 1: Master Data Export and Import sample package flow**

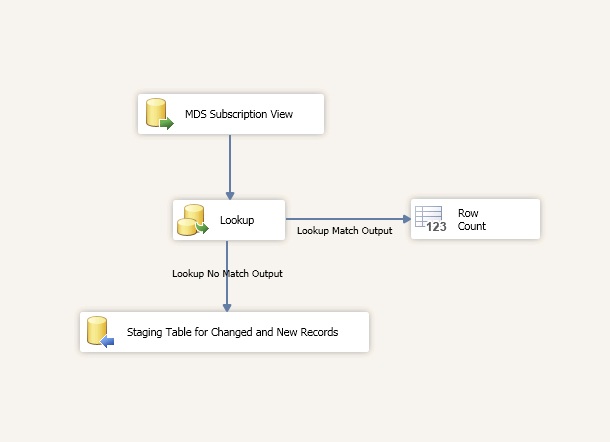
The sample works as follows:

1. In the Create Staging Table task, the package creates an MDS\_Merge\_Records staging table to store data returned from the lookup temporarily. If the MDS\_Merge\_Records table already exists, this step drops it.
2. In the MDS Subscription View task of the Load Data from MDS control flow, the package accesses the master data stored in three columns of the view, and then it passes the data to the lookup shape.
3. In the Lookup task, the package compares the data returned from the view with the data that already exists in the destination table. The package sends matched results to the Count Unchanged Rows task and unmatched results to the Staging Table for Changed and New Records task.
4. In the Count Unchanged Rows task, the count of matched rows is incremented if a row in the staging table matches a row in the destination table.
5. In the Staging Table for Changed and New Records task, all records in the staging table that have changed from the data currently in the destination table, and all new records, are saved in the staging table.
6. In the Merge New and Updated records task, all records in the staging table that have changed are updated in the destination table and all records that are new are inserted into the destination table.

The control flow and data flow of the SSIS package are as follows.



**Figure 2: Export and import control flow**



**Figure 3: Load data from MDS data flow**

## Set Up the Master Data Export and Import Sample

1. Install SQL Server 2012 on a test computer with Master Data Services and Integration Services installed and the Master Data Management Web application configured. This sample is designed to work on a single computer.
2. Deploy the Master Data Services sample model package product\_en.pkg. For more information, see the Jason’s Technical Topics blog post [Deploying MDS Samples in SQL Server 2012](http://blogs.msdn.com/b/jason_howell/archive/2011/12/15/deploying-mds-samples-in-sql-server-2012.aspx) (http://blogs.msdn.com/b/jason\_howell/archive/2011/12/15/deploying-mds-samples-in-sql-server-2012.aspx) in the MSDN blogs.
3. Create a destination SQL Server database named “Contoso”. In that database, create a table with the following configuration.

CREATE TABLE [dbo].[Product.Price](

[ProductName] [nvarchar](250) NULL,

[ProductCode] [nvarchar](250) NOT NULL,

[MSRP] [numeric](18, 2) NULL

)

1. In the Contoso database, create a staging table with the following configuration.

CREATE TABLE [dbo].[MDS\_Merge\_Records](

[Name] [nvarchar](250) NULL,

[Code] [nvarchar](250) NULL,

[MSRP] [numeric](34, 4) NULL

)

## Create a Subscription View

1. On the Master Data Management home page, select the model, and then click **Integration Management**.
2. From the menu bar, click **Create Views**.
3. On the Subscription Views page, click the **Add subscription view** icon.
4. In the Create Subscription View pane, in the **Subscription view name** box, type “ProductEntityLeafAttributes”.
5. From the Model list, select **product\_en**.
6. Click **Version option**, and then click **VERSION\_1**.
7. Click **Entity**, and then select **Product** from the list.
8. For Format, click **Leaf members**.
9. Click **Save**.
10. Open SQL Server Management Studio, move to the Views folder for your MDS database, and then verify that the view mdm.ProductEntityLeafAttributes has been created.

## Create the Import Package

1. Create a new Integration Services project as follows:
   1. On the Start menu, click **All Programs**, click **Microsoft SQL Server 2012**, and then click **SQL Server Data Tools**.
   2. On the Start page of Microsoft Visual Studio, and then click **New Project**. Under Business Intelligence in the Installed Templates pane, click **Integration Services**, and then in the middle pane, click **Integration Services Project**. Enter “Import Master Data” in the Name text box, and then click **OK**.
   3. In the Solution Explorer pane, rename the SSIS package (Package.dtsx) as ImportPackage.dtsx.
2. Create an MDS Load Data task as follows:
   1. With Control Flow selected, drag a Data Flow task onto the work space.
   2. Rename the task as “Load Data from MDS”.
   3. Double-click the **Load Data from MDS** task to open the Data Flow work space.
3. Create a source task as follows:
   1. Drag the Source Assistant onto the work space.
   2. In the **Add New Source** dialog box, with SQL Server selected, double-click **New**.
   3. In the **Connection Manager** dialog box, for Provider select **Microsoft OLE DB Provider for SQL Server** under Native OLE DB, and then click **OK**.
   4. Enter “localhost” (or “.”) for Server name.
   5. Leave **Use Windows Authentication** selected.
   6. With **Select** or **enter a database name** selected, select the name of your MDS database.
   7. Click **Test Connection**, ensure a successful test and click **OK**, and then click **OK** again.
   8. Rename the OLE DB Source task as “MDS Subscription View”.
   9. Double-click the MDS Subscription View task. In the **OLE DB Source Editor** dialog box, ensure that **Table or view** is selected for **Data Access mode**.
   10. For **Name of the table or the view**, select **[mdm].[ProductEntityLeafAttributes]**. Note that this is in the views section of the list.
   11. Click **Columns**, clear **Name** at the top, and then select **Name** (fifth row from the top), **Code**, and **MSRP**.
   12. Click **OK**.
4. Create a Lookup task as follows:
   1. Drag a Lookup task to the work space and position it under the MDS Subscription View task.
   2. Select the **MDS Subscription View** task, and then drag the left-pointing arrow from the MDS Subscription view to the Lookup task.
   3. Double-click the **Lookup** task.
   4. In the Lookup Transformation Editor, with **General** selected, select **No Cache** for the **Cache mode**, and select **OLE DB connection manager** for **Connection type**.
   5. For **Specify how to handle rows with no matching entries**, select **Redirect rows to no match output**.
   6. Select **Connection**. For **OLE DB connection manager**, click **New**.
   7. In the **Configure OLE DB Connection Manager** dialog box, click **New**.
   8. In the **Connection Manager** dialog box, for Provider select **Microsoft OLE DB Provider for SQL Server** under Native OLE DB, and then click **OK**. For Server name enter “Localhost”. For **Select** or **enter a database name**, select **Contoso**.
   9. Test the connection, and if successful, click **OK** twice.
   10. For Use a table or a view, select **[dbo].[Product.Price]**.
   11. Select **Columns**. Select **Name** at the top of the rightmost **Available Lookup Columns**, and then drag arrows from Name to ProductName, Code to ProductCode, and MSRP to MSRP. Leave Lookup Operation as <Add as new column> for all three mappings.
   12. Click **OK**.
5. Create a destination task as follows:
   1. Drag the Destination Assistant onto the work space.
   2. In the **Add New Destination** dialog box, with **SQL Server** selected, double-click **New**.
   3. In the **Connection Manager** dialog box, for **Provider** select **Microsoft OLE DB Provider for SQL Server** under **Native OLE DB**, and then click **OK**.
   4. Enter “localhost” for **Server name**.
   5. Leave **Use Windows Authentication** selected.
   6. With **Select** or **enter a database name** selected, select **Contoso**.
   7. Click **Test Connection**, ensure a successful test and click **OK**, and then click **OK** again.
   8. Rename the OLE DB Destination task as “Staging Table for Changed and New Records”.
   9. Select the **Lookup** task, and then drag the left-pointing arrow from the Lookup task to **Staging Table for Changed and New Records**.
   10. In the **Input Output Selection** dialog box, select **Lookup No Match Output** for **Output**. Click **OK**.
   11. Double-click the **Staging Table for Changed and New Records** task. Ensure that **Data access mode** is set to **Table** or **view**. For **Name of the table or the view**, select **MDS\_Merge\_Records**.
   12. Click **Mappings**, and verify that there are mappings from Name to Name, Code to Code, and MSRP to MSRP. Click **OK**.
6. Add a task to create the staging table as follows:
   1. Click **Control Flow**.
   2. Drag an Execute SQL task onto the work space above the Load Data from MDS task.
   3. Rename the task as “Create Staging Table”.
   4. Double-click the **Create Staging Table** task. In the Execute SQL Task Editor, ensure ConnectionType is set to **OLE DB**, and for **Connection** select **Localhost.Contoso**.
   5. Select **SQL Statement**, click the ellipsis, and then enter the following.

IF OBJECT\_ID ('[MDS\_Merge\_Records]') IS NOT NULL

DROP TABLE [MDS\_Merge\_Records]

GO

CREATE TABLE [MDS\_Merge\_Records] (

[Name] nvarchar(250),

[Code] nvarchar(250),

[MSRP] numeric(34,4)

)

* 1. Click **OK**, and then click **OK** again.
  2. Drag the arrow from the Create Staging Table task to the Load Data from MDS task.

1. Add a row count function as follows:
   1. Click **Data Flow**.
   2. Drag **Row Count** to the right of the Lookup task.
   3. Rename the Row Count task as “Count Unchanged Rows”.
   4. Select the **Lookup** task, and then drag the leftmost free arrow for **Lookup Match Output** from the Lookup task to the Count Unchanged Rows task.
   5. Click the Add Variable icon in the Variables pane. Name the variable “UnmatchedRowCount".
   6. Double-click the **Count Unchanged Rows** task. Select the **User::UnmatchedRowCount** variable.
   7. Click **OK**.
2. Merge records as follows:
   1. Click **Control Flow**.
   2. Drag an Execute SQL task to under the Load Data from MDS task.
   3. Right-click the task, click **Rename**, and then enter “Merge New and Updated Records”.
   4. Double-click the **Merge New and Updated Records** task. In Execute SQL Task Editor, ensure **ConnectionType** is set to **OLE DB**, and for **Connection** select **Localhost.Contoso**.
   5. Select **SQL Statement**, click the ellipsis, and then enter the following.

MERGE INTO [dbo].[Product.Price] AS Target

USING [dbo].[MDS\_Merge\_Records] AS MDS

ON MDS.Code = Target.ProductCode

WHEN MATCHED THEN

UPDATE SET Target.ProductName = MDS.NAME,

Target.MSRP = MDS.MSRP

WHEN NOT MATCHED THEN

INSERT (ProductName, ProductCode, MSRP) VALUES (MDS.Name, MDS.Code, MDS.MSRP);

* 1. Click **OK**, and then click **OK** again.
  2. Select the **Load Data from MDS** task, and then drag arrow from the Load Data from MDS task to the Merge New and Updated Records task.
  3. In Visual Studio, click **File** and then click **Save All**.

## Test the Solution

1. With ImportPackage.dtsx displayed, press F5.
2. When the run has completed, verify that there are green checks by all the tasks in ImportPackage.dtsx. Click **Data Flow**, and verify that there are green checks by all those tasks. Verify that 505 rows were sent from the MDS Subscription View task to the Lookup task, and from the Lookup task to the Staging Table for Changed and New Records task.
3. Open the SQL Server Management Studio, and connect to the Database Engine on the local server.
4. Right-click the dbo.Product.Price table in the Contoso database, and then click **Select Top 1000 Rows**. Verify that 505 rows are found in the table (if you have not changed the sample model).
5. Execute the following query: SELECT \* FROM [Contoso].[dbo].[Product.Price] WHERE ProductName = 'Bike Computer'. Verify that no records are returned.
6. Open Internet Explorer, and move to your Master Data Manager Web site.
7. With the sample model “product\_en” selected for Model, click **Explorer**.
8. Verify that the Product Entity is displayed. Click **Add Member**. In the Details pane, enter “Bike Computer” for Name, “BC-1112” for Code, and “79.99” for MSRP.
9. Open SQL Server Management Studio, move to the mdm.ProductEntityLeafAttributes view under the Views folder for your MDS database, and execute the following query: SELECT \* FROM [MDS].[mdm].[ProductEntityLeafAttributes] WHERE Name = 'Bike Computer'. Verify that a single row for the Bike Computer is displayed.
10. Open ImportPackage.dtsx in Visual Studio. Stop debugging if necessary. Press **F5**.
11. When the run has completed, verify that there are green checks by all the tasks in ImportPackage.dtsx. Click **Data Flow**, and verify that 506 rows were sent from the MDS Subscription View task to the Lookup task, the majority were sent to the Count Unchanged Rows task, and at least one row was sent from the Lookup task to the Staging Table for Changed and New Records task.
12. Open the SQL Server Management Studio.
13. Click the dbo.Product.Price table in the Contoso database, and then execute the following query: SELECT \* FROM [Contoso].[dbo].[Product.Price] WHERE ProductName = 'Bike Computer'.
14. Verify that a single record for the Bike Computer is returned.

# Conclusion

You can export master data from Master Data Services into an operational or analytic system using the functionality in MDS and SQL Server Integration Services. Subscription views in MDS give you an automated and customizable means of batch-exporting master data, using either the MDS Web UI or the MDS API to create the view. SSIS packages enable you to create a workflow that loads the master data from MDS; compares the master data to the destination SQL database; and inserts the master data, updates the destination data with the master data, or simply counts the matched rows. This is one example of an ETL process that MDS and SSIS enable you to create.

**For more information:**

<http://www.microsoft.com/sqlserver/>: SQL Server Web site

<http://technet.microsoft.com/en-us/sqlserver/>: SQL Server TechCenter

<http://msdn.microsoft.com/en-us/sqlserver/>: SQL Server DevCenter

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

* Are you rating it high due to having good examples, excellent screen shots, clear writing, or another reason?
* Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

[Send feedback](mailto:sqlfback@microsoft.com?subject=White%20Paper%20Feedback:%20Exporting%20Master%20Data%20from%20Master%20Data%20Services).