**DQS Cleansing Transform to the Data Flow**

The DQS Cleansing transformation uses Data Quality Services (DQS) to correct data from a connected data source, by applying approved rules that were created for the connected data source or a similar data source.

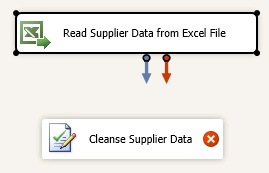
## DQS CLEANSING

Introduced in SQL Server 2012 was a component called Data Quality Services (DQS). This is not a feature of Integration Services, but it is very much connected to the data cleansing processes within SSIS. In fact, there is a data transformation called the DQS Cleansing Task. This task connects to DQS, enabling you to connect incoming Data Flow data and perform data cleansing operations.  
Because this Tutorial focuses on SSIS, a full DQS tutorial is not included; however, this section provides a brief overview of DQS and highlights a few data quality examples.

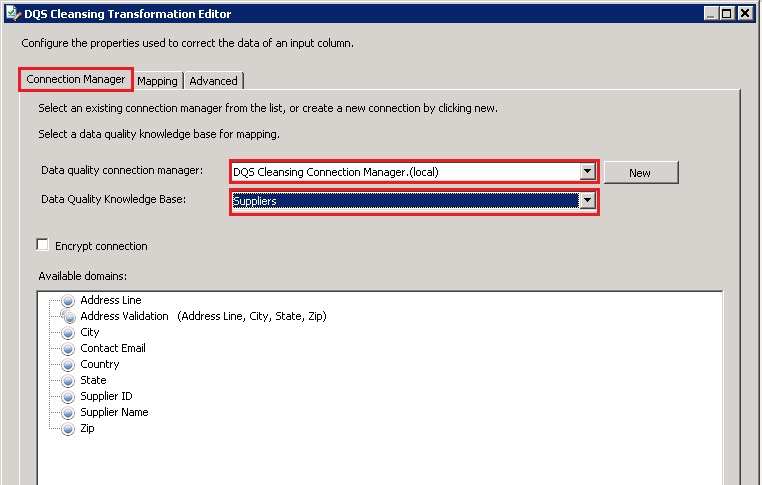
## DATA QUALITY SERVICES

The workflow to use DQS within SSIS requires a few preparatory steps. These need to be performed within the DQS client tool connected to a DQS service. The DQS client is available in the SQL Server 2014 Programs folder (from the Start button). There is a 32-bit version and a 64-bit version. In order to use Data Quality Services, you must have installed it during the SQL Server setup and run the configuration executable, called DQSInstaller.exe. The full setup instructions can be found on MSDN, Http://msdn.microsoft.com/enus/ library/gg492277(v=SQL.120).aspx**.**

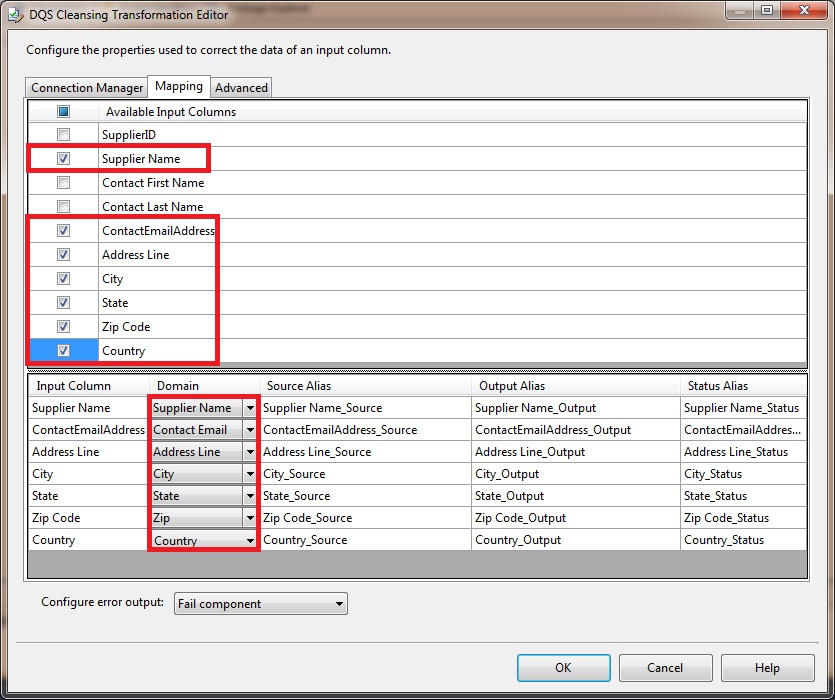
1. Right-click **DQS Cleansing** in the **Data Flow** tab, and click **Rename**. Type **Cleanse Supplier Data**, and press **ENTER**.
2. Select **Read Supplier Data from Excel File**; drag the blue connector to **Cleanse Supplier Data**. The components are now connected.



1. Double-click **Cleanse Supplier Data**.
2. In the **DQS Cleansing Transformation Editor**, click **New** next to the **Data Quality Connection Manager drop-down list**.
3. In the **DQS Cleansing Connection Manager** dialog box, type **(local)** or **period** (.) to connect to the local server. This lesson assumes that you have DQS installed on a local server.
4. Click **Test Connection** to test the connection to DQS server.
5. Click **OK** to close the dialog box.
6. Select **Suppliers** for the **Data Quality Knowledge Base**.



1. Switch to the **Mapping** tab at the top.
2. From **Available Input Columns**, select **Supplier Name**, **ContactEmailAddress**, **Address Line**, **City**, **State**, **Country**, and **Zip Code** by selecting the check boxes.



1. In the bottom pane, map these columns by using drop-down lists in the **Domain** column:

| **Column** | **Domain** |
| --- | --- |
| Supplier Name | Supplier Name |
| ContactEmailAddress | Contact Email |
| Address Line | Address Line |
| City | City |
| State | State |
| Country | Country |
| Zip Code | Zip |

1. Click **OK** to close the **DQS Cleansing Transformation Editor** dialog box.

Once you pull up the client and connect to the server, you will be in the DQS main screen, shown in Below screen shot.



You can perform three primary tasks with DQS:

**Knowledge Base Management** is how you define the data cleansing rules and policies.

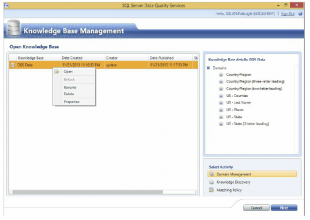
**Data Quality Projects** are for applying the data quality definitions (from the knowledge base) against real data. We will not be considering projects in this Advanced Data Cleansing in SSIS Topic ; instead, you will see how to use the SSIS DQS Cleansing Task to apply the definitions.

**Administration** is about configuring and monitoring the server and external connections.

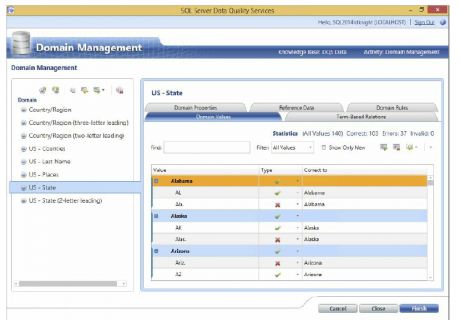
To begin the process of cleansing data with DQS, you need to perform two primary steps within the Knowledge Base Management pane:

1. Create a DQS Knowledge Base (DQS KB). A DQS KB is a grouping of related data quality definitions and rules (called domains) that are defined up front. These definitions and rules are applied against data with various outcomes (such as corrections, exceptions, etc.). For example, a DQS KB could be a set of domains that relate to address cleansing, or a grouping of valid purchase order code rules and code relationships within your company.
2. Define DQS domains and composite domains. A DQS domain is a targeted definition of cleansing and validation properties for a given data point. For example, a domain could be “Country” and contain the logic on how to process values that relate to countries around the world. The value mapping and rules define what names are valid and how abbreviations map to which countries.

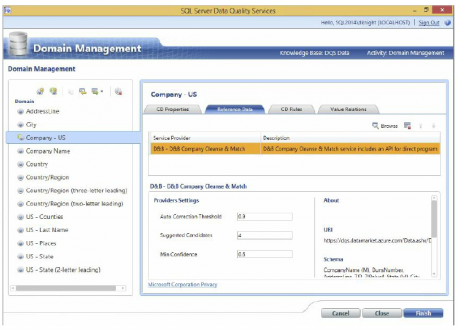
When you select the Open knowledge base option, you are presented with a list of KBs that you have worked with. The built-in KB included with DQS, DQS Data, contains several predefined domains and rules, and connections to external data. Below screen shot shows the right-click context menu, which enables you to open the KB and see the definition details.



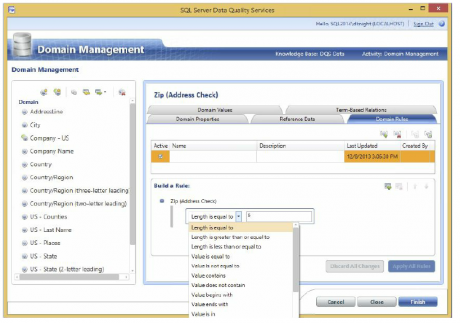
Knowledge bases are about domains, which are the building blocks of DQS. A domain defines what the DQS engine should do with data it receives: Is it valid? Does it need to be corrected? Should it look at external services to cleanse the data? For example, Below screen shot highlights the Domain Values tab of the State domain. It shows how values are cleansed and which values should be grouped. In this example, it lists state abbreviations and names and the Correct To value.



In the next example, a composite domain is selected. A composite domain is just what it sounds like: a group of domains. In this case, the domains involve companies, based on the business name, city, country, and state. Below screen shot shows the partial configuration of a composite domain. In this case, there is an external web service reference called “D&B – D&B Company Cleanse & Match” through which data will be processed. There are many sources you could connect to, such as Melissa Data for address cleansing (WWW.MELISSADATA.COM) or a host of data sources from the Windows Azure Data Marketplace (Https://datamarket.azure.com). There are a variety of premium data sources available here. Some can be free on a trial basis, while others have a paid subscription–based fee.



Domains can also contain rules that validate the data as it is processed through DQS. In the example in Below screen shot, the Zip (Address Check) field is validated so that the length is equal to 6. You can also see some of the other options in the list. Multiple rules can be applied with logical **AND** or **OR** conditions. If a data element fails the rules, it is marked as bad data during the processing.

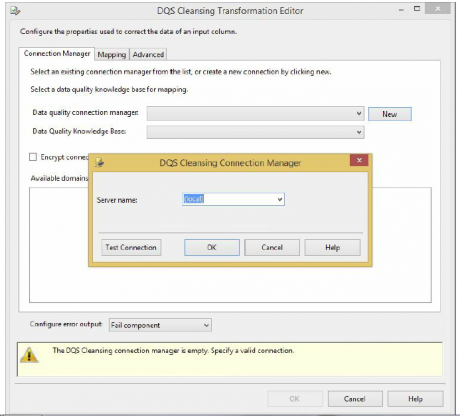


Other common rules include range rules to check that numeric data values fall within a given range and value lists to make sure that the data coming in meets specific requirements.

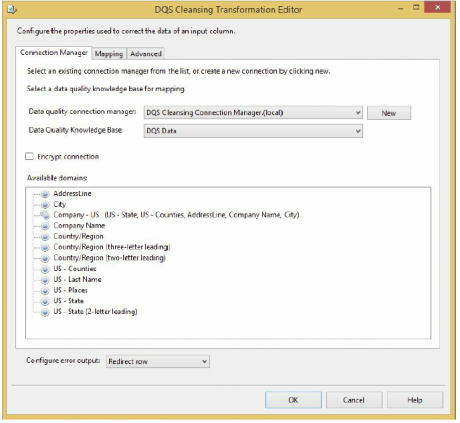
As shown in these few examples, DQS can serve as a powerful data quality engine for your organization. In addition to the common data validation and cleansing operations, you can apply a host of custom rules, matching criteria, and external links

The next step is to connect the source (or other transformation) to a DQS Cleansing Transformation and edit the task.

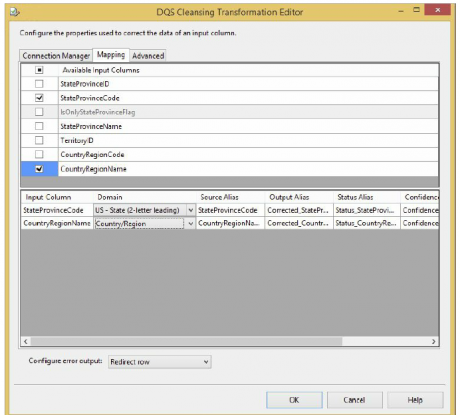
Figure below shows the Connection Manager tab of the DQS Cleansing Transformation. You need to connect to the DQS server and choose the knowledge base that you will be using for your source input data within SSIS.



In this example, the source data contains states/provinces and countries, so you will use the built-in DQS Data KB to connect the states and countries. To see the list of domains, choose DQS Data from the Data Quality Knowledge Base dropdown, as shown in below screen shot.

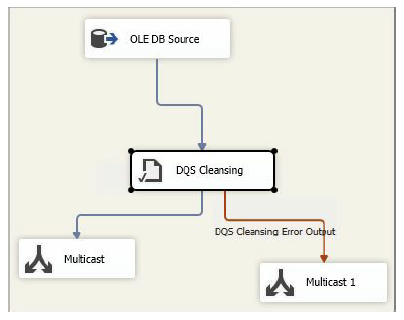


The Mapping tab contains the list of input columns that can be used against the KB domains. In below figure, both the StateProvinceCode and the CountryRegionName columns are selected in the input column list and matched to the US – State (2-letter leading) and Country/Region domains in the Domain dropdown.

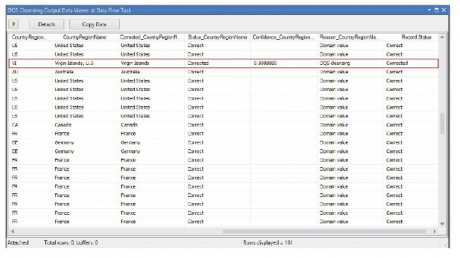


You are also able to redirect the errors to the error output for the rows that do not meet the domain criteria and rules, using the Configure Error Output dropdown at the bottom of the DQS editor.

Below screen shot shows the simple Data Flow with a couple of Multicast Transformations so that the data can be viewed (for demo purposes).



In addition to mapping the inputs to the DQS domain, the DQS Cleansing Transformation also provides additional data in the output of the transformation. Figure 10-30 shows a Data Viewer with the output rows and columns resulting from the DQS cleansing process.



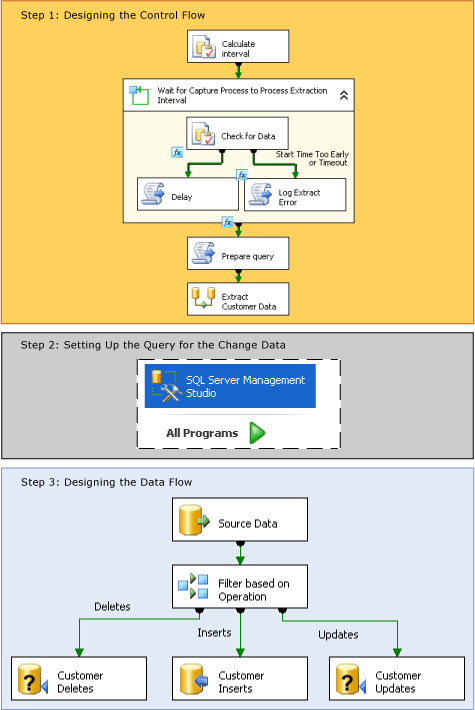
In this example, note the highlighted row indicating where the country was corrected and standardized to the DQS domain definition. Besides the original and corrected value returned, you can also see a reason code, as well as a confidence level on the correction. These are similar to the Fuzzy Component outputs shown earlier, except you have much more control and flexibility in terms of how you define your data cleansing process within DQS and apply it in SSIS.

# Change Data Capture

## What is Change Data Capture?

Source tables change over time. A data mart or data warehouse that is based on those tables needs to reflect these changes. However, a process that periodically copies a snapshot of the entire source consumes too much time and resources. Alternate approaches that include timestamp columns, triggers, or complex queries often hurt performance and increase complexity. What is needed is a reliable stream of change data that is structured so that it can easily be applied by consumers to target representations of the data. Change data capture in SQL Server provides this solution.

Once an administrator has enabled change data capture on the database, you can create a package that performs an incremental load of the change data. The following diagram shows the steps for creating such a package that performs an incremental load from a single table:



As shown in the previous diagram, creating a package that performs an incremental load of changed data involves the following steps:

**Step 1: Designing the Control Flow**  
In the control flow in the package, the following tasks need to be defined:

* Calculate the starting and ending **datetime** values for the interval of changes to the source data that you want to retrieve.

To calculate these values, use an Execute SQL task or Integration Services expressions with **datetime** functions. You then store these endpoints in package variables for use later in the package.

**For more information:** Specify an Interval of Change Data

* Determine whether the change data for the selected interval is ready. This step is necessary because the asynchronous capture process might not yet have reached the selected endpoint.

To determine whether the data is ready, start with a For Loop container to delay execution, if necessary, until the change data for the selected interval is ready. Inside the loop container, use an Execute SQL task to query the time mapping tables maintained by change data capture. Then, use a Script task that calls the **Thread.Sleep** method, or another Execute SQL task with a **WAITFOR** statement, to delay the execution of the package temporarily, if necessary. Optionally, use another Script task to log an error condition or a timeout.

**For more information:** Determine Whether the Change Data Is Ready

* Prepare the query string that will be used to query for the change data.

Use a Script task or an Execute SQL task to assemble the SQL statement that will be used to query for changes.

**For more information:** Prepare to Query for the Change Data

**Step 2: Setting Up the Query for Change Data**  
Create the table-valued function that will query for the data.

Use SQL Server Management Studio to develop and save the query.

**For more information:** Retrieve and Understand the Change Data

**Step 3: Designing the Data Flow**  
In the data flow of the package, the following tasks need to be defined:

* Retrieve the change data from the change tables.

To retrieve the data, use a source component to query the change tables for the changes that fall within the selected interval. The source calls a Transact-SQL table-valued function that you must have previously created.

**For more information:** Retrieve and Understand the Change Data

* Split the changes into inserts, updates, and deletes for processing.

To split the changes, use a Conditional Split transformation to direct inserts, updates, and deletes to different outputs for appropriate processing.

**For more information:** Process Inserts, Updates, and Deletes

* Apply the inserts, deletes, and updates to the destination.

To apply the changes to the destination, use a destination component to apply the inserts to the destination. Also, use OLE DB Command transformations with parameterized UPDATE and DELETE statements to apply updates and deletes to the destination. You can also apply updates and deletes by using destination components to save the rows to temporary tables. Then, use Execute SQL tasks to perform bulk update and bulk delete operations against the destination from the temporary tables.