

# Data Quality Solutions

SQL Server Technical Article

**Writers**: Elizabeth Vitt, Intellimentum  
 Hitachi Consulting

**Technical Reviewers**: Donald Farmer, Microsoft Corporation  
 Stacia Misner, Hitachi Consulting

Published: July 2006

Applies To: SQL Server 2005

Summary: This white paper describes how application developers can incorporate data quality into their Microsoft SQL Server 2005 Integration Services solutions.

### Copyright

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

 2006 Microsoft Corporation. All rights reserved.

Microsoft, Visual C#, and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

### Table of Contents

[Introduction 4](#_Toc139355327)

[Data Quality Strategy 4](#_Toc139355328)

[SSIS Data Integration Solutions 5](#_Toc139355329)

[Profiling 6](#_Toc139355330)

[SSIS profiling scenario 8](#_Toc139355331)

[Cleansing 12](#_Toc139355332)

[SSIS cleansing scenario 13](#_Toc139355333)

[Auditing 17](#_Toc139355334)

[SSIS auditing scenario 18](#_Toc139355335)

[SSIS Data Quality Partners 21](#_Toc139355336)

[Conclusion 21](#_Toc139355337)

[About the Authors 22](#_Toc139355338)

#### Introduction

The quality of the data that is used by a business is a measure of how well its organizational data practices satisfy business, technical, and regulatory standards. Organizations with high data quality use data as a valuable competitive asset to increase efficiency, enhance customer service, and drive profitability. Alternatively, organizations with poor data quality spend time working with conflicting reports and flawed business plans, resulting in erroneous decisions that are made with outdated, inconsistent, and invalid data.

To avoid the consequences of poor data quality, many organizations implement source system controls to ensure that their data satisfies quality standards at its point of origin. When properly implemented, source quality controls can effectively prevent the proliferation of invalid data. However, source system quality controls alone cannot enforce data quality. They cannot, for example, ensure that data quality is maintained throughout the data life cycle, especially when multiple data sources with varying levels of cleanliness are combined in downstream data integration processes. To address this potential problem, downstream applications must also include steps to ensure that data quality is preserved, if not enhanced, after data leaves the source system.

To meet this challenge, many successful enterprises adopt a flexible data quality strategy that incorporates data quality components directly into their data integration architecture. Successful application of this strategy requires a data integration platform that can implement a broad range of generic and specific business rules and also adhere to a variety of data quality standards.

With the release of Microsoft® SQL Server™ 2005 Integration Services (SSIS), a full-featured data integration engine and rich development environment for building comprehensive solutions, Microsoft has introduced a data quality strategy to help you easily incorporate data quality components into your SSIS solutions. SSIS addresses three primary data quality tasks: profiling, cleansing, and auditing. Using SSIS 2005, an enterprise can successfully create robust and reliable data integration solutions that reduce integration costs, mitigate data quality risks, and create valuable data assets. In addition, to complement the data cleansing functionality offered by SSIS, Microsoft works with providers who have built specialized data quality solutions on the SSIS platform. For more information on Microsoft data quality partners, see [SSIS Data Quality Partners](#_SSIS_Data_Quality_Partners) later in this white paper.

#### Data Quality Strategy

The primary goal of a data integration solution is to assemble data from one or more data sources. As you bring data together, you are likely to find a broad range of data quality issues that require attention. For example, you may discover missing customer profile information, such as blank phone numbers or addresses. You may also uncover incorrect data, such as a customer who lives in the city of Australia.

As the number of data sources that must be integrated increases, data quality issues also increase in number and complexity. Perhaps one of the most challenging data quality issues is duplication of data. Duplication arises when there are conflicting representations of the same entity across source systems. For example, the same customer may be stored in the CRM, order entry, and customer support systems and have different customer numbers and profile information. This conflicting data makes it difficult to recognize identical customers, especially across thousands or millions of source records.

The successful handling of data quality issues like these requires a flexible data quality strategy that can be applied to a broad range of issues and integration scenarios. This flexibility is precisely the goal of the data quality strategy for SSIS solutions. The strategy involves three key tasks of data quality solutions: profiling, cleansing, and auditing.

* **Profiling** – As the first line of defense for your data integration solution, profiling data helps you proactively assess whether a source data extract meets the baseline quality standards of your solution. Properly profiling your data saves execution time because you identify issues that require immediate attention at the outset and avoid unnecessarily processing unacceptable source data sets. Data profiling becomes even more important when you are working with mainframe file extracts or unaudited data sources that do not have referential integrity or quality controls.
* **Cleansing** – After a data set successfully satisfies profiling standards, it still requires data cleansing to ensure that all business and schema rules are properly met. Successful data cleansing requires the use of flexible, efficient, and intelligent techniques to handle complex quality issues hidden in the depths of large data sets.
* **Auditing** – Auditing is perhaps the most important aspect of data quality. Auditing provides a history of all of the data cleansing operations performed in your integration solution. With auditing, you can track and score the overall data quality of your integration solution to evaluate how well data processing has met desired business, technical, and regulatory standards.

#### SSIS Data Integration Solutions

SQL Server Integration Services (SSIS) 2005, a successor to SQL Server 2000 Data Transformation Services (DTS), is an enterprise data integration platform for a new generation of integration solutions that must quickly integrate and cleanse large volumes of data while preserving and enhancing data quality.

Behind the scenes, the high-performance architecture of SSIS includes two powerful engines:

* **Parallel run-time engine** - The run-time engine coordinates the flow of control between tasks within SSIS.
* **Fast, efficient data flow engine** – The data flow engine extracts data from one or more data sources, performs any necessary transformations on the extracted data, and then delivers that data to one or more destinations. To maximize efficiency, the data flow engine takes advantage of in-memory processing to eliminate the overhead normally added by physically copying and staging data in tables at different points in the data integration process. By manipulating the data in memory as it is transferred from source to destination, the data flow engine reduces the number of manual steps required to stage data, saves processing time, and helps resolve integration challenges within shorter data processing windows.

To support a broad range of data requirements for data quality solutions, SSIS provides the following data integration capabilities:

* **Flexible workflow –** SSIS provides you with the ability to use quality indicators to drive the workflow of your data operations. You can assign tasks that proceed on the success or failure of the previous task. Or, you can create more complex conditions that examine data quality indicators before moving on to the next task.
* **Robust data cleansing –** To ensure that data is integrated in a high quality manner, SSIS provides data cleansing tools that can be leveraged to address common data integration challenges, such as resolving inconsistencies, reassigning data values, and handling data duplicates.
* **Comprehensive logging –** Toaudit data processing activities, SSIS provides the ability to log execution details to a variety of data providers thereby helping you to create a data audit trail for your integration solution.

#### Profiling

When you profile data before integration, you proactively assess whether a source data extract satisfies the baseline quality standards of your data integration solution. By establishing and enforcing baseline quality indicators, you determine whether it is worthwhile to execute your data integration processes using data in its current state.

The first step is to choose baseline indicators. Baseline indicators are metrics or conditions that assess the quality of your entire source data set rather than focusing on the data quality issues of a specific record.

A good rule of thumb for selecting indicators is to identify any condition or issue that would cause you to stop integration processing and force you to start again from the beginning.

Examples of baseline quality indicators include:

* **Total number of source records**. For each source data set, you can check the source record count to confirm that the source file or query returns a record count that is > 0. If you know the expected number of source records, you can compare the count to that number or to the counts of previous data loads.
* **Percent of missing column values**. For any given data source, it is quite common for some fields to have missing values. Missing values may be represented by NULL or a dummy value such as NA, Unknown, or 9999. If you have logic to populate these fields, you can do so during data cleansing. However, a large percentage of records that have missing or NULL values for important columns may be a sign of something more serious, such as a failure in the source system.
* **Percent of referential integrity errors**. The degree of testing for referential integrity errors depends on your application. In many applications, referential integrity errors are simply a fact of life and require data cleansing. In other applications, incoming referential integrity errors may not be allowed, especially if incoming source data contains identical unique identifiers.
* **Missing file indicator**. If your integration solutions use source data files, you want to ensure that the files are present in their correct locations before executing the solution. This is especially true if you are using a collection of interdependent files that must all be in synchronization before you begin processing.
* **Source system schema change indicator**. Changes to the source system schema always require special consideration as they will probably require changes to your downstream applications. While it is customary and recommended that source system changes be identified and discussed ahead of time, sometimes extenuating circumstances arise and changes occur without notification. As a safety measure, consider including profiling logic to detect source schema modifications such as data type changes. A benefit of including this check in the profiling task is the identification of any schema changes before data integration starts so that you can avoid troubleshooting problems one at a time after errors arise during execution.
* **Flagging suspicious data values.** In some scenarios, profiling may include comparing data values to flag questionable records. For example, upon profiling your source data, you may discover that you have several twelve-year-old head of households with PhDs. While the values themselves do not violate any schema constraints (such as NULL or referential integrity constraints), the records appear to be suspicious. Perhaps they are the result of data entry errors in the source system. To flag suspicious data, you can incorporate data mining in your data profiling operations to analyze and compare data values to a set of cleansed data. Within SSIS, this type of profiling is made easier because data mining components can be integrated in your normal data profiling operations.

Once you have selected baseline indicators, the next step is to assign each indicator an upper and/or lower bound to determine whether the incoming data meets or violates a particular standard. Keep in mind that metrics-based indicators require you to make an informed judgment about acceptable ranges for your specific application.

The final step of profiling data is to decide the action to take when one or more threshold values are violated. You might consider taking one of the following actions:

* Send e-mail messages with a list of issues to source system owners so they can investigate and fix problems.
* If source system schema updates require changes to your data integration solution, incorporate all updates at once rather than troubleshoot schema problems one at a time.
* When an issue exists, but the indicator value is below your defined threshold, don’t take immediate action. Instead, track the trend of the indicator value over time, e-mail this information to stakeholders, and take preventative action to circumvent serious issues.
* Tweak threshold values by increasing or decreasing the threshold as needed.

Regardless of which action you choose, the use of clearly defined baselines quality indicators creates an effective front line of defense for your integration solutions.

##### SSIS profiling scenario

To understand how profiling can be used to proactively protect data quality, consider the following scenario in which you migrate sales data from a mainframe system using a data file that contains a list of sales order transactions.

For this specific source file, you have identified the following four baseline quality indicators. All indicator values must be within established bounds before subsequent processing can proceed as described here:

* The record count of the source file must be greater than 0.
* NULL customer names are not allowed.
* All order dates must be less than or equal to the current date.
* No duplicate unique identifiers are allowed.

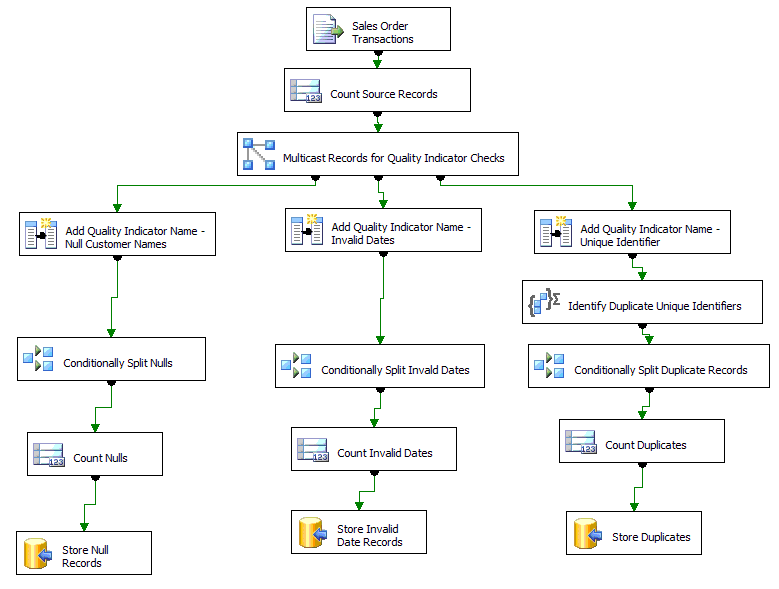
To satisfy these requirements, SSIS provides several functions to support the profiling of your source data sets according to custom business rules.

The Data Flow task in particular provides three useful transformations—Row Count, Multicast, and Conditional Split—that can be used together to gather data quality information.

* The **Row Count** transformation allows you to count records at any point in the SSIS data flow and store that record count as an SSIS variable that can then be used for further processing.
* The **Multicast** transformation permits you to use a single data set for a variety of data quality checks without having to repeatedly read data from the source.
* The **Conditional Split** transformation enables you to filter the SSIS data flow based on specific conditions for each quality indicator, such as NULL data values, duplicate identifiers, or invalid data ranges.

Figure 1 illustrates how these three data flow components can be used to inspect the sales order transaction data source.

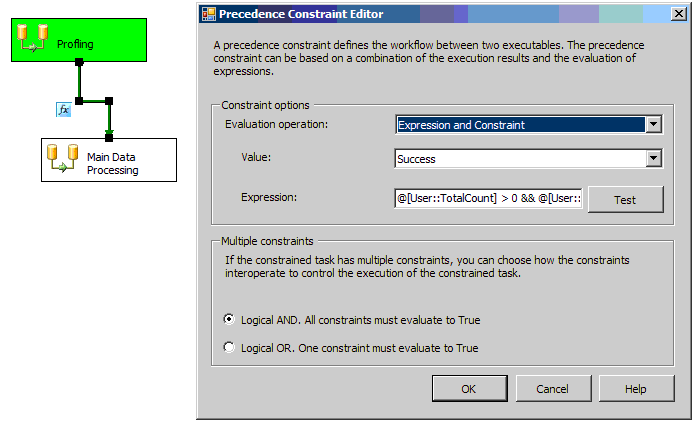
Figure 1   Profiling Data Flow

The SSIS solution uses the following steps to inspect data in the data flow:

1. SSIS first reads all records from the source file. When you configure the Flat File Source component, be sure to retain NULLS from the file in the data flow.
2. A Rowcount transformation is used to count the total number of source file records. The Rowcount is stored as a SSIS variable called TotalCount.
3. A Multicast transformation makes three copies of the source data so that it can filter and aggregate differently for each quality indicator. As a result, three new branches are created in the data flow.
4. For each branch, a Derived Column transformation adds the indicator name to a new column in the data flow. The indicator name displays the type of data quality check that each branch performs. The first branch checks for Null Customer Names. The second branch checks for Invalid Dates. The third branch checks for Unique Identifiers. This name is referenced in the final step of each branch. That step inserts the results of the quality check into a database table, so that you can track the results of each data quality check.
5. For the Null Customer Name and Invalid Date branches, the following checks are performed:
6. Conditional Split transformations filter the record sets so that only invalid data remains in the data flow.
7. Row Count transformations count the invalid records and store the counts as separate global variables.
8. The invalid records are then stored in a database table for more detailed investigation later.
9. In the Unique Identifier branch, the following checks are performed:
10. Duplicate identifiers are discovered by using an Aggregate transformation which groups data by the unique identifier, which in this example is the customer code. In addition, the Aggregate transformation contains a Count All operation to count the total number of instances for each customer code. A unique customer code should only have a count of 1.
11. A Conditional Split transformation filters the record set so only invalid data remains. In this scenario, invalid records are duplicate customer codes having a count greater than 1.
12. A Row Count transformation counts the number of customer codes that are duplicated.
13. The duplicate unique identifiers are then stored in a database table for more detailed investigation later.

The next step in the solution uses SSIS precedence expressions to control the workflow of the package.

Figure 2 presents the workflow components of the SSIS Profiling solution.



**Figure 2   Profiling Workflow**

Two Data Flow tasks are displayed—the Profiling task and the Main Data Processing task.

* The Profiling task contains all profiling logic described in the previous section.
* The Main Data Processing task contains all logic required to load the file into its destination.

Connecting the two tasks is done by a special type of control flow construct that uses a custom expression with a Success precedence constraint to specify when the Main Data Processing task executes.

In this scenario, the expression has been configured to enforce the combination of the four quality indicators:

@[User::TotalCount] > 0 && @[User::NullCount] == 0 && @[User::DuplicateKeyCount] == 0 && @[User::InvalidDateCount] == 0

In Figure 2, note that the Profiling task is Green to indicate successful execution; however, the Main Data Processing task has not yet run. This behavior is expected because the Profiling task uncovered issues that violate the baseline indicators.

To enhance this example, you can make one or more of the following changes to the package:

* Add an e-mail task to alert source system owners of the data issues and point them to the invalid record table where they can find more information.
* Depending on your source quality indicators, instead of profiling an entire file, you may want to limit profiling to a partial data set by using random sampling. SSIS provides two transformations to accomplish this—the Percentage Sampling transformation and the Row Sampling transformation.
* If you want to tweak threshold values on a case-by-case basis, you can change the thresholds from static values to variables. You can then assign variable values for each specific package execution either by providing values manually at runtime or by retrieving them from an SSIS configuration table.
* Perform additional validation by using an Analysis Services data mining model to flag suspicious data values for further inspection.

#### Cleansing

After thoroughly profiling your source data quality, use data cleansing to ensure that your integration solution processes data according to the highest quality standards. On a column-by-column and record-by-record basis, data cleansing enforces the business and schema rules of your application for each source record.

When a rule is violated, you have three choices:

• Fix the data issue by using business logic in your solution.

• Discard the record and continue processing.

• Stop processing.

While these choices may seem simple, selecting the right techniques for each data issue requires that you weigh all options.

Fixing a data issue is typically the most logical choice. The specific fix that you implement depends on the nature and complexity of the data quality issue. Following are some examples of common data issues:

* **Missing data** **values** - When data is missing, you may be able to retrieve the data from another data source. For example, if you are missing an employee name but have the employee’s e-mail address, you may be able to look up the name from the Human Resources (HR) reference data source. If you do not have a data source, you may have business rules that determine how to derive the missing data.
* **Data duplicates** – Data duplicates are easy to spot when a unique identifier exists. When no unique identifier exists, it becomes more difficult to spot them. To overcome this challenge, consider using fuzzy logic to perform imprecise data matches which can eliminate data duplicates.
* **Inconsistent data formats -** In some cases, data is not in a format that can be integrated with other sources. For example, if you have a multi-value address field to combine with a data source that stores normalized address data, you need to develop logic to extract the street address, city, state/province, country, and zip code.

While fixing data can be very effective, you may not always have enough information to do so. For example, if you are missing a customer’s mobile phone number, you might not have the data source or logic to populate the missing value. When data issues cannot be fixed, the record is usually classified as an error record and is discarded from processing. When designing a solution, the decision to discard records requires some thought since the error records are typically mixed in with clean records. In addition, you want to ensure that the discarded error records are not lost forever. Rather, they should be stored in an error table or log for additional investigation.

When processing stops, typically a data quality error has been encountered that cannot or should not be fixed in the solution. This type of issue is actually a great candidate for a baseline quality indicator. Hopefully, through data profiling, you can discover these types of issues before you go through a lot of unnecessary data processing.

There is a fourth action to consider and that is to take no action and leave the data as-is. This typically means that you really do not consider the violation important enough to require action. For this situation, set up data auditing to monitor your data processes so you can analyze data cleansing operations over time. After monitoring the solution for a while, you may decide to take action at a later date.

You are likely to use a combination of all options, with the majority of your efforts focused on fixing data quality issues.

##### SSIS cleansing scenario

To understand how data cleansing can be used to enforce data quality in your integration solutions, consider the following scenario. You want to refresh a customer table with customer profile data that is stored in a file. The file has the following data quality issues that you need to address:

* The source file is tab-delimited with the exception of a multi-value address column. An example record for the address column is as follows:

One Microsoft Way, Redmond, WA 98007, USA

To load the data, you must first split the address source column by using comma delimiters and then split the remaining state and zip code that are delimited by a blank space.

* Within the file, a customer may be listed multiple times with different contact information. To ensure that you only retrieve the most recent customer profile data, a business rule states that you must always retrieve the latest profile data by using the maximum entry date.
* To determine whether the customer already exists in the database, perform a lookup to the existing customer table using the customer code unique identifier. If the lookup returns an exact match, the customer already exists. In this scenario, you update the record with the latest customer information.
* If the lookup on the customer code does not return an exact match, apply Fuzzy Lookup logic to find the best possible match by using the customer’s profile information.
* If the Fuzzy Lookup transformation does not produce an acceptable match, the customer record is considered to be new and is inserted with no matched customer code.
* If the Fuzzy Lookup transformation finds an acceptable match, you need to do some special processing. Insert a new record for the customer and add columns to identify the matched customer code as well as the similarity and confidence indices for the match. With both the new customer code and the matched customer code, you can now track the multiple customer codes assigned to the same customer.

To satisfy these requirements, SSIS provides a wide range of data cleansing functions to sanitize your source data sets according to general or specific business rules. In particular, the SSIS Data Flow task provides these key capabilities:

* **Reassigning column values** - To detect NULL, missing, or incorrect data values, SSIS provides the ability to compare incoming data to a validated reference data set by using a Lookup transformation. SSIS also provides the ability to reassign values by using custom expressions in a Derived Column transformation.
* **Handling data duplicates** –With the Fuzzy Lookup and Fuzzy Grouping transformations, SSIS provides the ability to perform imprecise data matches. The Fuzzy Lookup transformation in particular is great for matching dirty source data to a known set of cleansed, standardized data in a reference table. To understand how Fuzzy lookups work, consider the following example.

You have the following customer source record.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Source Record Number | Customer Code | Address | City | State | Zip | Country |
| 1 | 111 | One Micrsoft Way | Redmond | Washington | 98052 | USA |

Your reference table consists of the following two records.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Reference Record Number | Customer Code | Address | City | State | Zip | Country |
| 1 | 999 | One Microsoft Way | Redmond | Washington | 98052 | USA |
| 2 | 888 | Two Microsoft Way | Redmond | Washington | 98052 | USA |

Upon initial examination of the data values, there appears to be no precise mechanism to match the source data to the reference data. However, you know that there is the potential for duplicate customers with different customer codes. Upon closer visual inspection, you identify a potential match between Source Record Number 1 and Reference Record Number 1 by using the address, city, state, zip, and country fields. The misspelling of the word Microsoft in the address field of the source record prevents an exact match; however, the remaining profile values appear to be the same. You decide that the records are adequately similar and that you have a high degree of confidence that the match is correct.

In SSIS, the Fuzzy Lookup transformation performs in much the same way by examining data values, recommending potential matches, and then assigning similarity and confidence indices to display the quality of the data match. The similarity threshold indicates how closely the input data resembles its proposed match. A similarity value of one indicates an identical match. The closer the value is to one, the closer the match. In this scenario, Source Record Number 1 and Reference Record Number 1 record have a high similarity.

To complement the similarity threshold, *confidence* describes the level of certainty SSIS has about the match. For example, when three distinct people have the same name, SSIS uses a confidence index to indicate the certainty that the correct person was chosen for the match. For this customer example, Source Record Number 1 also bears a high similarity to Reference Record Number 2. Because of this, SSIS must identify which match is more likely to be accurate. In this case, SSIS has a higher confidence in matching with Reference Record Number 1.

Depending on the requirements of your application, you can establish your own similarity and confidence thresholds to determine what is and isn’t an acceptable match.

* **Extracting data** - To extract columns from the multi-value address field, SSIS provides the ability to define column delimiters for source file connections. In addition, SSIS also provides string manipulation functions to extract data values embedded in source fields.

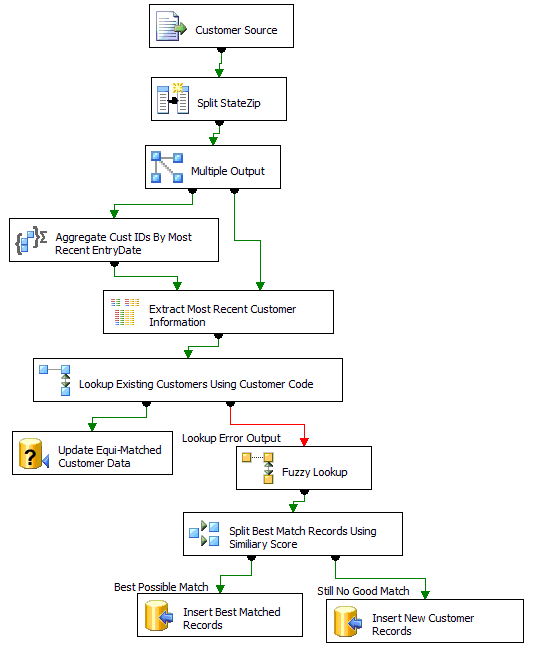
Figure 3 displays how these SSIS components can be used to clean the customer file.

Figure 3 –Cleansing Data Flow

The SSIS solution uses the following steps to implement the data cleansing.

1. SSIS first reads all records from the source file. The columns in the source file are configured to use tab delimiters with the exception of the address field, which is configured to use comma delimiters.
2. The comma delimiters split all columns except the State/Zip field. To split the combined State/Zip field, a Derived Column transformation is used to create two new columns for State and Zip. The Substring and FindString string manipulation functions are used to split the columns in the following expressions:

State = SUBSTRING(StateZip,1,FINDSTRING(StateZip,"",1) - 1)

Zip = SUBSTRING(StateZip,FINDSTRING(StateZip,"",1) + 1,LEN(StateZip) - FINDSTRING(StateZip,",",1) - 1)

1. The next step is to filter the source data so that only the most recent customer entries are used. To accomplish this task, the Multicast, Aggregate, and Merge Join transformations work together to retrieve the most recent records. You use the Aggregate transformation to find the most recent customer information; however, you also need to maintain the detailed data so that you have the detail to refresh your customer table. To accomplish this, you use a Multicast transformation to create a copy of the detailed source data. On one branch of the Multicast, you aggregate data by customer code using the Max entry date. To join back to the detailed branch, a Merge Join transformation joins the aggregate and detailed branches together using the customer code and the Max entry date.
2. With the filtered data set, you use a Lookup transformation to determine whether the customer code already exists in the table.
3. If the lookup is successful, data is updated in the destination table.
4. If the lookup is not successful, an error branch is created for additional processing.
5. In the error branch, a Fuzzy Lookup transformation performs a best match lookup to the customer table by using the customer’s address and profile information. Through iterative testing, in this scenario, you decide that an acceptable match must have a similarity threshold of at least .7 and a confidence level of at least .9.
6. A Conditional Split transformation examines the value of the similarity and confidence thresholds:
7. If this value is greater than .7 and the confidence level is greater than .9, the match is considered acceptable and a new record is inserted into the customer table. However, to indicate that this record is matched to another customer record with a different customer code, the match value, similarity value, and confidence thresholds are also stored in the table.
8. If the value is less than .7, the customer is considered to be new, and the source record data is loaded into the table.

#### Auditing

Auditing provides proof that your data integration solution satisfies necessary business, technical, regulatory standards. More specifically, auditing serves the following purposes:

* **Data lineage trail** – On a record-by-record and column-by-column basis, you can track all data integration operations such as inserts, updates, and deletes. You can also track any data quality issues that you encounter while executing your solution along with the action taken to resolve the issue.
* **Data validation –** To ensure that you have successfully processed all data, you can use auditing to perform data validation comparisons between sources and destinations.
* **Data execution statistics** – Data execution statistics help you track the overall data quality of your integration solution. You can track the success, failure, and execution duration of every component of your integration solution.

To maximize the value of auditing, consider creating an audit schema as a central part of your data integration architecture. A very common and effective approach for maintaining an audit history is to use a series of RDBMS tables. From these auditing tables, you can build reports to summarize data quality metrics over time.

An effective auditing schema must be able to handle both detail and summary levels of logging. The core component of data integration processing is a unit of work. A typical example of a unit of work is the loading of a table. As you load a table, you may encounter one or more data quality issues per record. For each issue, you must document a resolution. If there is an issue you cannot fix, such as an error record, you also need to capture the error record for additional investigation.

At the most summarized level of auditing, each unit of work will likely be grouped together into a workflow using a batch or parent program. For the batch program, you need to track the success, failure, and execution duration of the entire program along with every component or child that the batch executes.

To support varying levels of audit detail, the following list details a flexible auditing schema that can track both detailed and summarized data:

* **Audit Errors table** – Stores error details and warnings that require immediate and manual attention.
* **Audit Detail table** - Tracks detailed operations including data cleansing, the number of records processed as well as the number of records inserted, updated, or deleted per data destination.
* **Audit Error Records table(s)** – Depending on your design, you might designate one table to store all error records or you might create several error record tables, one for each destination table.
* **Audit Component table** – Tracks the execution of specific units of work such as an individual package execution.
* **Audit Batch table** – Tracks the execution of the entire data load across all integration components such as a group of package executions.

##### SSIS auditing scenario

To understand how auditing can be used to track the data quality of your integration solution, consider the following scenario.

You have a data integration solution that consists of two types of packages:

* The first type of package is a child or component package that contains multiple tasks or units of work that insert and update data in a specific destination table.
* The second type of package is a parent or batch package that executes a group of child packages based on a user-defined workflow.

For each of these packages, audit and store the following events:

* **Audit Batch** - Completion status and execution duration of a parent package.
* **Audit Component** – Execution status and execution duration of a child package.
* **Audit Detail** - Execution status and duration for each task in a child package as well as the number of inserts and updates that each task performs.
* **Audit Errors -** Any errors or failures that arise during package execution.
* **Audit Error Records** – Any error records that can not be loaded into a destination.

In addition to the above requirements, the specific package execution that created or last updated a particular record needs to be tracked for each source record and stored in a data lineage column.

To support data auditing activities, SSIS provides extensive logging capabilities that can be customized to meet the needs of your specific application.

* Logging providers allow you to log SSIS package executions to several data sources including SQL Server, text files, Windows event log, and SQL Profiler. For SQL Server destinations, logging information is stored in an SSIS-generated table called sysdtslog90. You can choose whether this table is stored in the same database as your application data or whether it is stored in a dedicated auditing database.
* For each logging provider, you can configure which events SSIS logs. Examples of events include informational messages, warnings, and errors.
* To enhance the logging provided by SSIS in the sysdtslog90 table, you can use event handlers to add your own custom logging events to track inserts, updates, and other data cleansing functions.
* To ensure that auditing is persisted in the lineage of the source records, you can tag each record that is inserted or updated with an SSIS package execution identifier.
* To further augment auditing, you can also extend SSIS logging with your own custom logic by creating custom log providers for which you can dynamically change log settings according to the data quality issues that you find.

To address the specific auditing needs of this scenario

1. To expose the necessary logging events, you must enable logging for each package. Figure 4 displays the options you can select when you configure SSIS for logging.

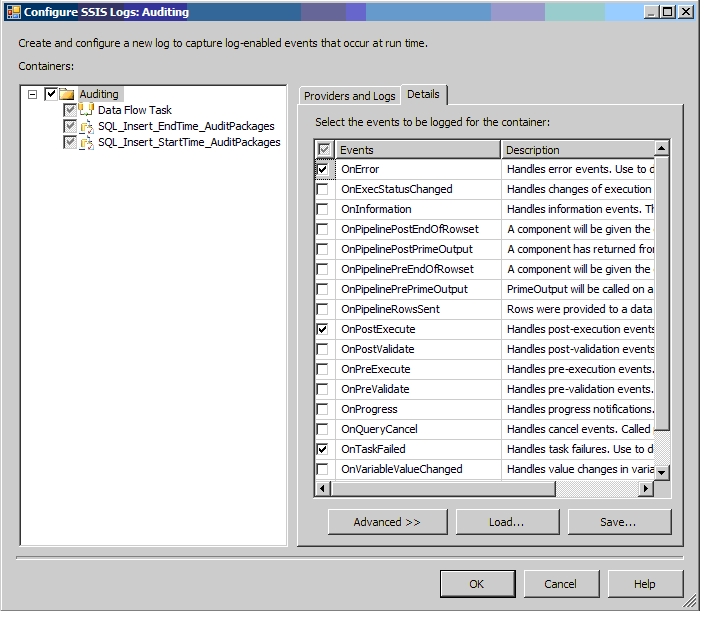


Figure 4   Logging Options

1. To meet the error logging needs of this scenario, the OnError and OnTaskFailed events have been enabled. This logging information is stored in the sysdtslog90 table. This table can be used to satisfy the **Audit Errors** requirement for this scenario.
2. In addition to OnError and OnTaskFailed events, the OnPostExecute event has also been enabled. The OnPostExecute event is accessed in an event handler to generate an application-specific log of inserts and updates. Figure 5 displays the OnPostExecute event handler for a Data Flow task. Within the event handler, an Execute SQL task logs the number of inserted and updated records to a custom Audit Detail table that tracks the operations of each Data Flow task. The event handler logging satisfies the **Audit Detail** requirements of this data load.

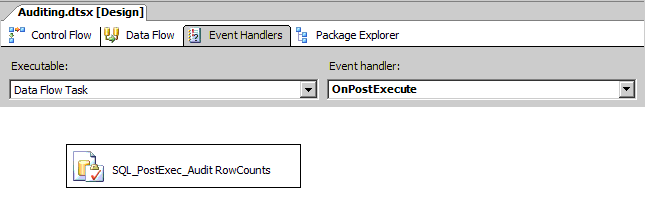


Figure 5   Event Handlers

1. In addition, within each Data Flow task, an OnError Branch has been configured to redirect records to an **Audit Errors** table. An audit error table has been created for each destination table.
2. To satisfy the requirements for **Audit Batch** and **Audit Component**, enhancements are added to the supplied SSIS logging capabilities. While the SSIS-generated sysdtslog90 table contains a large amount of execution information for each package, in this scenario, you decide to break out execution information by package type so you can quickly see how long an entire batch load takes to execute. To accomplish this, you create and populate separate audit tables for parent and child packages called Audit Batch and Audit Component respectively.
3. To log start and end times into these tables, you use SQL statements at the beginning and end of each parent and child package that insert the time details. Along with the time details, for each child package execution, you also log the parent package execution that called it. To pass the parent package execution ID to a child package, you use SSIS Configurations using variables.
4. The final requirement is to attach the specific package execution that created or last updated a particular record. To satisfy this requirement, two columns are created on each destination table: **created\_by** and **last\_modified\_by**. The columns are populated by the Package ID that either created or updated the record.
5. To expose the package information within a Data Flow task, you can use the Audit transformation to add the package ID and/or package name as one or more columns in your data flow. For insert operations, you populate the **created\_b**y field with the package ID. For update operations, you populate the **last\_modified\_by** field with the package ID.

At the end of this process, you have established SSIS logging at the most summarized and most detailed levels of execution. In addition, you have created an auditing architecture that can be used across your SSIS integration solutions to enforce consistent auditing and logging practices.

#### SSIS Data Quality Partners

The Microsoft partner system includes the following partners who provide specialized data quality solutions that enhance and extend the data integration functionality provided by the SSIS platform. Partners are listed in alphabetical order.

* **ABM Dataminers** - eCartography from ABM Dataminers is a Predictive Data Management product that is written in Microsoft Visual C#® (available with Visual Studio® 2005). It fully integrates with Microsoft SQL Server 2005 SSIS, and uses the .NET 2.0 Framework. eCartography can access most databases and file formats to create a broad range of data profiling, data standardization, and data cleansing solutions. [www.predictivedatamanagement.com](http://www.predictivedatamanagement.com)
* **DQ Global** - DQ Global software de-duplicates, cross-matches, and links data to create a single customer view. DQ products also suppress and enhance UK data and correct addresses in up to 230 regions. <http://www.dqglobal.com>
* **IntelligentSearch** – Intelligent Search Technology, Ltd. has developed fast and accurate name and address search and matching software. For more information see the benchmark tests conducted by the New York State Division of Criminal Justice. ([Benchmark Study](http://www.intelligentsearch.com/benchmark.html)) IntelligentSearch’s integration of existing products (fuzzy searching/matching and data de-duplication) with SSIS allows database administrators (DBAs) to create custom interactive searches, batch merge/purge, and de-duplication solutions by adding data transformation components into their data flows. [www.intelligentsearch.com](http://www.intelligentsearch.com)

#### Conclusion

SSIS provides a data integration platform for effectively and efficiently incorporating data quality components into integration solutions by:

* Providing the capabilities to inspect data prior to loading it into its destination.
* Supplying a wide range of data cleansing functionality to scrub data according to specific or generic data standards.
* Enabling operational auditing for summarized and detailed tasks to ensure that data integration solutions process data effectively.
* Providing a comprehensive, robust, and scalable platform and a set of development tools to create and manage large-scale data integration solutions.
* Maximizing return on investment (ROI) through the lowest total cost of ownership when compared with competitive platforms.
* Providing a system of partners who provide value-added solutions to SSIS data integration.

For more information:

<http://msdn.microsoft.com/sql/>

#### About the Authors

Elizabeth Vitt, Intellimentum

Elizabeth Vitt has ten years of business development, project management, consulting, and training experience in business intelligence. Her industry experience includes BI implementations in retail, manufacturing, and financial services. She has specialized experience as an educator in data warehousing, extraction, transformation, and loading (ETL), and OLAP design and implementation. Elizabeth is an author of Microsoft Official Curricula courses for Microsoft Business Intelligence product offerings as well *Business Intelligence: Making Better Decisions Faster* from Microsoft Press. Elizabeth has successfully implemented several Microsoft SQL Server 2005 BI solutions using Analysis Services, Integration Services, and Reporting Services and is the author of several SQL Server 2005 BI white papers. [www.intellimentum.com](http://www.intellimentum.com)

Hitachi Consulting

As Hitachi, Ltd.’s (NYSE: HIT) global consulting company, Hitachi Consulting is a recognized leader in delivering proven business and IT solutions to Global 2000 companies across many industries. We leverage decades of business process, vertical industry, and leading-edge technology experience to understand each company’s unique business needs. From business strategy development through application deployment, our consultants are committed to helping clients quickly realize measurable business value and achieve sustainable ROI.

Hitachi Consulting is also a Microsoft Certified Gold Partner for Business Intelligence, exclusive provider of curriculum and instructors for the Microsoft SQL Server 2005 Business Intelligence Ascend training program, and an experienced systems integrator with successful SQL Server 2005 BI implementations at companies participating in the Microsoft Technology Adoption Program (TAP).

We offer a client-focused, collaborative approach and transfer knowledge throughout each engagement. For more information, visit [www.hitachiconsulting.com](http://www.hitachiconsulting.com/).

Hitachi Consulting – Inspiring your next success®