THE EXPERT'S VOICE® IN SQL SERVER

SQL Server 2012 Integration Services Design Patterns

IMPROVE YOUR EFFICIENCY AS A DATA INTEGRATION DEVELOPER

Andy Leonard, Matt Masson, Tim Mitchell, Jessica Moss, and Michelle Ufford

Apress®

SQL Server 2012 Integration Services Design Patterns

Andy Leonard
Matt Masson
Tim Mitchell
Jessica M. Moss
Michelle Ufford

Apress®

SQL Server 2012 Integration Services Design Patterns

ISBN 978-1-4302-3771-6

ISBN 978-1-4302-3772-3 (eBook)

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

President and Publisher: Paul Manning
Lead Editor: Jonathan Gennick
Developmental Editor: Richard Carey
Technical Reviewers: David Dye, Sergio Filho, Allan Mitchell, and David Stein
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh
Coordinating Editor: Adam Heath
Copy Editor: Chandra Clarke
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com/9781430237716. For detailed information about how to locate your book's source code, go to www.apress.com/source-code.

For my loving wife, Christy.
--Andy Leonard

Contents at a Glance

Contents

Foreword

For me, one of the great pleasures of working at Microsoft was shepherding new products from concept to release. However, it was even more fulfilling to witness the birth and growth of new communities of users, for what is a product without a user? Just bits and bytes on a disk. In my role as Group Product Manager of the SQL Server Integration Services team, it was my privilege to watch the evolution of both the SSIS application and the social network of users.

The Integration Services team, under the exceptional leadership of Kamal Hathi, delivered a product in 2005– SQL Server Integration Services– that was intended to be not only a powerful application in its own right, but a platform for customers and partners to extend and expand as their data integration needs changed and grew over time. Over the years (and through several versions of the product) SQL Server Integration Services has grown to become an industry-leading technology.

When we started developing what users now call SSIS, anyone building a data warehouse had only two choices: expensive, highly specialized tools for Extraction Transformation and Loading (ETL), or tedious, difficult-to-maintain, custom coding. With SSIS we wanted to break through those traditional restrictions: to deliver a truly scalable tool, simple enough for the beginner, but with the extensibility and programmability of a platform for the expert.

Little did we anticipate how eagerly the SQL Server user community would embrace this tool! Our user base grew quickly, and, as in any group endeavor, natural leaders emerged. The authors of this splendid book are, quite simply, among the most outstanding contributors to the SSIS social network. They are leaders not only because of their skills, but because of their tireless support and commitment to helping others. This book distills that learning, and that community focus, into a volume to keep by your keyboard for years.

The challenge with a tool such as SSIS is that there are simply so many possibilities facing the user. If I can choose a prebuilt component, which one do I choose? If I can extend the capabilities with script, when should I do that? How do I choose between the many ways to load a slowly-changing-dimension table, or for handling XML?

SQL Server 2012 Integration Services Design Patterns not only provides solutions to such problems; even more usefully, this book channels the authors' extensive experience into *patterns*. In recent years, design patterns have proved their value to software developers as flexible templates for addressing recurring problems that still need specific implementation details. *SQL Server 2012 Integration Services Design Patterns* takes this approach, quite uniquely, into the world of data warehousing and ETL.

The result is a collaborative work by experts, suitable for beginners and advanced users alike.

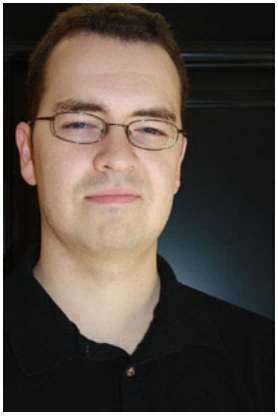
Even though I moved on from the SSIS team, and from Microsoft, some years ago now, it is a pleasure for me to remain in touch with the user community I admire so much. And it is a honor for me to introduce you to this much-anticipated and valuable book.

Happy integrating!

Donald Farmer
VP Product Management, QlikTech

About the Authors

Andy Leonard is a SSIS trainer and consultant, SQL Server database and Integration Services developer, SQL Server data warehouse developer, community mentor, SQLBlog.com blogger, and engineer. He is co-author of *Professional SQL Server 2005 Integration Services* and *SQL Server MVP Deep Dives*. His background includes web application architecture and development, Visual Basic, ASP, SQL Server Integration Services (SSIS), and data warehouse development using SQL Server 2000, 2005 and 2008.



Matt Masson is a software development engineer working with the SQL Server Integration Services (SSIS) team. Matt has worked on many aspects of the SSIS product, including upgrade, performance, and overall user experience. He is a frequent presenter at Microsoft conferences, and maintains the SSIS Team blog (http://blogs.msdn.com/b/mattm/). Prior to joining Microsoft in 2006, Matt was a developer on a number of business intelligence reporting and analytical products. He lives in Montreal, Quebec, and works remotely with his Redmond-based team.

Tim Mitchell is a business intelligence consultant, database developer, speaker, and trainer. He has been working with SQL Server for more than eight years, primarily in business intelligence, ETL/SSIS, database development, and reporting. He has earned a number of industry certifications, holds a bachelor's degree in computer science from Texas A&M University at Commerce, and is a Microsoft SQL Server "Most Valuable Professional." Tim is a business intelligence consultant for Artis Consulting in the Dallas, Texas area. As an active member of the community, Tim has spoken at venues including numerous SQL Saturday events, Houston Tech Fest, and various user groups and PASS virtual chapters. He is a board member and speaker at the North Texas SQL Server User Group in Dallas, serves as the co-chair of the PASS BI Virtual Chapter, and is an active volunteer for PASS. Tim is an author and forum contributor on SQLServerCentral.com and has published dozens of SQL Server training videos on SQLShare.com. You can visit his website and blog at TimMitchell.net or follow him on Twitter at @Tim_Mitchell.

Jessica M. Moss is a well-known author, and speaker on Microsoft SQL Server business intelligence. She has created numerous data warehouse and business intelligence solutions for companies in different industries, and has delivered training courses on Integration Services, Reporting Services, and Analysis Services. While working for a major clothing retailer, Jessica participated in the SQL Server 2005 TAP program, where she developed best implementation practices for Integration Services. Jessica has authored technical content for multiple magazines, websites, and books, and has spoken internationally at conferences such as the PASS Community Summit, SharePoint Connections, and the SQLTeach International Conference. As a strong proponent of developing user-to-user community relations, Jessica actively participates in local user groups and code camps in central Virginia. In addition, Jessica volunteers her time to help educate people through the PASS organization.

Michelle Ufford is a SQL Server database developer, Integration Services developer, Microsoft SQL Server MVP, and self-proclaimed scripting junkie. She specializes in performance tuning and high-volume VLDB (very large database) development, although her experience also includes database automation, operational predictive analytics, and all stages of the data lifecycle— from OLTP to data warehousing. Michelle is an active member of the SQL Server community and a frequent presenter, most notably at PASS Summit. Michelle has a very popular blog at `SQLFool.com` and can be found on Twitter at `@sqlfool`.

About the Technical Reviewers

David Stein is a Senior Business Intelligence Consultant, specializing in designing, developing, and maintaining data warehouses using Microsoft BI Tools focusing on the health care sector. He enjoys helping others as an active volunteer with his local PASS Chapter, contributor to SQL University, and presenting at the local and regional level. He also blogs regularly at `Made2Mentor.com`.

Allan Mitchell is the joint owner of Copper Blue Consulting Ltd. Copper Blue Consulting focus on getting the right data to the right people at the right time and in the right format. We are passionate about data integrity and suitability. We have worked all over the world in a variety of industries and on projects both large and small. We specialise in Extract, Transform and Load Complex Event Processing Master Data Management Data Visualisation Operational and Predictive Analytics. We offer training as well as consultancy.

David Dye is a Microsoft SQL Server MVP, instructor, and author specializing in relational database management systems, business intelligence systems, reporting solutions, and Microsoft SharePoint. For the past 9 years David's expertise has been focused on Microsoft SQL Server development and administration. His work has earned him recognition as: a Microsoft MVP in 2009 and 2010, a moderator for the Microsoft Developer Network for SQL Server forums, Innovator of the Year runner-up in 2009 by SQL Server Magazine, and in the Training Associates Technical Trainer Spotlight in April 2011. David currently serves as a technical reviewer and coauthor with APress Publishing in the SQL Server 2012 series, and as an author with Packt Publishing.

Acknowledgments

I would like to thank my coauthors for agreeing to work with me on this project: Matt Masson, Tim Mitchell, Jessica Moss, and Michelle Ufford are awesome people and outstanding technologists, as this book will bear out. Our editorial team at Apress is top-shelf. I sincerely appreciate the leadership of Jonathan Gennick who shepherded this project through many months of writing, editing, and rewriting to deliver this manuscript in its current form. Kudos to Adam Heath and Mark Powers for their help and communications, and to several unnamed - yet vital - people at Apress for making this book possible.

I would like to extend special thanks to Donald Farmer for leading the Microsoft program to develop SSIS, and the many members of the SSIS team - past and present - who have labored to produce and support an outstanding enterprise data integration product. Many thanks to the members of the SSIS Community who selflessly share their expertise with me and others on forums and social media.

Whenever I sign up for a book project, Christy signs up, as well. For her unwavering love and dedication to me and our family, I thank her. I would also like to thank Stevie Ray, Emma, and Riley for as much patience as children their age can muster; and Steve and Tina Smith for their support and help.

–Andy Leonard

I am incredibly grateful to the large group of people who contributed to making this book a reality. To Andy Leonard, who first approached me about this idea some 2 years before its printing, I express my thanks for including me in this project. To Jessica Moss, Matt Masson, and Michelle Ufford, I am honored just to be named in the same publication as all of you. To Jonathan Gennick, Adam Heath, Mark Powers, and rest of the team at Apress, thank you for believing in us, and for keeping us on task and on schedule.

Just as importantly, I'd like to thank a smaller team closer to home. To my wife Rachel, to my kids Ryan, Evan, and Kaylee: thank you for being patient through this whole process. Even though your names aren't on the cover, you had as much invested in this book as I did.

–Tim Michell

The writing of this book has been a great experience with a dedicated team. Thank you to my wonderful coauthors: Andy Leonard, Matt Masson, Tim Mitchell, and Michelle Ufford and the Apress team: Jonathan Gennick, Adam Heath, and Mark Powers. Thank you to my friends and family for their patience as I disappeared to write for hours on end and the SQL community for letting me share my knowledge.

– Jessica M. Moss

First and foremost, I want to extend my deepest thanks to Andy Leonard for his efforts on this book. Andy is one of the smartest and nicest people I know, and I am deeply honored he invited me to be a part of this project. I also want to thank John Hoang and Brian Davis for their time and invaluable contributions to my chapters; the entire SSIS, PDW, and SQLCAT teams for building such incredible products and for sharing their knowledge with me; and Chris Leonard, who encouraged me to get involved in the SQL Server community one afternoon on our way to Starbucks. Little did I know the impact his advice would have on my life and career. I dedicate my writing to my children, Chloe and Ethan, who constantly amaze me and fill my life with such love and joy. To Eliza, whose inquisitive mind inspires me. And most of all, to my husband John, who is the most remarkable father, supportive friend, and loving husband anyone could hope for.

–Michelle Ufford

CHAPTER 1

Metadata Collection

The first Integration Services design pattern we will cover is metadata collection. What do we mean by "metadata collection"? Good question. This chapter could also be called "Using SSIS to Save Time and Become an Awesome DBA." Many DBAs spend a large portion of time on monitoring activities such as verifying backups, alerting on scheduled job failures, creating schema snapshots ("just in case"), examining space utilization, and logging database growth over time, to name just a very few. Most RDBMS systems provide metadata to help DBAs monitor their systems. If you've been a DBA for a few years, you may even have a "tool bag" of scripts that you use to interrogate metadata. Running these scripts manually is easy when you have just one or two servers; however, this can quickly become unwieldly and consume a large portion of your time as your enterprise grows and as the number of database servers increases.

This chapter examines how to use Integration Services and the metadata that exists within SQL Server to automate some of these routine tasks.

Introducing SQL Server Data Tools

One of the major features of SQL Server 2012 is the introduction of SQL Server Data Tools (SSDT). SSDT replaces Business Intelligence Development Studio (BIDS) and leverages the maturity of the Visual Studio product to provide a unified development platform for SQL Server, Business Intelligence (BI), and .NET applications. This book is written using SSDT, although the appearance of the Integration Services designer interface is largely the same as BIDS 2008. SSDT provides backward compatibility for Integration Services 2008 packages via the SSIS Package Upgrade Wizard.

Tip Don't have SQL Server Data Tools installed? SSDT is a free component of the SQL Server platform and is available to all SQL Server users. You can install SSDT from your SQL Server installation materials under the "Feature Selection" menu.

A Peek at the Final Product

Let's discuss the Integration Services package we will be creating in this chapter.

In SQL Server, we will do the following:

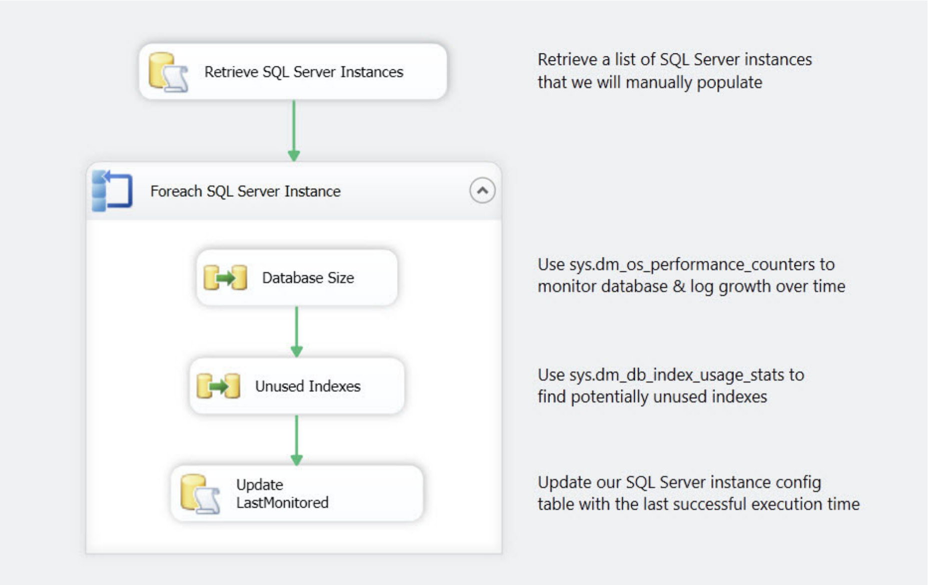
1. Create a database to act as our central repository for database monitoring.
2. Create a table to store a list of SQL Server instances that we wish to monitor.

3. Create a table for each of the data elements we wish to monitor (unused indexes and database growth).

In Integration Services, we will do the following:

1. Create a new Integration Services package.
2. Retrieve a list of SQL Server instances and store the list in a variable.
3. Create an OLE DB connection with a dynamically populated server name.
4. Iterate through each database and
    a. Retrieve current database and log file sizes for historical monitoring.
    b. Retrieve a list of index candidates for potential redesign or dropping.
    c. Update the Last Monitored value for each SQL Server instance.

This is a very flexible model that can easily be expanded to include many more monitoring tasks. A screenshot of the completed package is displayed in Figure 1-1.



Retrieve a list of SQL Server instances that we will manually populate

Use sys.dm_os_performance_counters to monitor database & log growth over time

Use sys.dm_db_index_usage_stats to find potentially unused indexes

Update our SQL Server instance config table with the last successful execution time

Figure 1-1. *The MetadataCollection package*

If this is not your first Integration Services package, maybe you've noticed that this package is missing a few best practices, such as error handling. In the interest of clarity, the package we create will focus only on core design patterns; however, we will call out best practices when applicable.

Also, please note that the T-SQL examples will only work with SQL Server 2005 or later.

SQL Server Metadata

Although metadata can be collected from any RDBMS that provides an interface for accessing it, this chapter uses SQL Server as its metadata source. The focus of this chapter is not on the actual metadata, but rather the pattern of metadata collection. Still, it is useful for you to have a basic understanding of the type of metadata that is available.

SQL Server exposes a wealth of information through catalog views, system functions, dynamic management views (DMVs), and dynamic management functions (DMFs). Let's briefly examine some of the metadata we will be using in this chapter.

■ **Tip** SQL Server Books Online is a great resource for learning more about the types of metadata available in SQL Server. Try searching for "metadata functions," "catalog views," and "DMVs" for more information.

sys.dm_os_performance_counters

The sys.dm_os_performance_counters DMV returns server performance counters on areas including memory, wait stats, and transactions. This DMV is useful for reporting file sizes, page life expectancy, page reads and writes per second, and transactions per second, to name but a few.

sys.dm_db_index_usage_stats

The sys.dm_db_index_usage_stats DMV contains information on index utilization. Specifically, a counter is incremented every time an index has a seek, scan, lookup, or update performed. These counters are reinitialized whenever the SQL Server service is started. If you do not see a row in this DMV for a particular index, it means that a seek, scan, lookup, or update has not yet been performed since the last server reboot.

sys.dm_os_sys_info

The sys.dm_os_sys_info DMV contains information about server resources. Perhaps the most frequently used piece of information in this DMV is the *sqlserver_start_time* column, which tells you the last time the SQL Server service was started.

sys.tables

The sys.tables catalog view contains information about every table that exists within the database.

sys.indexes

The sys.indexes catalog view contains information about every index in the database. This includes information such as whether an index is clustered or nonclustered and whether the index is unique or nonunique.

sys.partitions

The sys.partitions catalog view gives visibility into the partitioning structure of an index. When an index has more than one partition, the data in the index is split into multiple physical structures that can be accessed using the single logical name. This technique is especially useful for dealing with large tables, such as a transaction history table. If a table is not partitioned, the table will still have a single row in sys.partitions.

sys.allocation_units

The sys.allocation_units catalog view contains information about the number of pages and rows that exist for an object. This information can be joined to the sys.partitions catalog view by joining the *container_id* to the *partition_id*.

Setting Up the Central Repository

Before we can begin development on our Integration Services package, we need to set up some prerequisites in SQL Server. First and foremost, we need to create a database that will act as our central data repository. This is where our list of SQL Server instances will reside and where we will store the metadata we retrieve for each SQL Server instance. Many enterprises also find it convenient to store all error and package logging to this same central database. This is especially beneficial in environments where there are numerous DBAs, developers, and servers, as it makes it easy for everyone to know where to look for information. The T-SQL code in Listing 1-1 creates the database we will use throughout the rest of this chapter.

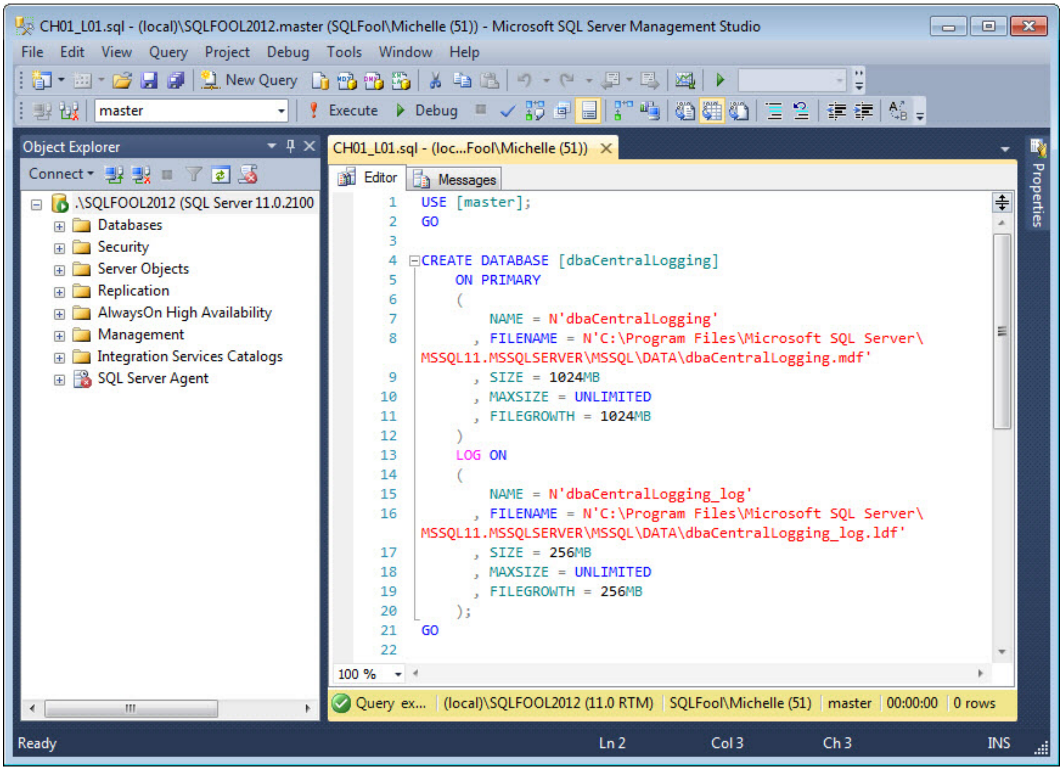
Listing 1-1. Example of T-SQL Code to Create a SQL Server Database

```
USE [master];
GO

CREATE DATABASE [dbaCentralLogging]
    ON PRIMARY
    (
       NAME = N'dbaCentralLogging'
      ,FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\
MSSQL\DATA\dbaCentralLogging.mdf'
     , SIZE = 1024MB
     , MAXSIZE = UNLIMITED
     , FILEGROWTH = 1024MB
    )
    LOG ON
    (
       NAME = N'dbaCentralLogging_log'
     , FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\
MSSQL\DATA\dbaCentralLogging_log.ldf'
     , SIZE = 256MB
     , MAXSIZE = UNLIMITED
     , FILEGROWTH = 256MB
    );
GO
```

Please note that your file directory may differ from the one in the preceding example.

This code can be executed from SQL Server Management Studio (SSMS), as demonstrated in Figure 1-2, or from your favorite query tool.


CH01_L01.sql - (local)\SQLFOOL2012.master (SQLFool\Michelle (51)) - Microsoft SQL Server Management Studio
File
Edit
View
Query
Project
Debug
Tools
Window
Help
New Query
master
Execute
Debug
Object Explorer
Connect
.\SQLFOOL2012 (SQL Server 11.0.2100
Databases
Security
Server Objects
Replication
AlwaysOn High Availability
Management
Integration Services Catalogs
SQL Server Agent
CH01_L01.sql - (loc...Fool\Michelle (51))
Editor
Messages
Properties
100 %
Query ex...
(local)\SQLFOOL2012 (11.0 RTM)
SQLFool\Michelle (51)
master
00:00:00
0 rows
Ready
Ln 2
Col 3
Ch 3
INS

```
USE [master];
GO

CREATE DATABASE [dbaCentralLogging]
    ON PRIMARY
    (
        NAME = N'dbaCentralLogging'
      , FILENAME = N'C:\Program Files\Microsoft SQL Server\
MSSQL11.MSSQLSERVER\MSSQL\DATA\dbaCentralLogging.mdf'
      , SIZE = 1024MB
      , MAXSIZE = UNLIMITED
      , FILEGROWTH = 1024MB
    )
    LOG ON
    (
        NAME = N'dbaCentralLogging_log'
      , FILENAME = N'C:\Program Files\Microsoft SQL Server\
MSSQL11.MSSQLSERVER\MSSQL\DATA\dbaCentralLogging_log.ldf'
      , SIZE = 256MB
      , MAXSIZE = UNLIMITED
      , FILEGROWTH = 256MB
    );
GO

```

Figure 1-2. *SQL Server Management Studio 2012*

Next, we need a list of SQL Server instances that need to be monitored. The easiest way to accomplish this is to store a list of database instance names in a file or table. We will use the latter method. Using the code in Listing 1-2, create that table now.

Listing 1-2. Example of T-SQL Code to Create a Table for Monitoring SQL Server Instances

```
USE dbaCentralLogging;
GO

CREATE TABLE dbo.dba_monitor_SQLServerInstances
(
SQLServerInstance       NVARCHAR(128)
LastMonitored           SMALLDATETIME           NULL

        CONSTRAINT PK_dba_monitor_SQLServerInstances
                PRIMARY KEY CLUSTERED(SQLServerInstance)
);
```

You will then need to populate the table with the list of SQL Server instances you wish to monitor. The code in Listing 1-3 will walk you through how to do this, although you will need to use real SQL Server instances.

Listing 1-3. Example of T-SQL Code to Insert Data into the dba_monitor_SQLServerInstances Table

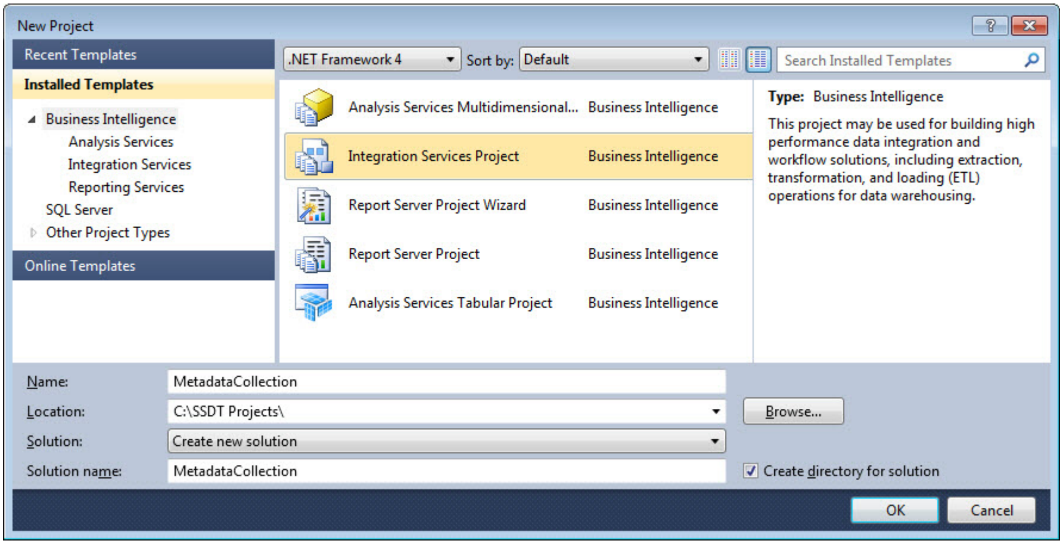
```
INSERT INTO dbo.dba_monitor_SQLServerInstances
(
          SQLServerInstance
)
SELECT @@SERVERNAME-- The name of the server that hosts the central repository
UNION ALL
SELECT 'YourSQLServerInstanceHere'-- Example of a SQL Server instance
UNION ALL
SELECT 'YourSQLServerInstance\Here';-- Example of a server with multiple instances
```

We still need to create two tables to store the metadata we collect, but we will create these as we get to the appropriate section in the package. Next, we will create our Integration Services package.

The Iterative Framework

In this section, we lay the foundation for our iterative framework. In other words, we will create a repeatable pattern for populating a variable with a list of SQL Server instances, then iterating through the list and performing an action on each server. Let's do this now.

First, open SSDT. Create a new project by navigating to **File➤New➤Project**. Click **Business Intelligence** under Installed Templates, and then click **Integration Services Project** in the Installed Templates window. Name the project **Meta data Collection**, as illustrated in Figure 1-3.



.NET Framework 4 Sort by: Default

Analysis Services Multidimensional...	Business Intelligence
Integration Services Project	Business Intelligence
Report Server Project Wizard	Business Intelligence
Report Server Project	Business Intelligence
Analysis Services Tabular Project	Business Intelligence

Type: Business Intelligence

This project may be used for building high performance data integration and workflow solutions, including extraction, transformation, and loading (ETL) operations for data warehousing.

Name: MetadataCollection
Location: C:\SSDT Projects\
Solution: Create new solution
Solution name: MetadataCollection
☑ Create directory for solution

Figure 1-3. *New integration services project*

Please note that your default Location will most likely be different from the directory pictured in Figure 1-3.

We now need to create two variables. The first variable will be used to store the list of SQL Server instances we retrieve. The second variable will store a single instance name as we iterate through our list.

To access the variable menu, select **Variables** under the SSIS menu (Figure 1-4); you can also access the Variables menu by right-clicking the designer surface.

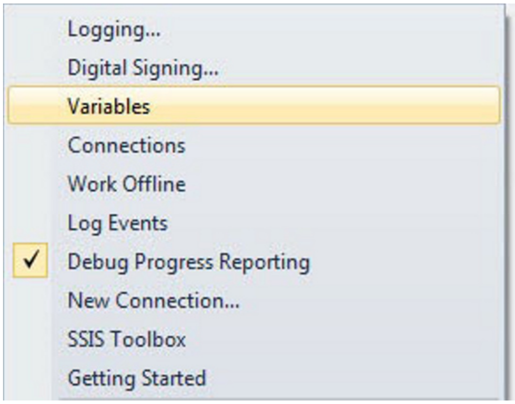
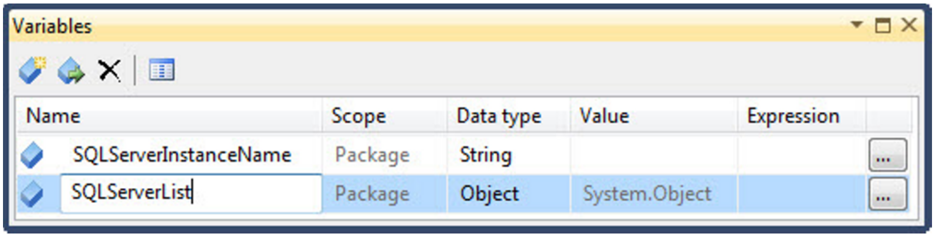


Figure 1-4. *Opening the Variables menu*

Add the following variables by clicking the **Add Variable** icon on the far left of the Variables menu, as illustrated in Figure 1-5:

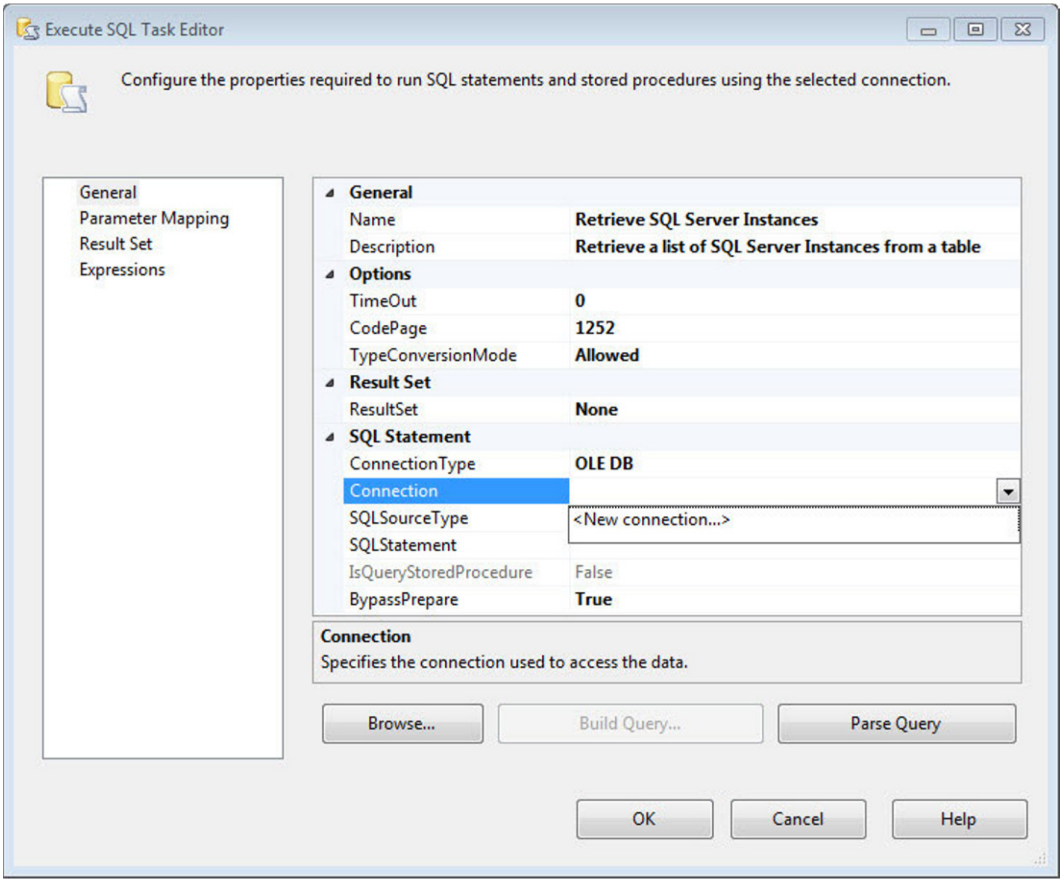
- SQLServerList – Object
- SQLServerInstanceName – String



Name	Scope	Data type	Value	Expression
SQLServerInstanceName	Package	String		...
SQLServerList	Package	Object	System.Object	...

Figure 1-5. *Package-scoped variables*

Now that we have a place to store our list of instances, we're ready to retrieve them. Drag a new **Execute SQL Task** from the SSIS Toolbox onto the designer surface. Rename the task **Retrieve SQL Server Instances**, and then double-click it to open the Execute SQL Task Editor. Click the drop-down under Connection, and then select < **New connection...**>, as seen in Figure 1-6.



Execute SQL Task Editor

Configure the properties required to run SQL statements and stored procedures using the selected connection.

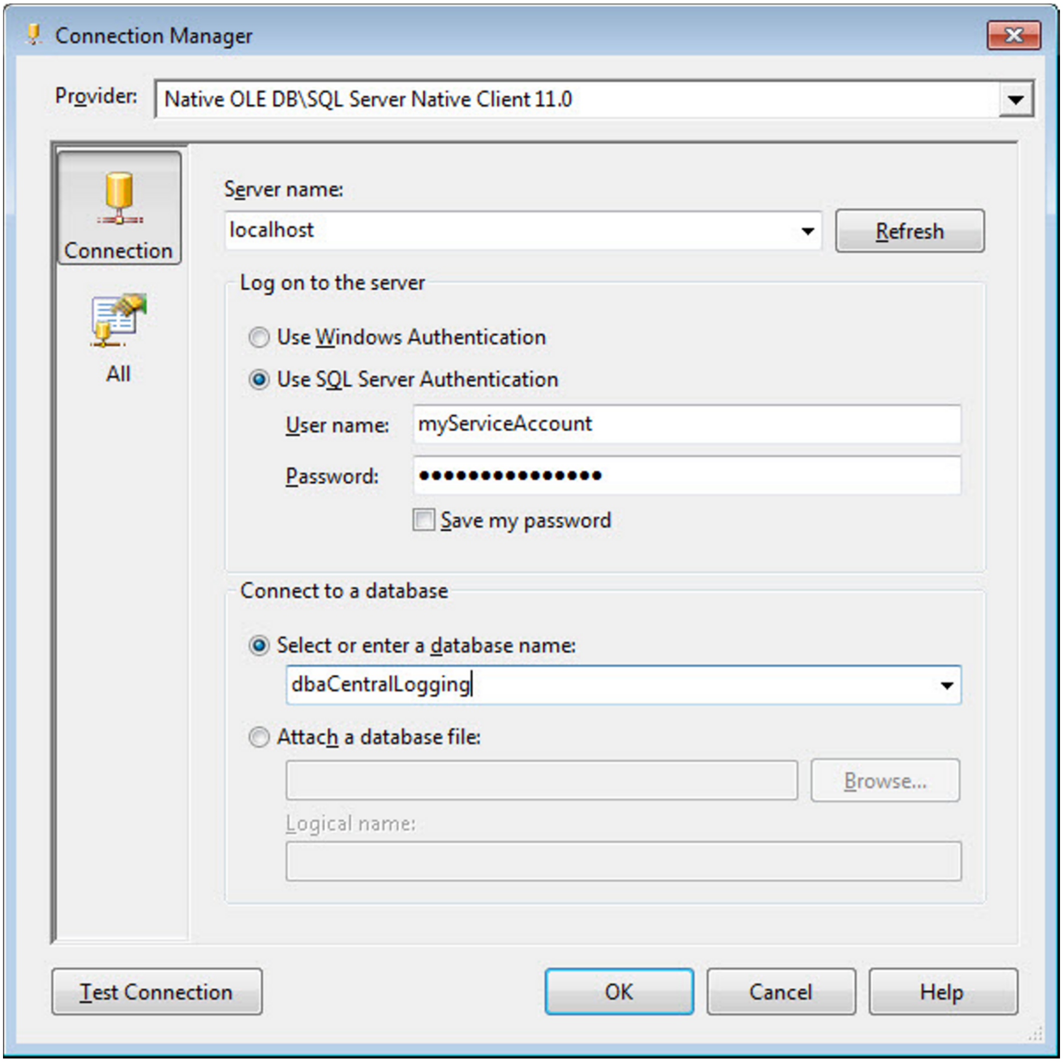
General	
Name	**Retrieve SQL Server Instances**
Description	**Retrieve a list of SQL Server Instances from a table**
Options	
TimeOut	**0**
CodePage	**1252**
TypeConversionMode	**Allowed**
Result Set	
ResultSet	**None**
SQL Statement	
ConnectionType	**OLE DB**
Connection	
SQLSourceType	<New connection...>
SQLStatement	
IsQueryStoredProcedure	False
BypassPrepare	**True**

Connection
Specifies the connection used to access the data.

Figure 1-6. *The Execute SQL Task Editor*

In the Configure OLE DB Connection Manager menu, click **New**. In the Server Name field, enter the database server where you created the database in Listing 1-1. Regardless of whether you are using Windows or SQL Server authentication, make sure that the account has sufficient permissions to each of the instances in our *dba_monitor_SQLServerInstances* table. Under “Select or enter a database name,” select **dbaCentralLogging** from the drop-down menu, as illustrated in Figure 1-7. Click **OK** to return to the Execute SQL Task Editor.

■ **Note** Permissions requirements vary depending on the type of metadata you wish to retrieve. For more information on the permissions necessary to access a specific object, please refer to the object page within SQL Server Books Online.



Provider: Native OLE DB\SQL Server Native Client 11.0

Server name: localhost

○ Use Windows Authentication

● Use SQL Server Authentication

User name: myServiceAccount

Password: •••••••••••••••

□ Save my password

● Select or enter a database name: dbaCentralLogging

○ Attach a database file:

Figure 1-7. The Connection Manager

We now need to write the SQL statement that will retrieve the list of SQL Server instances. Click the [**...**] icon to the right of the SQLStatement field, and then enter the T-SQL code from Listing 1-4.

Listing 1-4. T-SQL Statement to Retrieve SQL Server Instances

```
SELECT SQLServerInstance FROM dbo.dba_monitor_SQLServerInstances;
```

Because we are retrieving an array of values, select **Full result set** from the ResultSet drop-down. Your Execute SQL Task Editor should now resemble Figure 1-8; however, your Connection values will likely be different.