Data Integration Life Cycle Management with SSIS

A Short Introduction by Example

Andy Leonard

Apress®

Data Integration Life Cycle Management with SSIS

A Short Introduction by Example

Andy Leonard

Apress®

Data Integration Life Cycle Management with SSIS

Andy Leonard
Farmville, Virginia, USA

ISBN-13 (pbk): 978-1-4842-3275-0 ISBN-13 (electronic): 978-1-4842-3276-7
https://doi.org/10.1007/978-1-4842-3276-7

Library of Congress Control Number: 2017960764

Cover image designed by Freepik

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano
Copy Editor: Mary Behr
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

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. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

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

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484232750. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

For Christy.

The original version of this book was revised. An erratum to this book can be found at https://doi.org/10.1007/978-1-4842-3276-7_13

Table of Contents

About the Author

Andy Leonard is Chief Data Engineer at Enterprise Data & Analytics. He also is an SSIS trainer, consultant, and developer. Andy is a Business Intelligence Markup Language (Biml) developer and BimlHero. He is also a SQL Server database and data warehouse developer, community mentor, engineer, and farmer. Andy is a co-author of *SQL Server Integration Services Design Patterns* and *The Biml Book*.

Acknowledgments

I thank God first for He leads me in right paths for his name's sake (Psalm 23:3). I thank Christy, my lovely bride, and our children—Stevie Ray, Emma, and Riley—for sacrificing some Dad time. Thanks to the awesome team at Enterprise Data & Analytics for their patience and hard work while I wrote: Kent Bradshaw, Nick Harris, and Penny Trupe. I also have an awesome team at Apress: Jill Balzano kept the wheels on the bus going 'round and 'round and Jonathan Gennick is the best editor in the business.

Foreword

I've had the honor and privilege of knowing and working with Andy Leonard for several years. Most of that time has been spent collaborating on projects as independent contractors. We've watched and learned as SSIS has matured through the years. Andy is one of the most knowledgeable and technically savvy people that I've had the pleasure of working with. And, with close to 40 years in the field, I've worked with a lot of people.

This book is filled with best practices we learned through years of trial and error with the practical application of SSIS to solve real problems. We learned by doing and then rethinking to look for better solutions. Things change quickly so we learn new things all the time. We are constantly reevaluating what we've done in the past and comparing it with what we've learned since then.

Data Integration Life Cycle Management with SSIS grew out of the many iterations working on various projects. It's not just about development but also the effort of ongoing maintenance. Understanding both sides is critical in developing processes that work well in the DevOps enterprise. This approach is a methodology that helps make the development and deployment processes more efficient, effective, and predictable. That makes us all a little happier.

This book is also about sharing what has been learned along the journey with others who can benefit from it. Andy is all about that, and it's one of the things that I appreciate most about him. I sincerely hope that you find this book helpful now and for a long to come.

Enjoy!

Kent Bradshaw
Providence Forge, VA
Summer 2017

CHAPTER 1

Introduction to DILM

DevOps is a combination of the words "Development" and "Operations." DevOps is about process improvement, which manifests in faster time to market, higher quality, repeatable automation, and code that is easier to support and maintain. Software testing is a major part of DevOps, starting with unit-testing conducted by software developers. Software testing occurs at all enterprise application tiers (Development, Quality Assurance, User Acceptance, Production, etc.). DevOps developers support Operations by surfacing process instrumentation and building repeatable configurations scripts. Configurations scripting supports higher quality and faster disaster recovery, and rapidly and reliably adding enterprise application tiers.

Software developers follow best practices to build robust enterprise software. For decades, developers have applied a collection of best practices called application lifecycle management, or *ALM*, when building applications. What is ALM? Application lifecycle management includes design principles that support DevOps. An important development concept is *separation of concerns*, design best practices including externalization (parameterization) and decoupling.

Data integration is moving data from one location to another. Data is often collected from disparate sources and loaded into a database to support centralized reporting. The reporting databases are known by different names in different enterprises: operational data store (ODS),

The original version of this chapter was revised. An erratum to this chapter can be found at https://doi.org/10.1007/978-1-4842-3276-7_13

A. Leonard, *Data Integration Life Cycle Management with SSIS*,
https://doi.org/10.1007/978-1-4842-3276-7_1

staging database, central repository, data warehouse (DW), or enterprise data warehouse (EDW), to list a few.

Data integration *is* software development. Data integration lifecycle management, or *DILM*, is the art and science of managing data integration in the modern DevOps enterprise. In this book, I share what I've learned managing SQL Server Integration Services (SSIS) data integration solutions in enterprises large and small.

My goal in writing this book is to teach you how to manage SSIS in your enterprise. As with all other software development platforms, lifecycle management is a best practice with SSIS. As with all other software development, it's possible to practice DevOps with SSIS.

Some History

SQL Server Integration Services (SSIS) was released in November 2005. Development on SSIS started long before, and I'm told the product was originally slated to be dubbed Data Transformation Services (DTS) 2.0. Many assemblies sport the namespace "Dts."

I want to introduce this book by making a few statements about SSIS.

SSIS Is a Software Development Platform

I cannot recall how many times I've seen job postings for database administrators "with SSIS experience."

Database administrator, database developer, and SSIS developer are different roles.

I cringe a little when I read said postings. SSIS is a software development platform and I am a software developer. I am also a database developer. I am *not* a database administrator (DBA). I've tried to do the job of a DBA and failed. Miserably. It's not that I don't appreciate the role of a DBA; I promise I do. Some of my best friends are DBAs. I want to begin by expressing to you this simple truth: DBA, database developer, and SSIS developer are different roles.

SSIS Is an Enterprise Data Integration Engine

SSIS is designed to move data from one location to another, which is the essence of data integration. Data integration can include reshaping, cleansing, and transformation of data; but at its heart, data integration is data relocation. The SSIS Data Flow Task was revolutionary when it was introduced. In my humble opinion, it remains pretty slick technology. The pipeline architecture surfaces most of the levers one needs to tweak to achieve enterprise scale.

Is SSIS the perfect data integration engine for *every* data integration need? No. But it is an amazingly flexible solution to most data integration requirements.

A number of times I've accomplished what-cannot-be-done with SSIS. Years ago I even got into a flame war online with a really smart, internet-famous software developer who wrote a post listing all the things wrong with SSIS. I shared with this individual that I teach people SSIS and all my students know the solution to almost every item listed in the post.

> *Someone recently asked me, "When is Microsoft going to deliver an enterprise data integration engine?" My response? "2005."*
>
> – Andy Leonard

SSIS Is Difficult to Learn

How do I know? I (and others) have made a living for more than a decade teaching people how to use SSIS.

All software platforms have "corners," - quirks and edge cases that the language just isn't the best at managing. SSIS has about 30 corners. I describe the data flow task to students in this manner: "SSIS wants you to *think* like a data flow and thinking like a data flow is hard."

"SSIS wants you to think like a data flow."

Lifecycle Management

SSIS, even at the time of this writing, is difficult to manage in the enterprise lifecycle. Exhibit A is comparing SSIS packages. SSIS is XML-based. XML is a self-describing data format with less respect for order than traditional data stores. For example, the following XML snippets are equivalent:

```
<Book>
    <Title>Data Integration Lifecycle Management</Title>
    <Author>Andy Leonard</Author>
   <Year>2017</Year>
</Book>
<Book>
    <Author>Andy Leonard</Author>
    <Title>Data Integration Lifecycle Management</Title>
   <Year>2017</Year>
</Book>
```

You may look at this example and quip, "But Andy, I can figure out that these data are equivalent just by looking at them." You are correct; examining five rows of data is pretty simple. Imagine looking at five

hundred rows of XML, with tags and attributes in a different order, and you begin to understand the complexity of comparing XML data.

XML is not bad or wrong. XML's semi-structured nature makes SSIS difficult to compare.

Solutions and Credit Where Credit Is Due

The remainder of this book focuses on solutions. With the exception of the BimlExpress Metadata Framework, Kent Bradshaw, Kevin Hazzard, and I developed the solutions contained in this book. Scott Currie and his team at Varigence, Inc. built Business Intelligence Markup Language (Biml) and taught us how to author Biml. Some of what Scott and his team taught us made its way into the BimlExpress Metadata Framework.

A more accurate rendering of the facts is that Kent, Kevin, and I learned what we know from experience and from others. In a very real sense, none of us is *self-taught*. Rather, we are *community-taught*.

Some of these solutions are simply ways of using the technology Microsoft shipped "in the box" with SSIS. Some are best practices. Some are manual and others are automated. Some are free tools and utilities by vendors. Some are free utilities and tools the team at Enterprise Data & Analytics (`entdna.com`), Tudor Data Solutions (`tudords.com`), and DevJourney (`devjourney.com`) have developed; many are part of the DILM Suite (`dilmsuite.com`). One, BimlExpress, is a third-party product from Varigence (`varigence.com`). A couple, SSIS Framework Community Edition and BimlExpress Metadata Framework, are free versions of for-sale implemented solutions that Enterprise Data & Analytics sells as part of consulting engagements.

Do we have all the answers? Goodness no! We have some. Like you, we learn new stuff every day. Here in this book is some of what we know today.

CHAPTER 2

SSIS

This book is not intended to teach you SSIS. If you read this book and work through the examples, you may learn more about SSIS. That is my hope, but it's not my goal in writing this book to teach you SSIS. My goal is teach you how to *manage* SSIS in your enterprise. If you desire to learn SSIS, I recommend the *Stairway to Integration Services at SQL Server Central* (`sqlservercentral.com/stairway/72494/`) for beginners and *SQL Server Integration Services Design Patterns* (`amazon.com/Server-Integration-Services-Design-Patterns/dp/1484200837`) for more advanced learning.

Learning by example is best. In this chapter, you will build an SSIS project for demonstration purposes. This SSIS project will include one SSIS package, a connection manager, variables, project parameters, and package parameters. You will use this SSIS project, DILMSample, throughout the remainder of this book. The SSIS project is built in SQL Server Data Tools (SSDT). Please see Appendix A for links to the tools and utilities you will use throughout this book.

I will discuss data integration instrumentation and messaging to surface log messages during the execution of the SSIS package. These messages serve people troubleshooting failed executions and surface important data integration instrumentation metadata.

A. Leonard, *Data Integration Life Cycle Management with SSIS*,
https://doi.org/10.1007/978-1-4842-3276-7_2

The Demo

Open SSDT and create a new SSIS project named DILMSample. When SSDT creates a new SSIS project, it loads a default SSIS project. The default SSIS project includes

- Project parameters, stored in the `Project.params` file
- A virtual folder for project-scope connection managers
- A virtual folder for package parts
  - A virtual folder for control flow package parts
- A virtual folder for miscellaneous items
- A default (empty) SSIS package named `Package.dtsx`

Rename `Package.dtsx` as `SimplePackage.dtsx`, as shown in Figure 2-1.

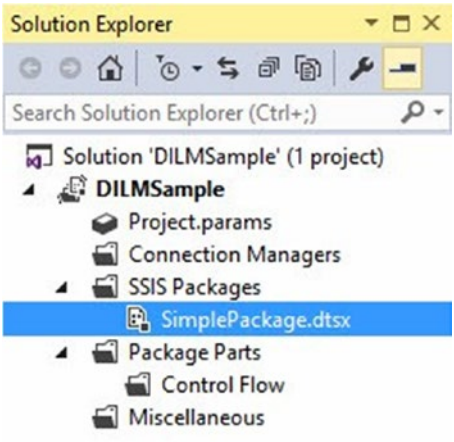

Figure 2-1. *The DILMSample SSIS project*

Renaming the package communicates intent; leaving the default package name communicates laziness.

Adding Package Parameters

Since we plan to use this project and package to demonstrate lifecycle management, let's add parameters. To start, click the Parameters tab on the `SimplePackage.dtsx` package. Add two package parameters with the settings shown in Table 2-1.

Table 2-1. *Package Parameter Settings*

Name	Data type	Value	Sensitive	Required	Description
IntPkgParam	Int32	42	False	False	
StringPkgParam	String	Hi There!	False	False	

When completed, your Parameters tab should appear similar to that shown in Figure 2-2.

Figure 2-2. *Package parameters*

You add these demo parameters at the package scope. *Scope* is an important concept in software development *and* lifecycle management. The "Dev" part of DevOps focuses on building code that is easily manageable by Operations. How does scope help? Later you will *externalize* parameter values into Transact-SQL (T-SQL) scripts. Operations personnel will ultimately manage externalized values by deploying and maintaining T-SQL scripts. One goal of DevOps development is to communicate as much information as possible to

Operations people, especially Operations people who have no idea how SSIS works. Scoping parameters and variables to their proper level (no higher or lower) communicates where and how parameter values are used in the SSIS application, thus shedding some light on the "black box" of the data integration project.

Adding Project Parameters

Open the Project Parameters window by double-clicking the `Project.params` artifact in Solution Explorer, shown in Figure 2-3.

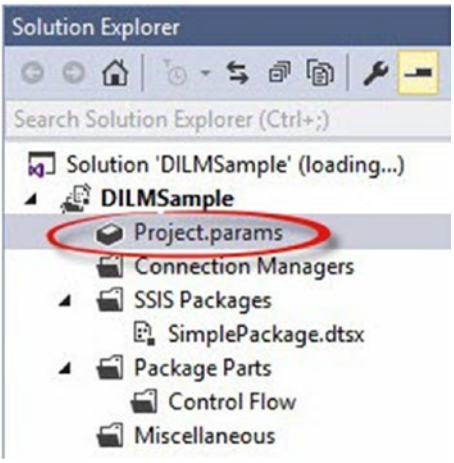

Figure 2-3. *Project parameters*

Add two project parameters with the settings from Table 2-2.