AUGUST 20, 2018

USING SSIS CATALOG COMPARE

PRODUCT DOCUMENTATION

Version	Date	Author(s)	Summary
0.1	18 Jul 2016	Andy Leonard	Initial draft
1.5.2 (beta)	22 Dec 2016	Andy Leonard	Updated for v2.0 release
2.0.0	5 Jan 2017	Andy Leonard	Updated to include CatCompare
3.2.2.1	20 Aug 2018	Andy Leonard	Updated for version 3

Contents

0 PREFACE

Data integration is moving data from one location to another. Sometimes data integration involves transforming the data in some way. The size of the data may change as data is aggregated or expanded. The shape of the data may change as data is placed into a data model that facilitates a function different from the function in which it is stored. Data values may be updated to make the data easier to understand in reports.

Let's face it, data integration is hard. Data integration requires knowledge of the source and destination data platform as well as the data integration engine.

SQL Server Integration Services (SSIS) is one data integration engine. When SSIS was introduced, there was precious little execution support shipped out of the box. With the introduction of the SSIS Catalog with SSIS 2012, Microsoft took a big step in remedying execution support.

The SSIS Catalog is a good, but not quite complete, solution for enterprise data integration support. SSIS Catalog Compare seeks to complete the functionality supplied in the SSIS Catalog.

SSIS Catalog Compare supports Azure Data Factory SSIS Integration Runtime.

Andy

1 Connect to an Instance of an SSIS Catalog

When you open SSIS Catalog Compare, it will appear similar to Figure 1:

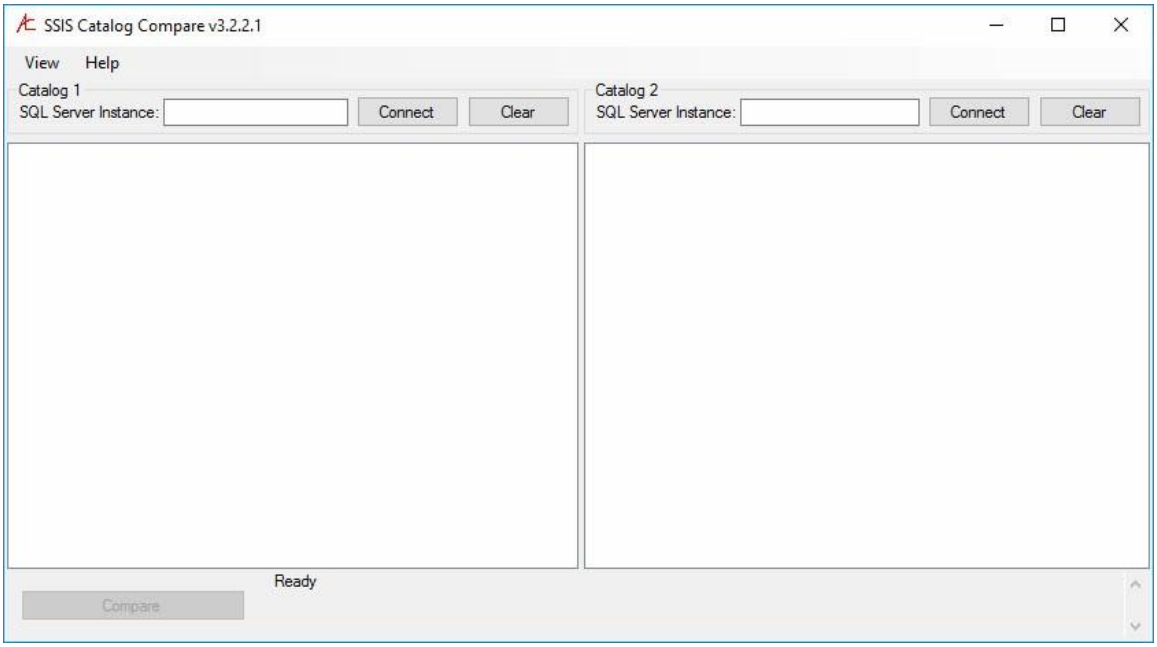


Figure 1

You must use Integrated Security (Windows Authentication) to connect to an on-premises SSIS Catalog instance. Connecting to local instances of the SSIS Catalog via SQL Server Login is not supported.

*SSIS Catalog Compare respects the security model of the SSIS Catalog. This has several implications. One important implication is SSIS Catalog Compare follows SSIS and SSIS Catalog conventions for Sensitive values. In short: SSIS Catalog Compare should **never** print or save values marked as Sensitive in the SSIS Catalog, Catalog Environments, SSIS Projects, or SSIS Packages.*

For Azure Data Factory SSIS Integration Runtime, SSIS Catalog Compare supports connection via SQL Logins.

For *blanket administrative* rights, a user must be a member of the sysadmin or ssis_admin role. Specific permissions may be granted using the SSIS Catalog's built-in security model (see SSIS Catalog Access Control Tips for more information). In SSIS 2016, a user may *view* artifacts in an SSIS Catalog if they are a member of the ssis_logreader role.

For more information, please see the MSDN article: Integration Services Roles (SSIS Service).

To learn more about Azure Data Factory SSIS Integration Runtime, please see the MSDN article: Integration runtime in Azure Data Factory.

1.1 Connect with Integrated Security

To connect using Integrated Security (Windows Authentication):

- Enter the name of a SQL Server instance that hosts an SSIS Catalog
- Click the Connect button or press the Enter key

Figure 2

1.2 Connect to Azure Data Factory SSIS Integration Runtime

To connect to Azure Data Factory SSIS Integration Runtime using a SQL Login:

- Enter the name of the server which hosts Azure Data Factory SSIS Integration Runtime
- Click the Connect button or press the Enter key
- When prompted, enter SQL Login credentials
- Click the Connect button of press the Enter key

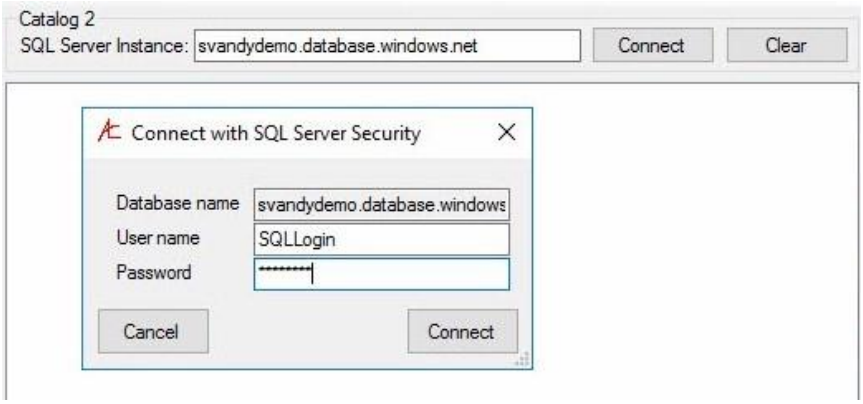


Figure 3

2 View SSIS Catalog Artifacts

SSIS Catalog Compare provides a rich view of the SSIS Catalog. The initial view will appear similar to that shown in Figure 4:

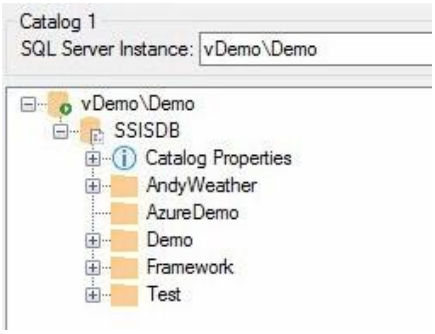


Figure 4

The initial view presents a hierarchical view of the SSIS Catalog which includes:

- Instance - the name of the SQL Server instance which hosts the SSIS Catalog
  - SSISDB – the name of the SSIS Catalog
    - Catalog Properties – a list of SSIS Catalog properties

- Catalog Folders – a list of Catalog Folders contained in the SSIS Catalog.

This view is similar to the view of Catalog Folders available in the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer, shown in Figure 5:

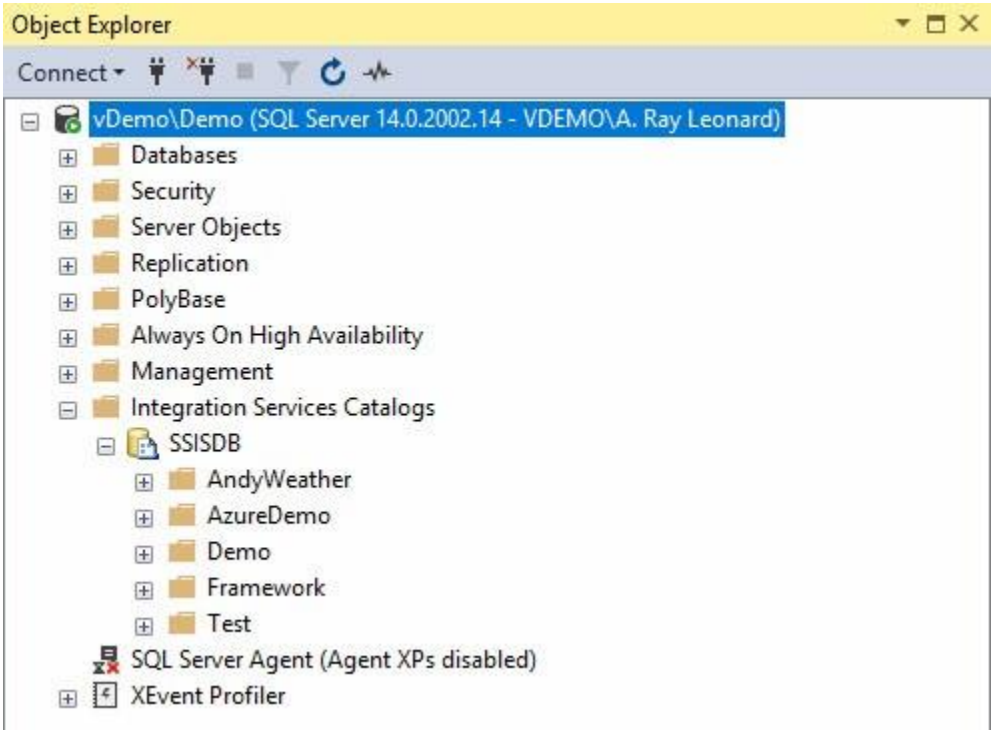


Figure 5

2.1 Exploring a Catalog Folder

Exploring the Stage Catalog Folder in SSIS Catalog Compare, we see two virtual folders named Environments and Projects as shown in Figure 6:

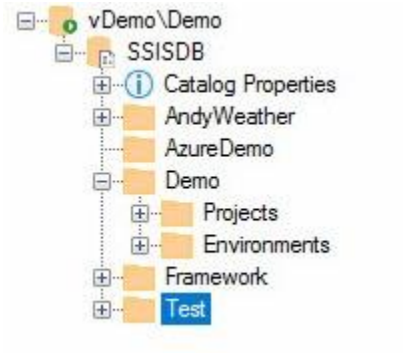


Figure 6

Again, this view is similar to that presented in the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer, shown in Figure 7:

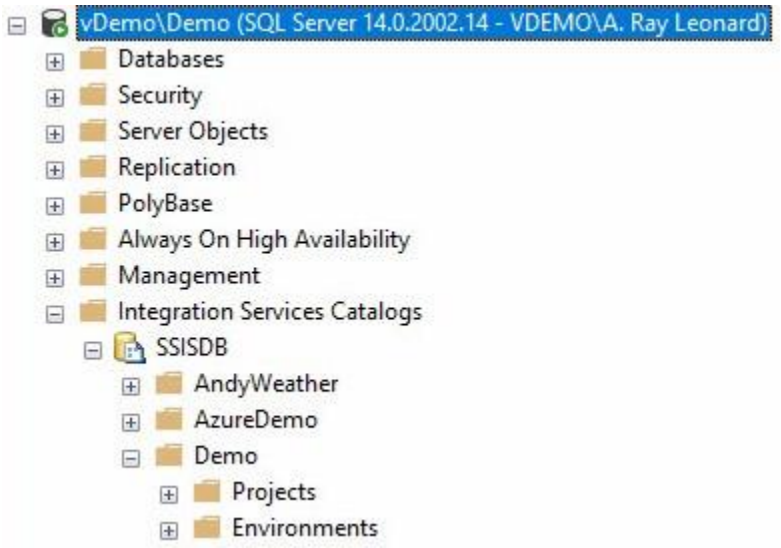


Figure 7

The views of the SSIS Catalog presented by the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer and SSIS Catalog Compare begin to diverge once we drill beneath the Projects and Environments virtual folders.

2.1.1 Exploring the Environments Virtual Folder

The view of the Environments virtual folder from the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer is shown in Figure 8:

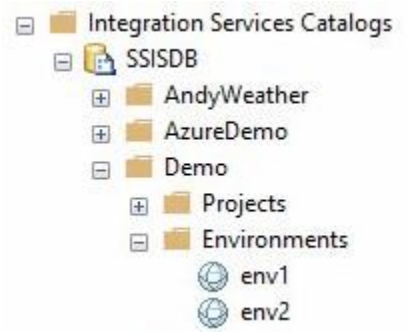


Figure 8

The view of the Environments virtual folder from SSIS Catalog Compare is shown in Figure 9:

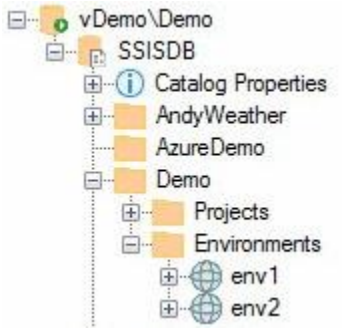


Figure 9

Note the presence of child nodes for each Catalog Environment in the SSIS Catalog Compare view. Expand the child nodes to view Environment Variables and their values as shown in Figure 10:

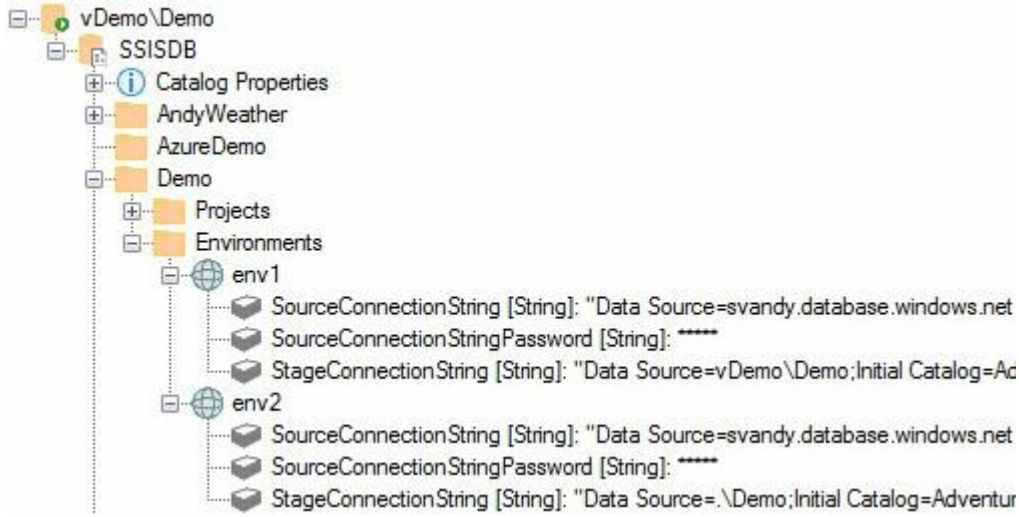


Figure 10

Can you view Catalog Environment Variables and values using the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer? Yes.

Double-click the Catalog Environment node, or right-click the node and click Properties, as shown in Figure 11:

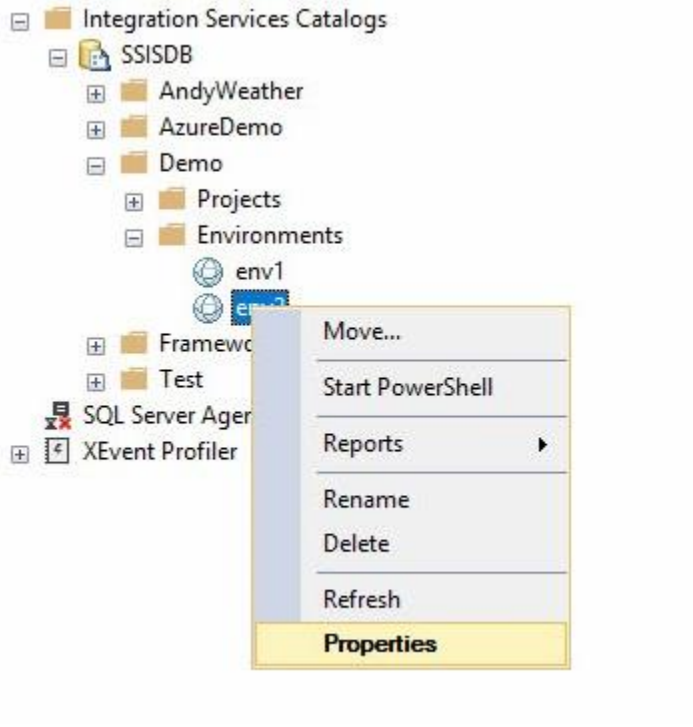


Figure 11

When the Environment Properties window opens it will appear as shown in Figure 12:

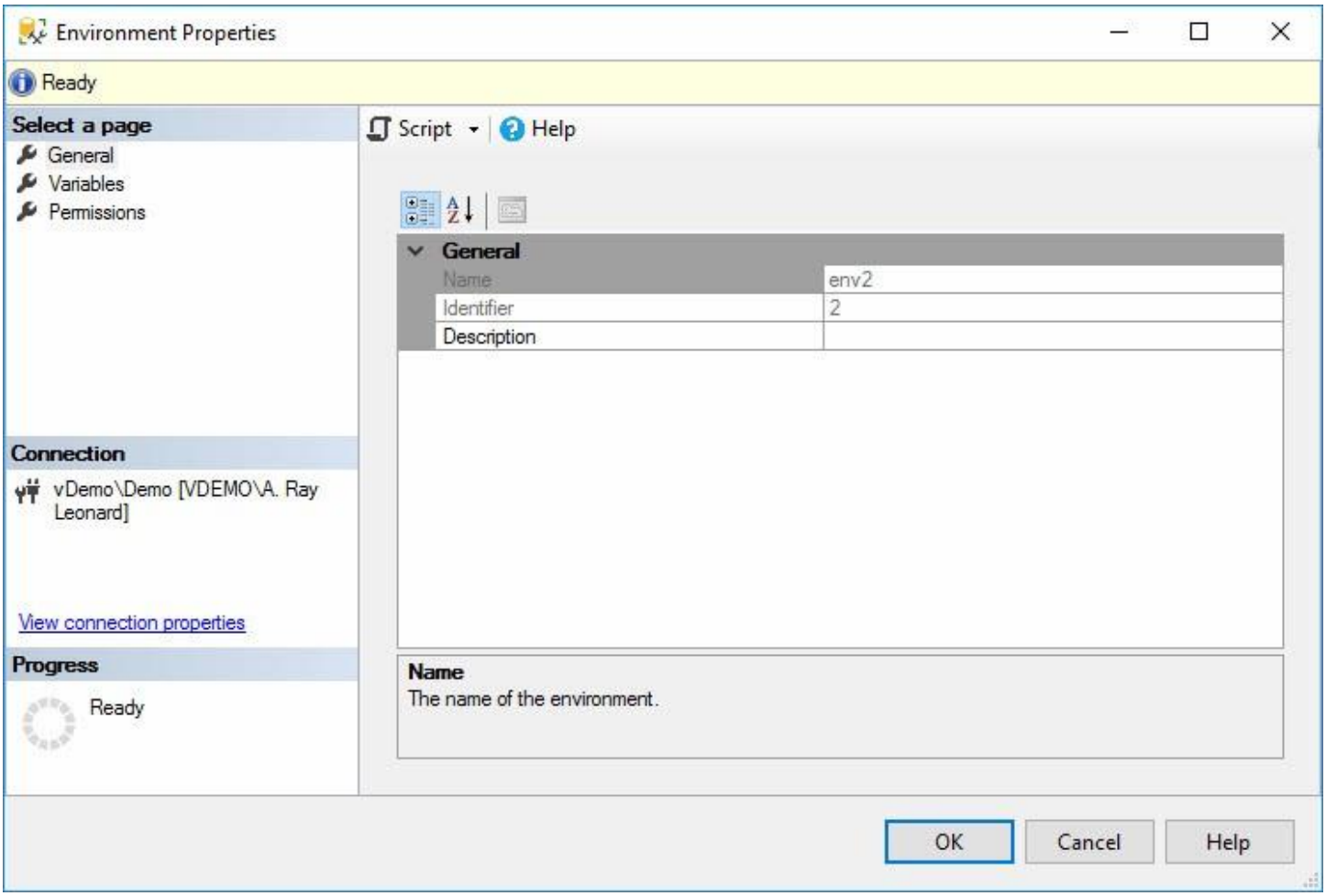


Figure 12

Click the Variables page in the listbox located in the upper left of the form to view the Catalog Environment Variables and their values as shown in Figure 13:

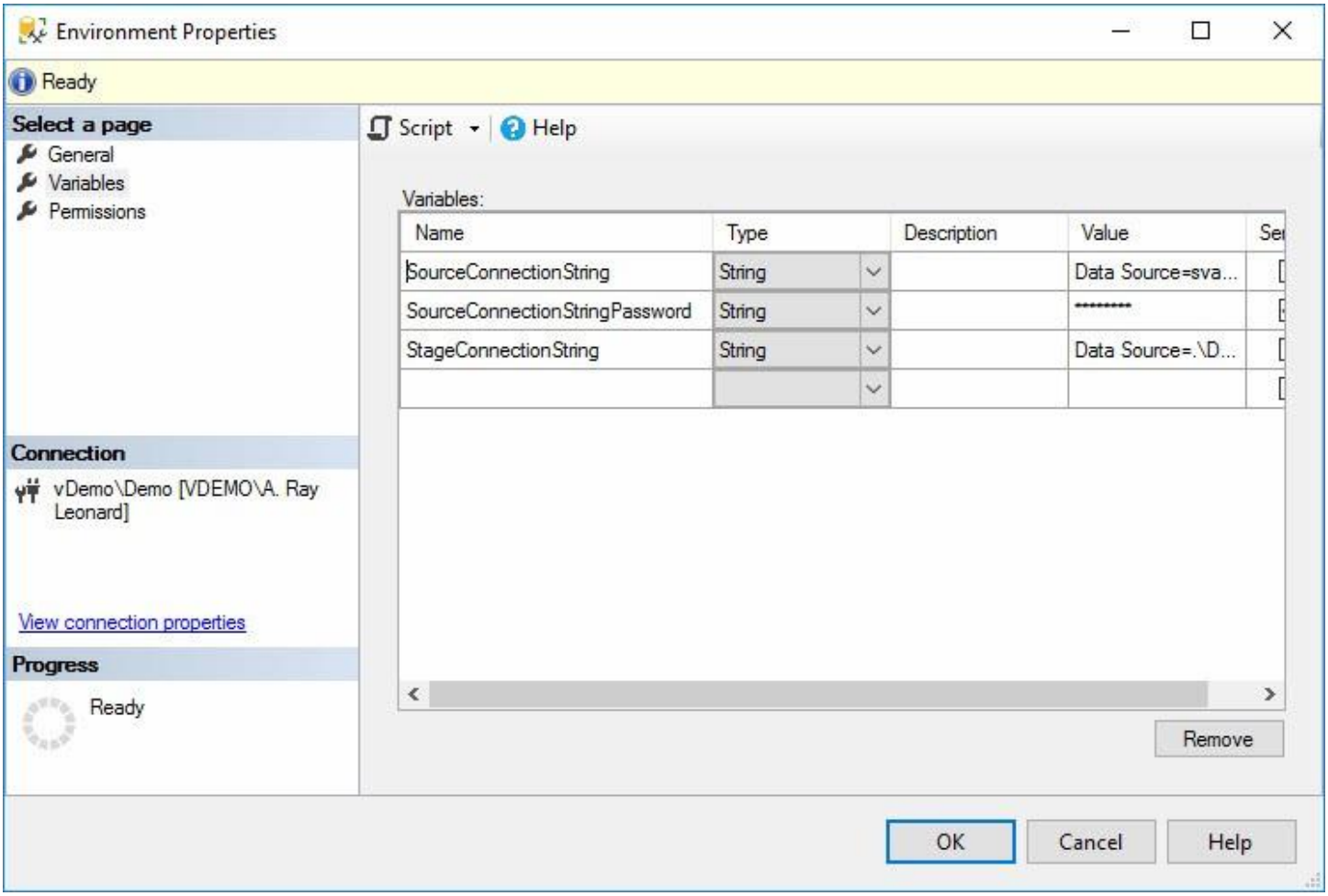


Figure 13

2.1.1.1 Update an SSIS Catalog Environment Variable

In the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer, you can modify the Value, Description, and Sensitive properties of a Catalog Environment Variable from the Variables page of the Environment Properties window shown in Figure 13.

You can modify the value of a Catalog Environment Variable in SSIS Catalog Compare by right-clicking the variable node and then clicking “Update Value…” as shown in Figure 14:

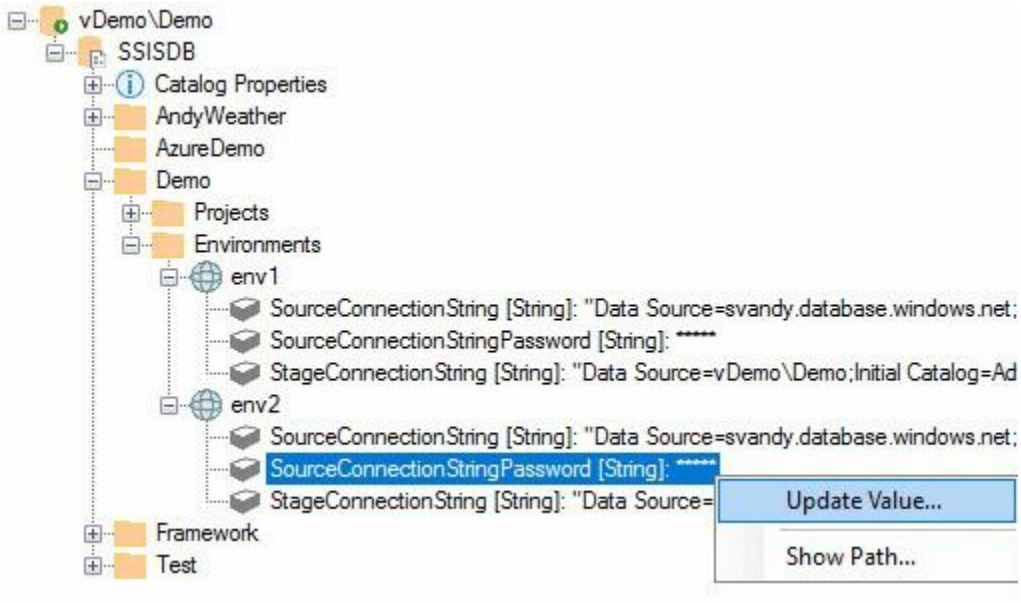


Figure 14

The Update Value dialog displays, allowing you to update the Catalog Environment Variable Value, Description, and Sensitive properties as shown in Figure 15:

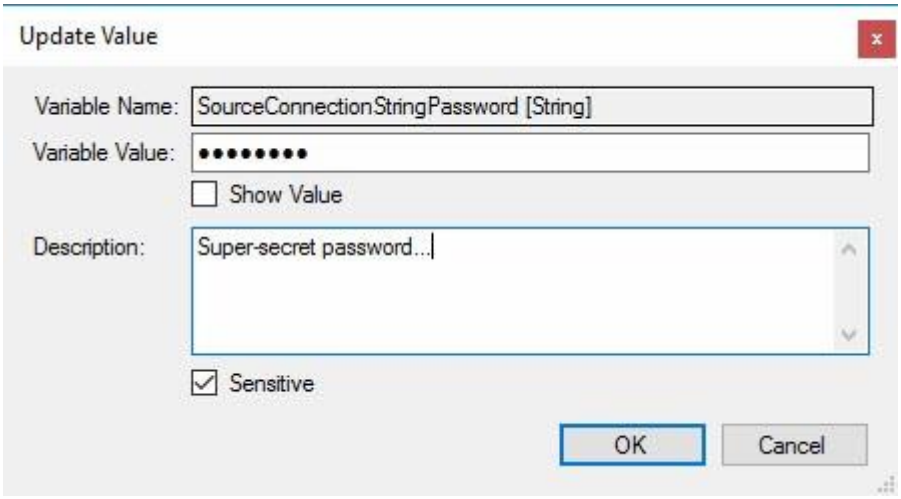


Figure 15

After making edits to the value, click the OK button to store the changes to the SSIS Catalog. The updated value will appear as shown in Figure 16:

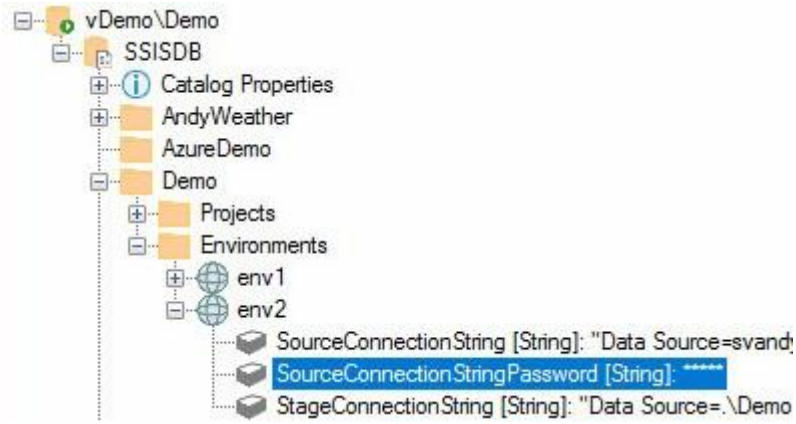


Figure 16

2.1.2 Exploring the Projects Virtual Folder

The view of the Projects virtual folder from the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer is shown in Figure 17:

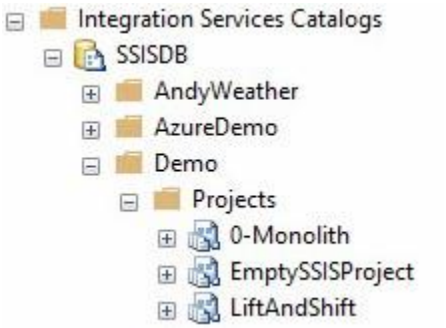


Figure 17

The view of the Projects virtual folder from SSIS Catalog Compare is shown in Figure 18:

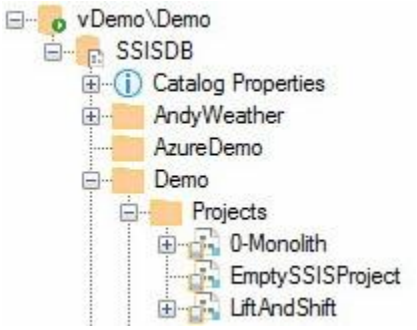


Figure 18

The views of the Projects virtual folder presented by SSIS Catalog Compare and the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer are almost identical. Note SSIS Catalog Compare's view indicates there are no child nodes for the SSIS project named "EmptySSISProject," which contains no SSIS packages. You can see

this in the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer if you expand the "EmptySSISProject" node as shown in Figure 19…

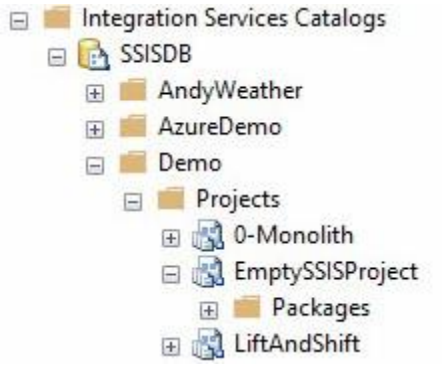


Figure 19

… and then click to expand the Packages virtual folder node as shown in Figure 20:

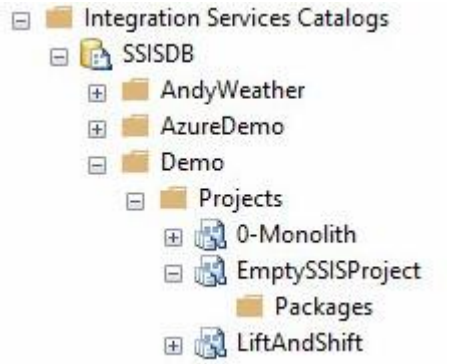


Figure 20

The Integration Services Catalogs node in the SQL Server Management Studio Object Explorer now indicates there are no SSIS packages in the SSIS project named "EmptySSISProject."

Turning our attention to a project that contains packages, the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer displays SSIS packages under the Packages virtual node as shown in Figure 21:

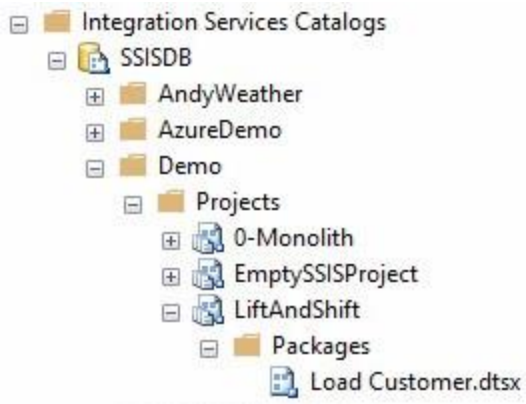


Figure 21

SSIS Catalog Compare presents a richer view of artifacts in the SSIS Catalog related to SSIS projects. Depending on the configuration of the project in the SSIS Catalog, there are zero-to-three virtual folders visible under the Project node, listed here and shown in Figure 22:

1. Packages
2. Project Connections
3. Project Parameters
4. Project References

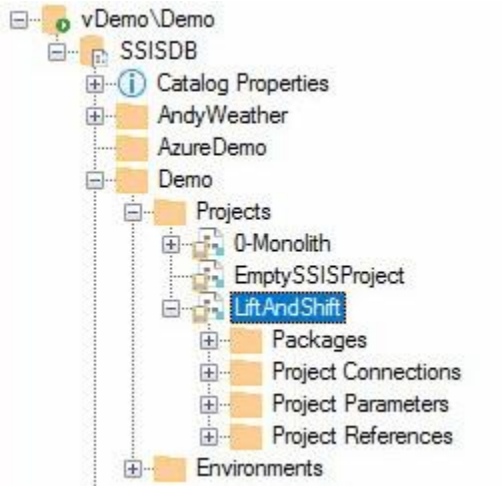


Figure 22

> *Note: As demonstrated with the SSIS project named "EmptySSISProject," SSIS Catalog Compare will present 0 child nodes if an SSIS Project stored in an SSIS Catalog contains no configuration artifacts (see Figure 18).*

2.1.2.1 Exploring the Packages Virtual Folder

As with the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer, the Packages virtual folder presents the SSIS Packages deployed to the SSIS Catalog instance, shown in Figure 23:

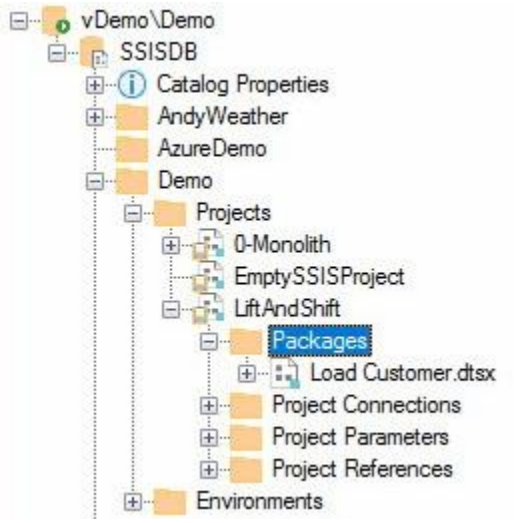


Figure 23

SSIS Catalog Compare presents a richer view of artifacts in the SSIS Catalog related to SSIS packages. Depending on the configuration of the package in the SSIS Catalog, there are one-to-three virtual folders visible under the Package node, listed here and shown in Figure 24:

1. Package Properties
2. Package Connections
3. Package Parameters
4. Package References

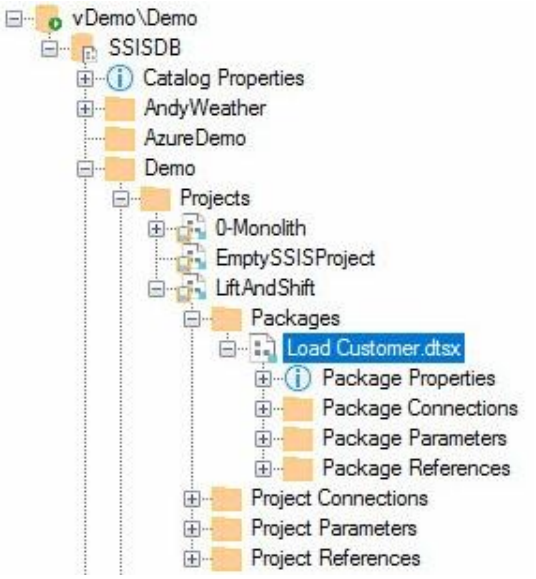


Figure 24

2.1.2.1.1 Exploring the Package Properties Virtual Folder

The Package Properties virtual folder presents a list of package properties for the given SSIS Package including Package Version, Package Version Comments, and Package GUID – as shown in Figure 25:

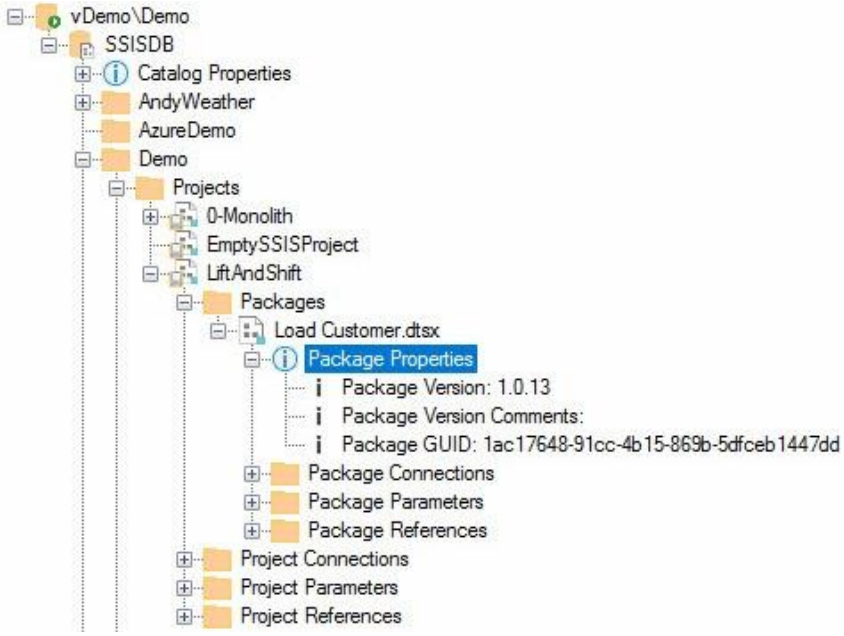


Figure 25

2.1.2.1.2 Exploring the Package Connections Virtual Folder

The Package Connections virtual folder surfaces a list of connection managers configured at the SSIS package level, as shown in Figure 26:

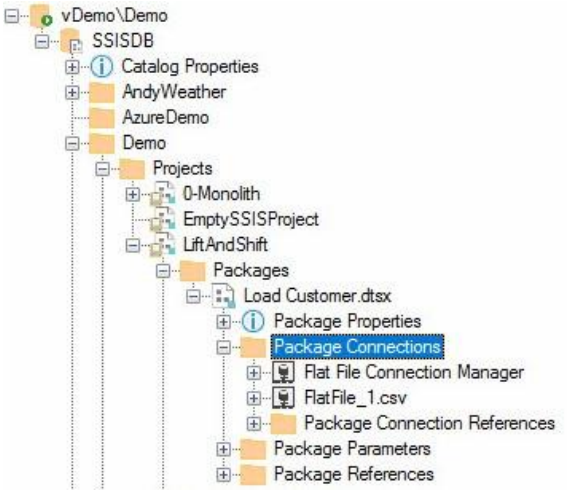


Figure 26

2.1.2.1.2.1 Exploring a Connection in the Package Connections Virtual Folder

Each property of package connection managers in the SSIS Package is surfaced as package connection sub-nodes in the Package Connections virtual folder. Connection properties mapped via Reference Mappings are underlined and the label surfaces the name of the EnvironmentVariable, as shown in Figure 27:

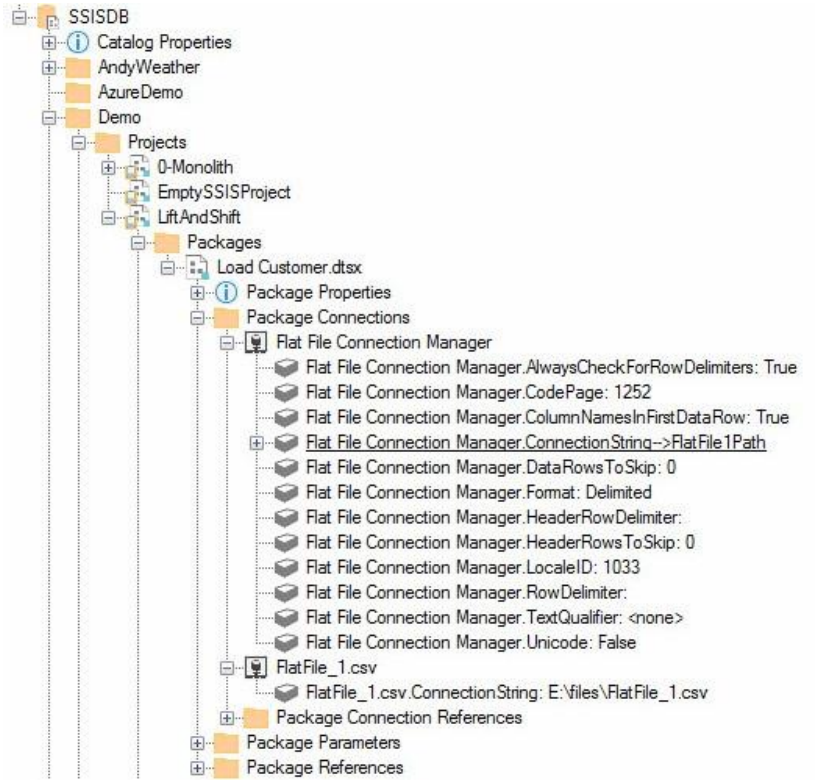


Figure 27

2.1.2.1.2.1.1 Exploring Values Everywhere for a Connection Property in the Package Connections Virtual Folder

Values Everywhere surfaces SSIS package connection manager properties that are mapped via SSIS Catalog References. In SSIS Catalog Compare, these relationships are identified as *Reference Mappings*. Values Everywhere first surfaces a node for each Reference. References represent a relationship between an SSIS Catalog Environment and an SSIS project or package deployed to an SSIS Catalog. Reference Mappings define the consumption of an Environment Variable by an SSIS Parameter. Figure 28 shows the first level of Values Everywhere for Package connection manager reference mappings – one node for each Reference:

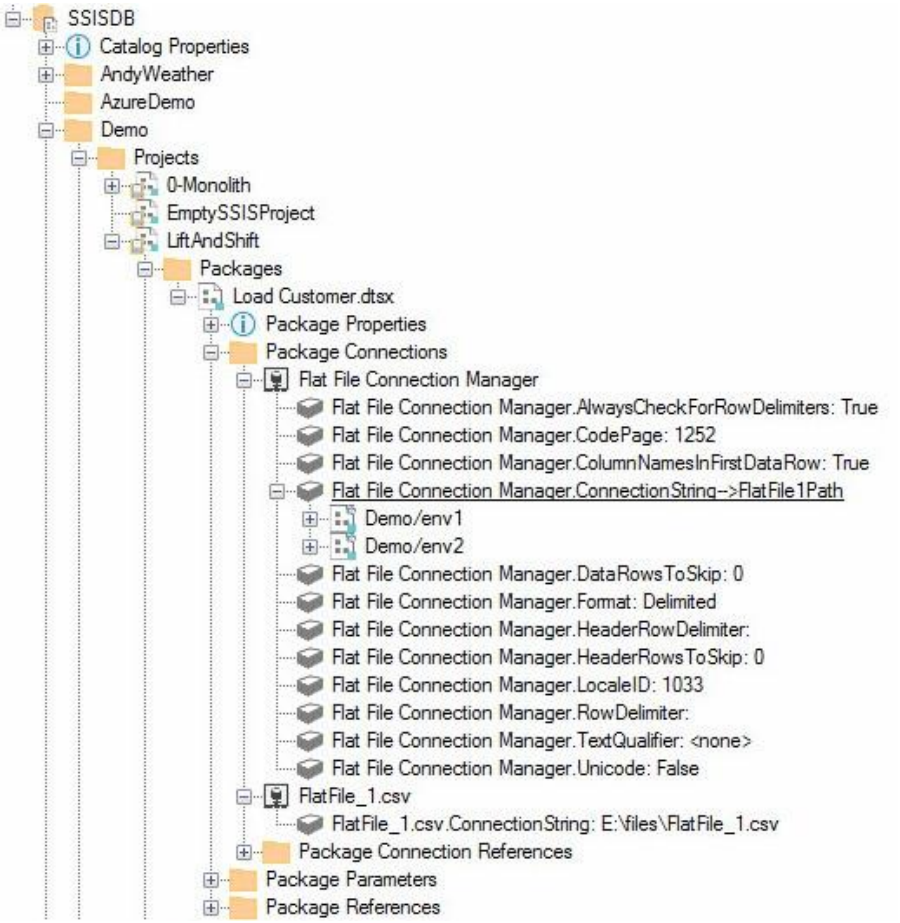


Figure 28

2.1.2.1.2.1.1.1 Exploring Values Everywhere Parameter Values for a Connection Property in the Package Connections Virtual Folder

Beneath each reference node, Values Everywhere displays the reference-mapped environment variable value. Values Everywhere surfaces values at the point of consumption in a reference mapping, as displayed in Figure 29:

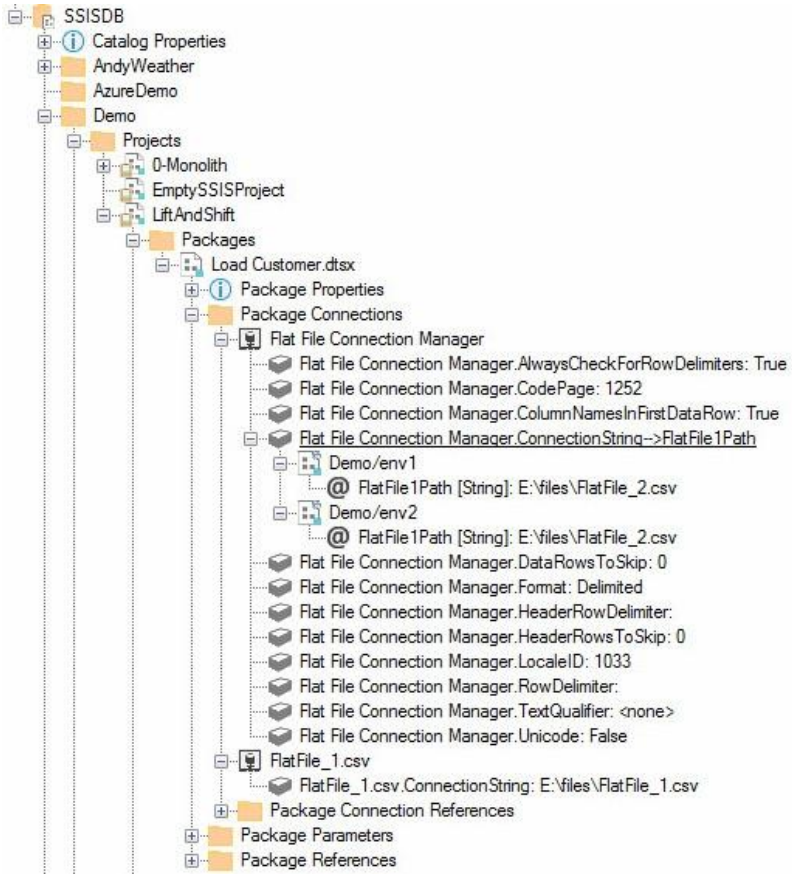


Figure 29

2.1.2.1.3 Exploring the Package Parameters Virtual Folder

The Package Parameters virtual folder presents a list of package parameters for the given SSIS Package, as shown in Figure 30:

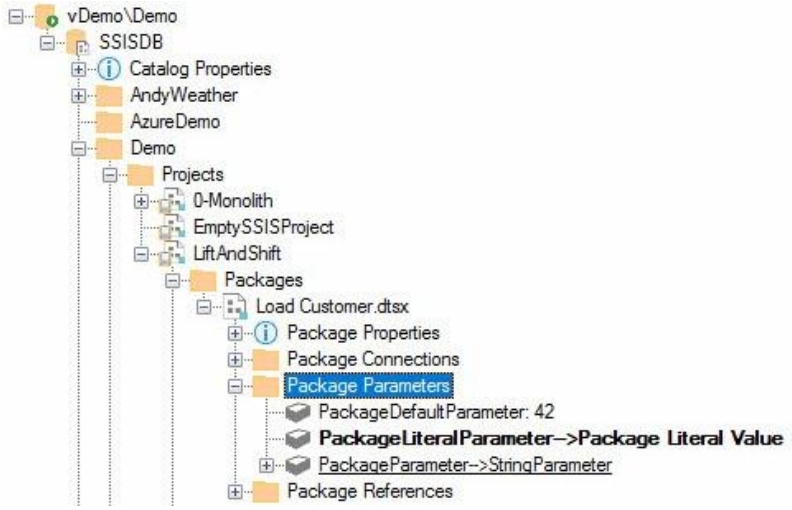


Figure 30

There are three sources for package parameter values:

1. Design-time default – the value entered for the package parameter by the developer. Deign-time defaults are denoted by *no* text decoration of the package parameter node font.
2. Overridden Value Mapping – a value assigned to the package parameter in one instance of an SSIS Catalog. Overridden value mappings are denoted by **bold** text decoration of the package parameter node font. The name of the package parameter is followed by a mapping indicator (→) which is in turn followed by the overridden value.
3. Reference Mapping – a value stored in a Catalog Environment Variable. Reference mappings are denoted by underlined text decoration of the package parameter node font. The name of the package parameter is followed by a mapping indicator (→) which is in turn followed by the name of the Catalog Environment Variable.

2.1.2.1.3.1 Design-time Defaults

Design-time defaults are the values configured in the SSIS Package when the package is deployed to the SSIS Catalog. These values are stored with the SSIS Package. There is no text decoration of the package parameter node font. There is no mapping indicator (→) included in the package parameter node text. There is only the name and design-time default value of the package parameter as shown in Figure 31:

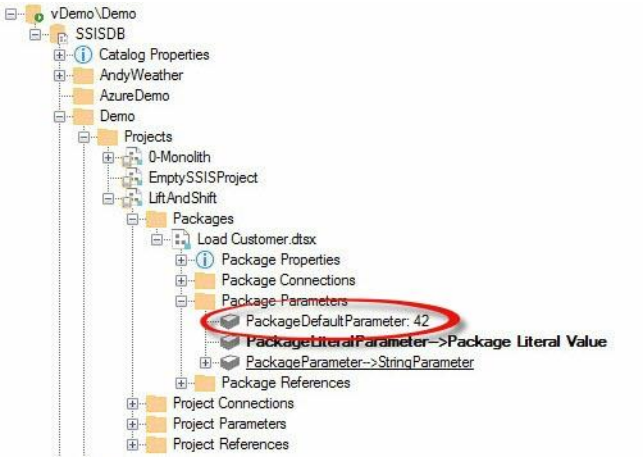


Figure 31

2.1.2.1.3.2 Overridden Value Mapping

The SSIS Catalog allows package parameters to be overridden. Overridden value mappings are called "Literal" values in SSIS Catalog parlance and can be thought of as package parameter values that are "hard-coded" in a single instance of an SSIS Catalog. The overridden values are not stored in the SSIS Package, they are stored in a table (internal.object_parameters) in the SSISDB database. The package parameter node font is decorated **bold**. The package parameter name is followed by a mapping indicator (→). The mapping indicator is followed by the value of the override as shown in Figure 32:

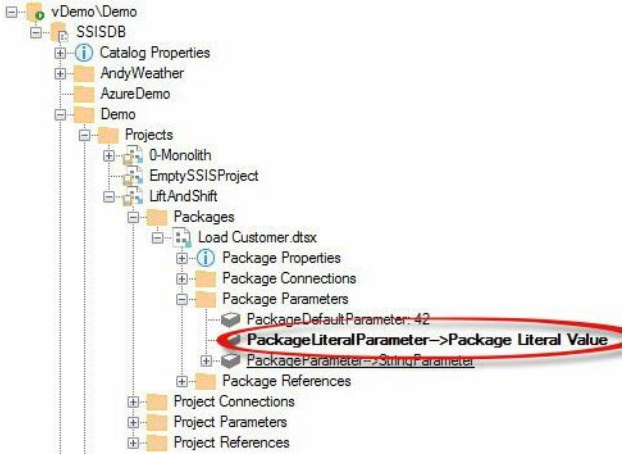


Figure 32

2.1.2.1.3.3 Reference Mappings

Reference Mappings are complex. The Environment Variable value is a property of a Catalog Environment Variable. A Catalog Environment is a collection of zero-to-many Catalog Environment Variables. A *Reference* is, well, a reference from an SSIS Project (or Package) to a Catalog Environment. A *Reference Mapping* is the winding path from an SSIS Project Parameter (or SSIS Package Parameter) through the Project (or through the Package and then the Project), through the Reference, through the Catalog Environment, to the Catalog Environment Variable's value property. One way to represent the relationships in a Reference Mapping is shown in Figure 33:

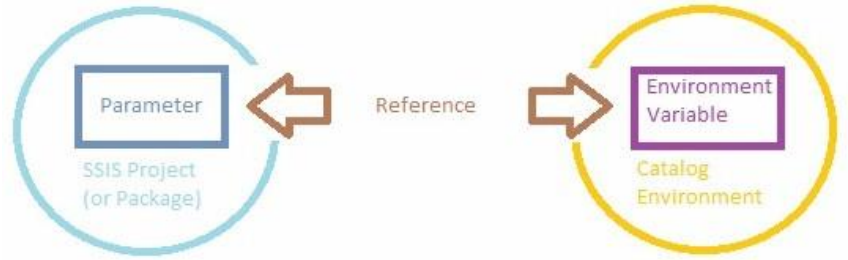


Figure 33

Values mapped from references are not stored in the SSIS Package. As with overridden value mappings, they are stored in a table named internal.object_parameters in the SSISDB database. The package parameter node font is underlined. The package parameter name is followed by a mapping indicator (→). The mapping indicator is followed by the name of the Catalog Environment Variable as shown in Figure 34:

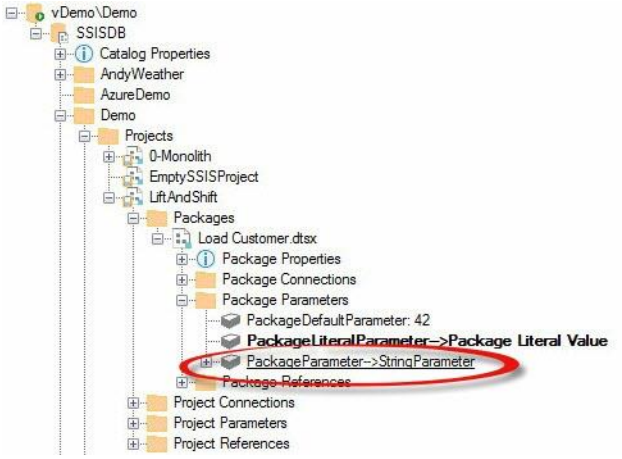


Figure 34

Values Everywhere surfaces reference mapping values where they are configured. Expanding the reference-mapped node (underlined) displays a list of References as shown in Figure 35:

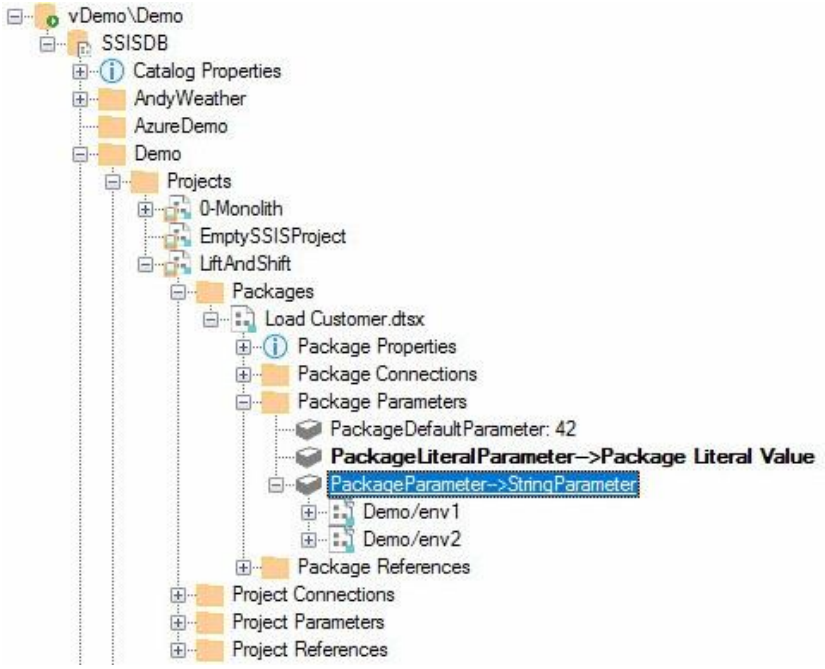


Figure 35

Expanding each Reference node – named for the Catalog Environment which it references – surfaces the value of the Catalog Environment Variable, as shown in Figure 36:

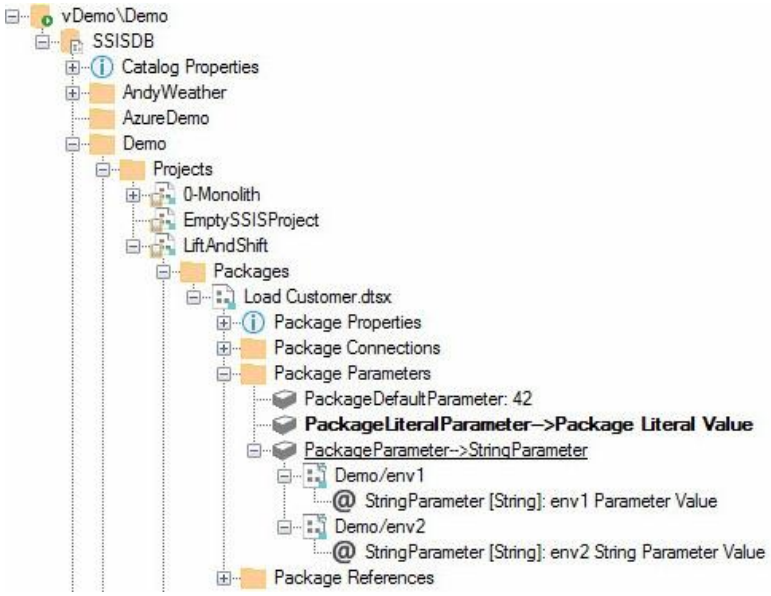


Figure 36

You can configure package parameters using the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer by right-clicking a package and clicking "Configure…" as shown in Figure 37:

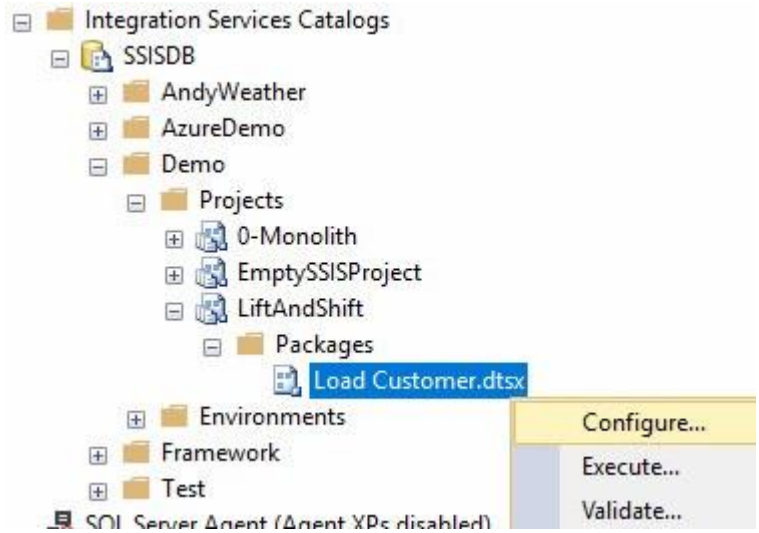


Figure 37

The Configure <*Package Name*> dialog displays, showing the Parameters page and the Parameters tab by default. You can configure the value property of the package parameter by clicking the ellipsis as shown in Figure 38:

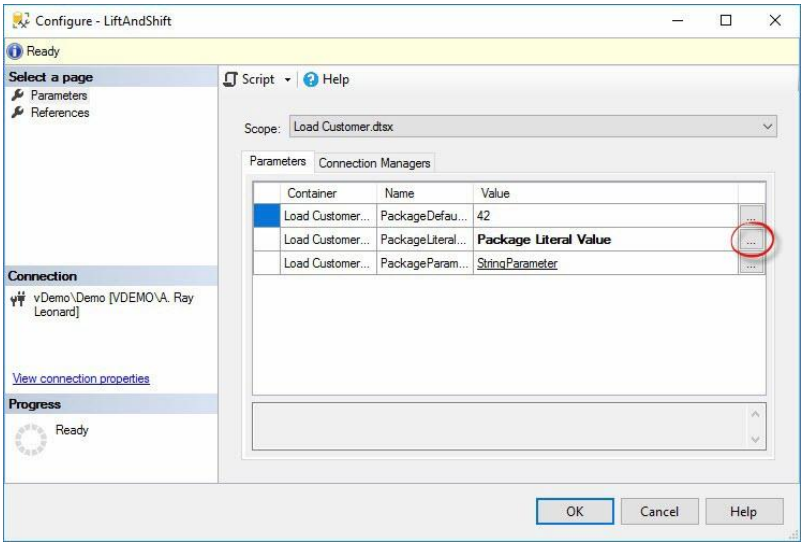


Figure 38

The Set Parameter Value dialog displays as shown in Figure 39:

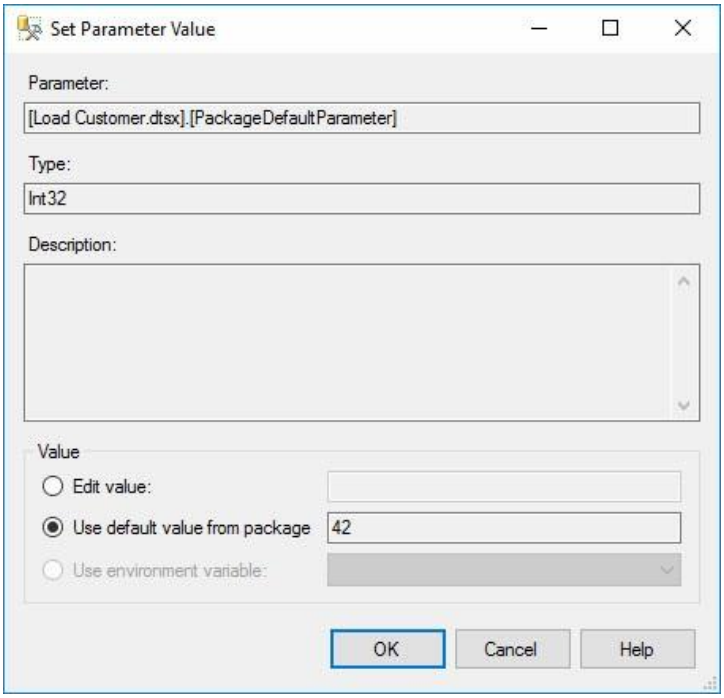


Figure 39

The value may be overridden here using the “Edit value” option. Note the design-time default value remains stored with the SSIS Package and is accessible for configuration using the “Use default value from package” option. The “Use environment variable” option is disabled even though the package displays configured References on the References page as shown in Figure 40:

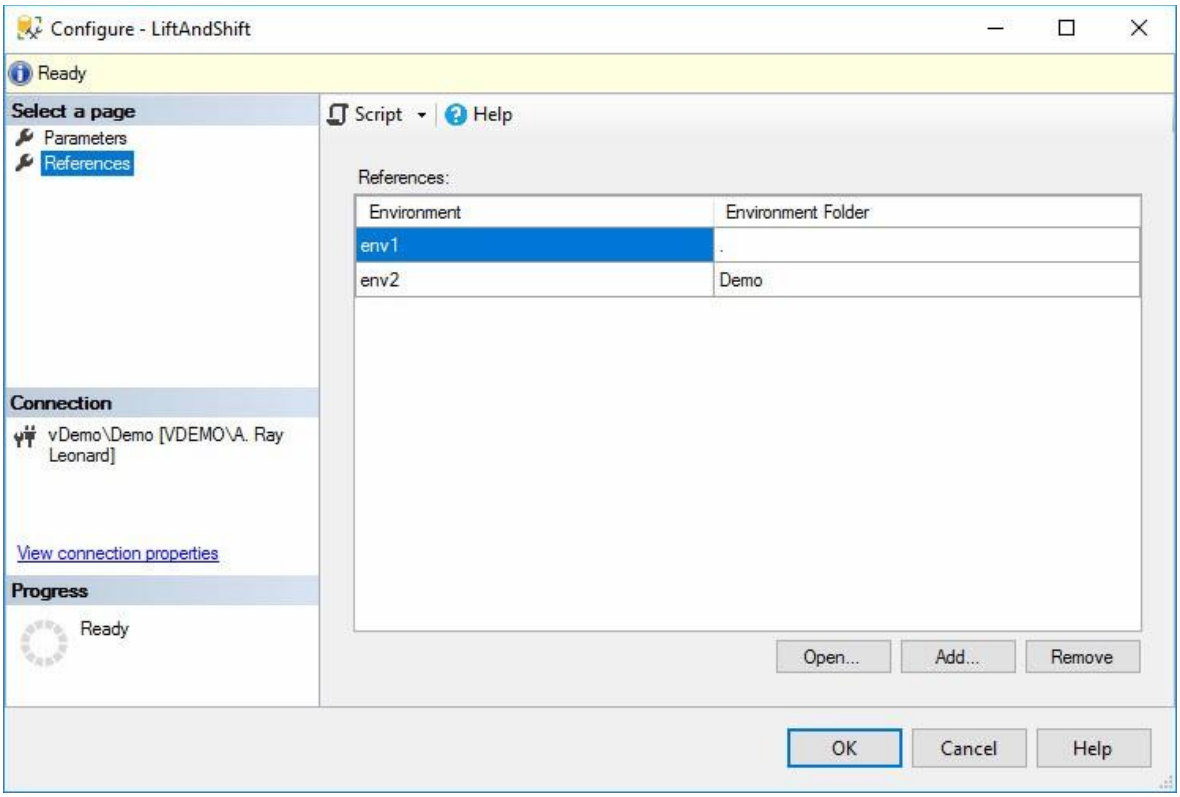


Figure 40

Why can’t we configure a reference mapping for the package parameter? Because there are no Catalog Environment Variables of the same data type defined in any of the Catalog Environments referenced by the package. The SSIS Catalog checks first and disables the option if there are no valid choices available.

The remainder of the package parameters are accessible via the Connection Managers tab on the Parameters page as shown in Figure 41:

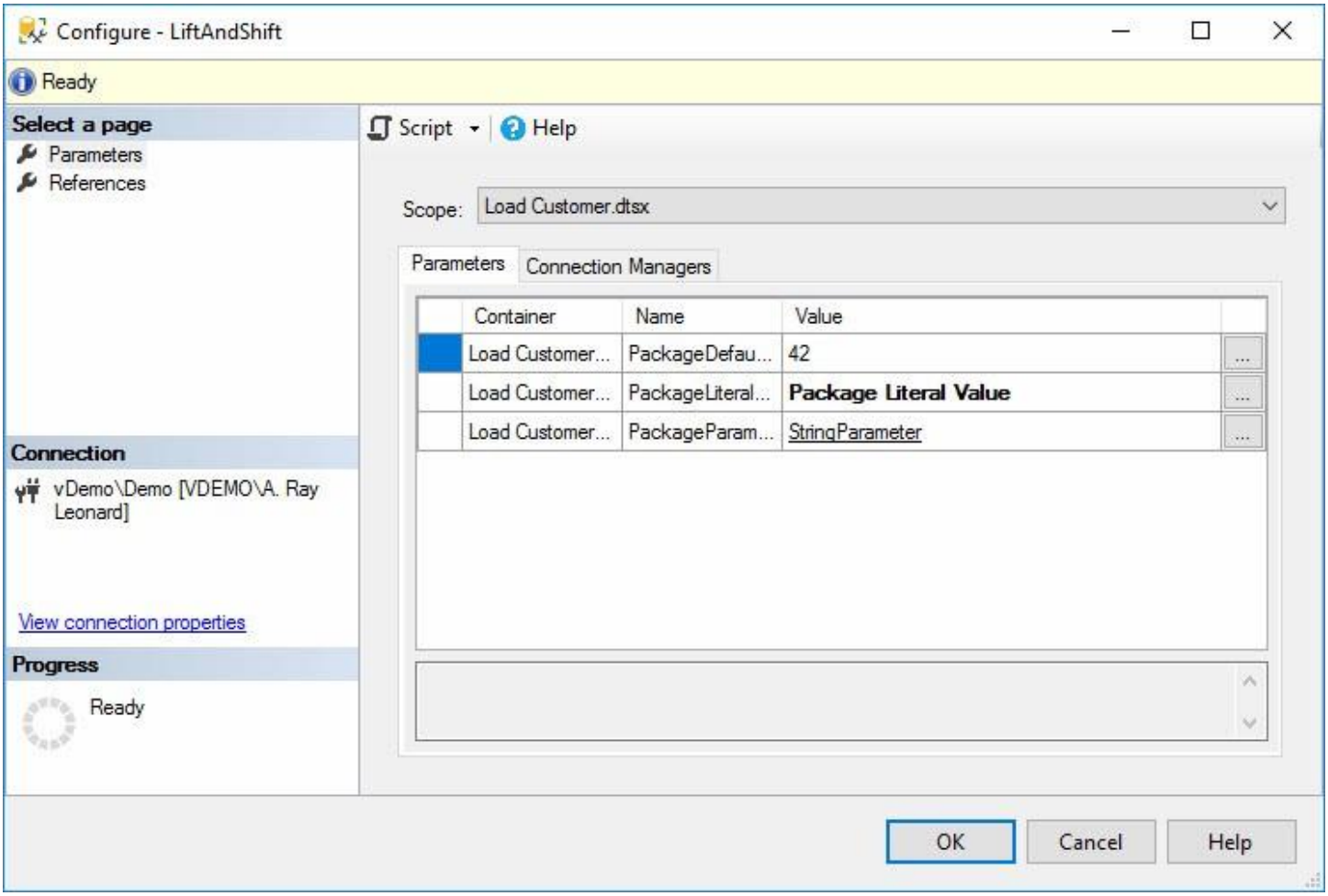


Figure 41

Note SSIS Catalog Compare reflects the text decorations used by the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer.

2.1.2.1.4 Exploring the Package References Virtual Folder

The Package References virtual folder presents a list of package references for the given SSIS Package, as shown in Figure 42:

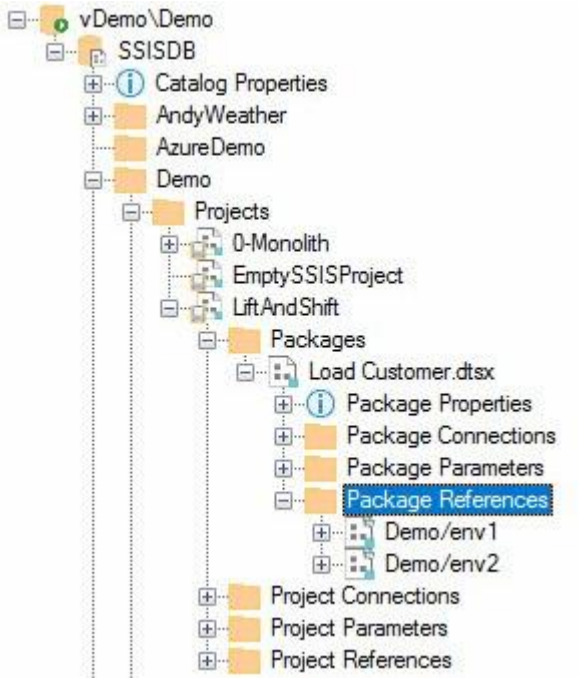


Figure 42

The Package References virtual folder lists only the Catalog Environments for which Reference Mappings exist at the package level. Figure 42 reflects the Integration Services Catalogs node in the SQL Server Management Studio Object Explorer version of Package References. While Figure 42 is *technically* correct, it is not an accurate reflection of the Catalog Environments *in use* by the Package's parameters.

Expanding the Package Reference node presents a list of package parameters mapped via package parameter reference mappings. The combined result of the SSIS Catalog Compare presentations of Package Parameters and Package References is a comprehensive view of the parameter value externalization – at a glance – as shown in Figure 43:

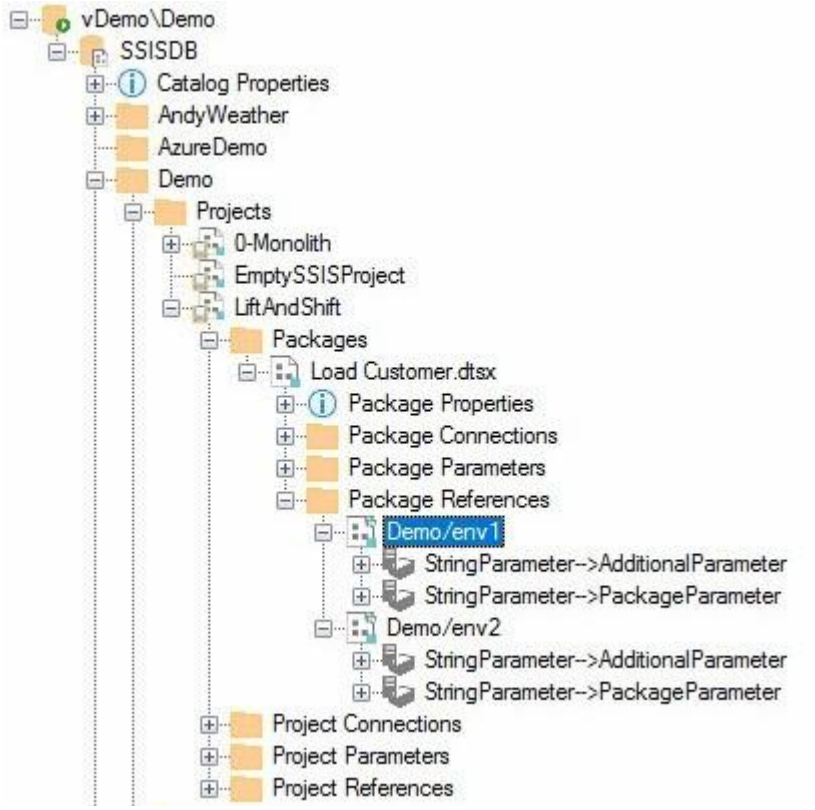


Figure 43

Please note: An Environment Variable may be consumed to override more than a single Package parameter. Figure 43 demonstrates the Environment Variable named StringParameter is used in two reference mappings at the package level – overriding the package parameters named PackageParameter and AdditionalParameter.

Values Everywhere surfaces the values of the Catalog Environment Variables – visually displaying these values at the point where they are used, as shown in Figure 44:

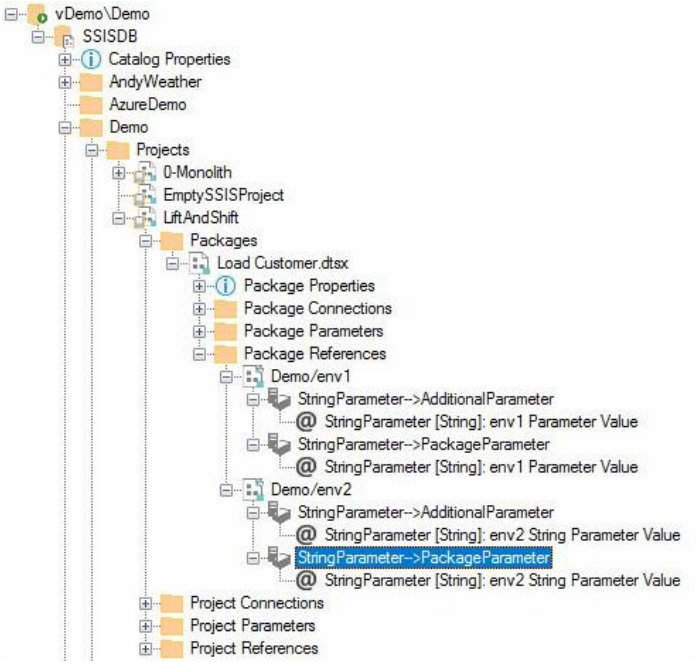


Figure 44

2.1.2.1.5 Externalization Visibility in SSIS Catalog Compare

From the Package Parameters collection, SSIS Catalog Compare presents a package parameter mapped via a reference mapping to an environment variable. From the Package References collection, SSIS Catalog Compare presents the same configuration information.

In *both* locations, the Values Everywhere feature displays the value of the SSIS Catalog Environment Variable that is mapped via reference mapping to the parameter.

3 COMPARE CATALOG ARTIFACTS

SSIS Catalog Compare is designed to compare two SSIS Catalog instances. Compare operations are permitted once two instances are loaded into the treeview controls. Once two SSIS Catalog instances are loaded, the Compare button is enabled as shown in Figure 45:

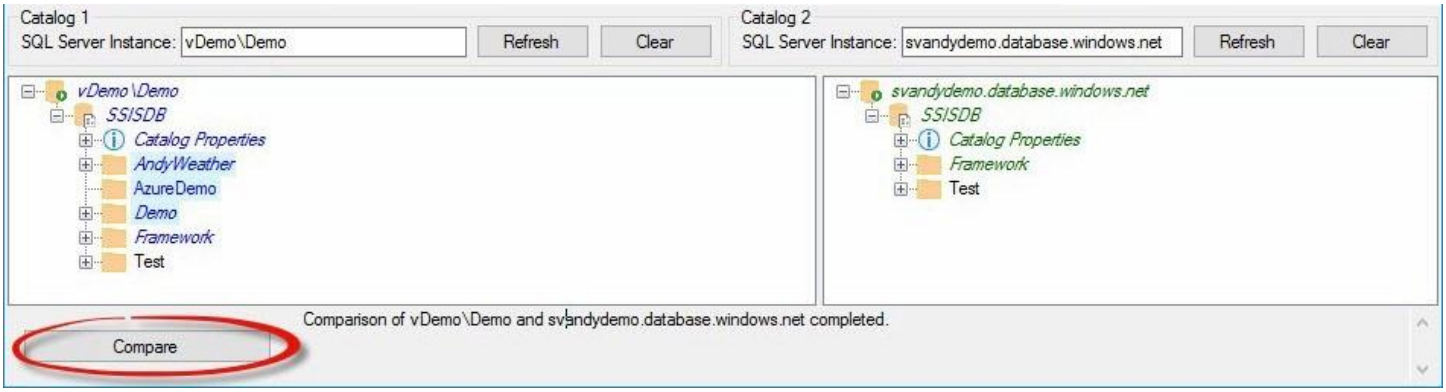


Figure 45

When compared, items that are missing or different show up highlighted as shown in Figure 46:

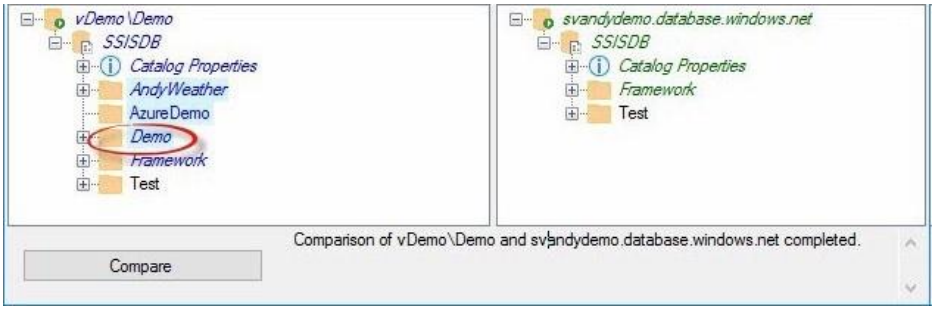


Figure 46

The parent nodes of missing or different items are displayed with *italics* font as shown in Figure 47:

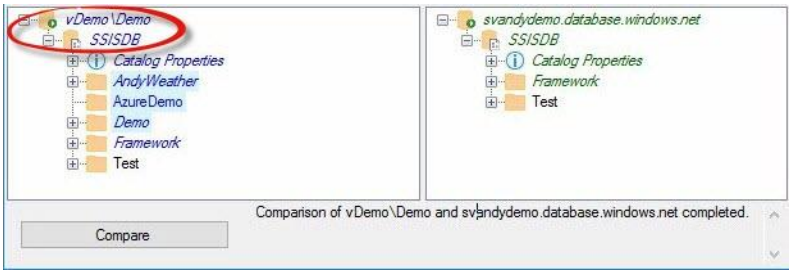


Figure 47

Items for which no changes are detected are shown denoted with a white background and no text decoration as shown in Figure 48:

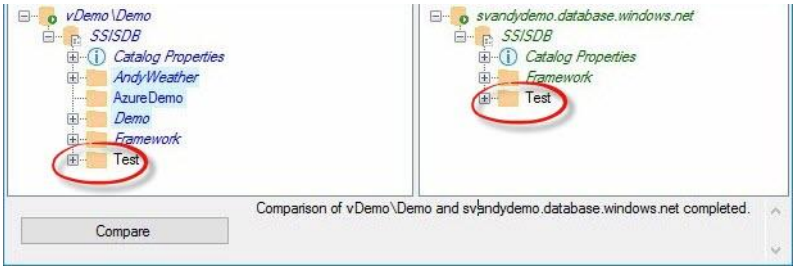


Figure 48

You can right-click nodes containing differences and expand only the differences as shown in Figure 49:

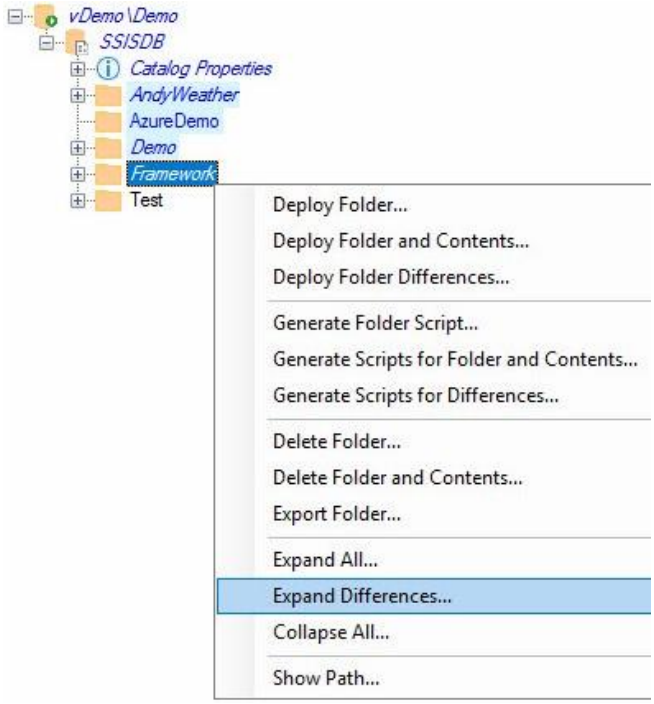


Figure 49

Only “different or missing” nodes are highlighted, as shown in Figure 50:

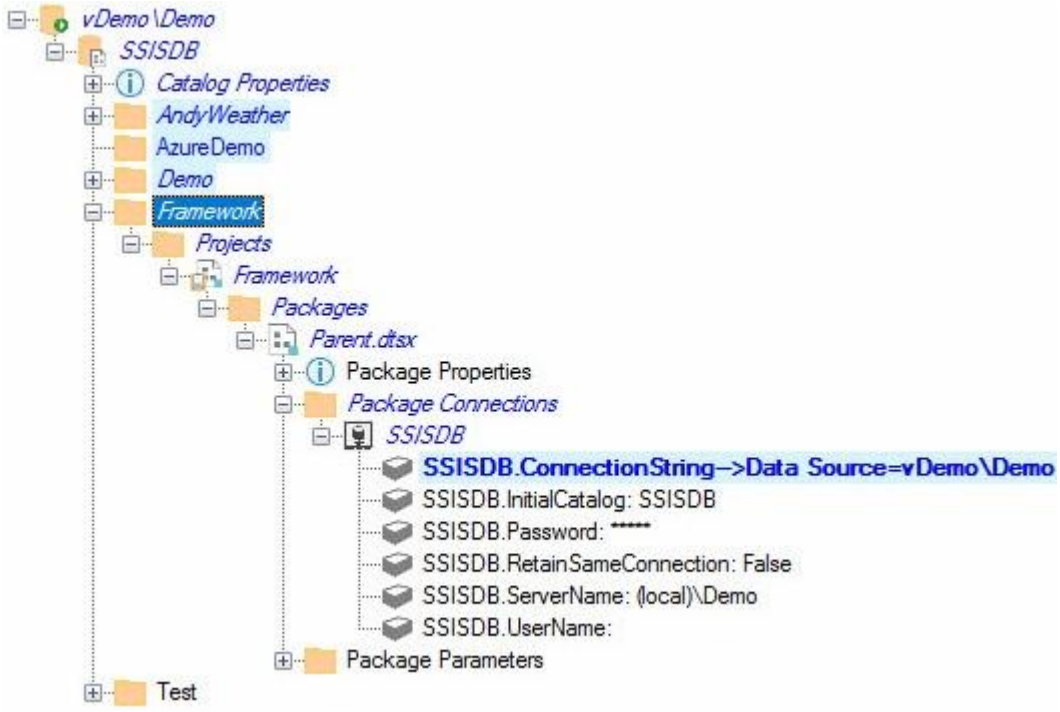


Figure 50

One can collapse all expanded nodes by right-clicking and clicking ‘Collapse All” as shown in Figure 51:

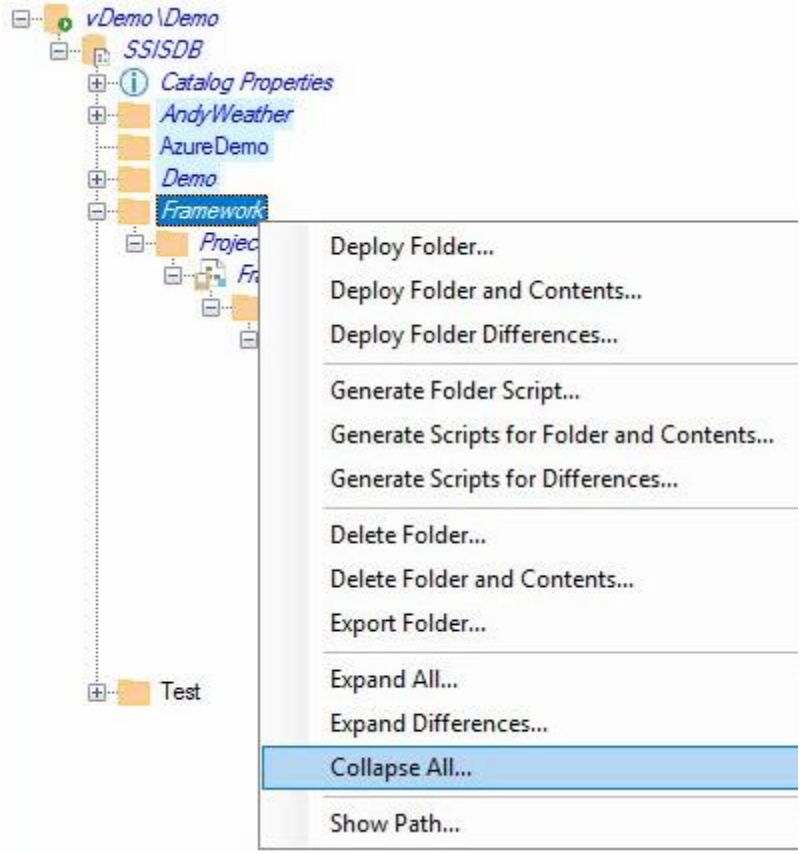


Figure 51

Once collapsed, the node appears as shown in Figure 52:

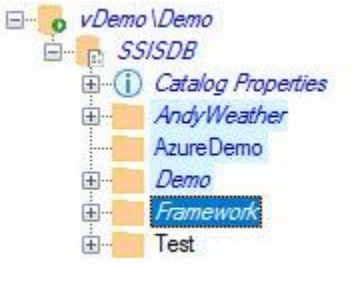


Figure 52

All child nodes can be displayed by right-clicking the node and clicking “Expand All” as shown in Figure 53:

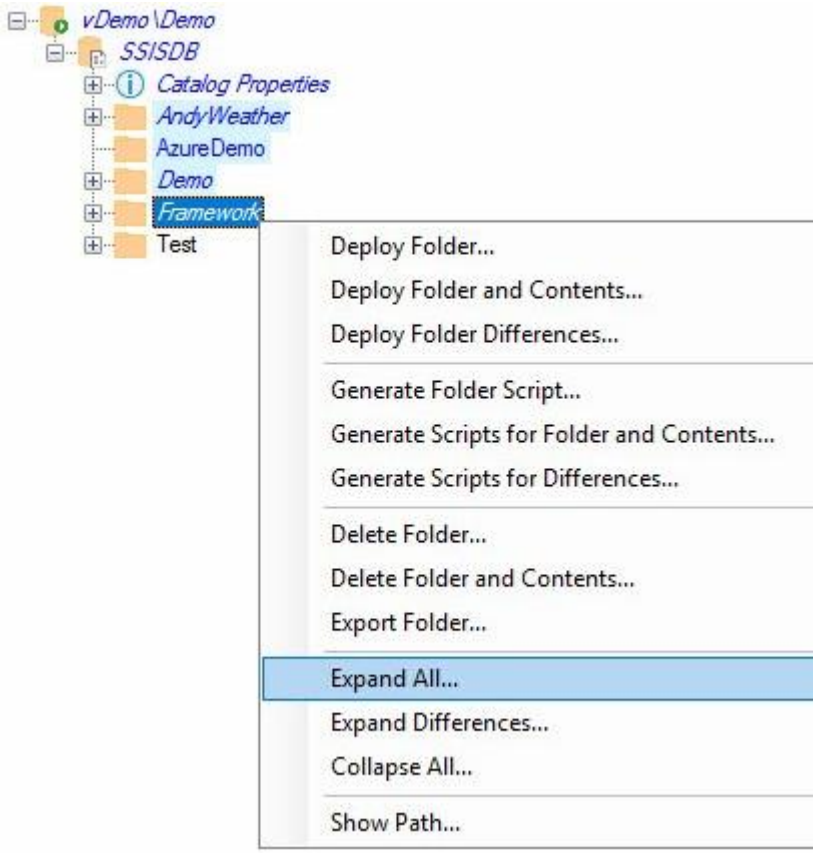


Figure 53

An expanded (All) folder appears similar to that shown in Figure 54:

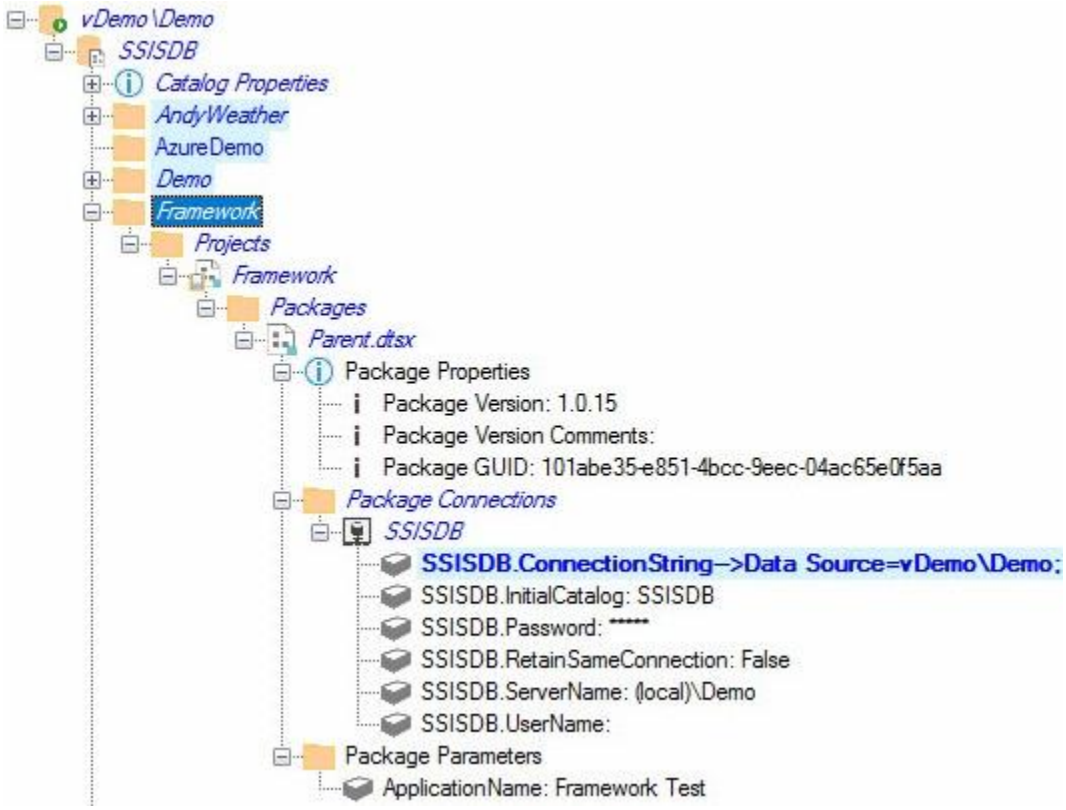


Figure 54

After a compare operation has been executed, users can right-click the Compare button and click "Refresh Both TreeViews and Compare…" as shown in Figure 50:

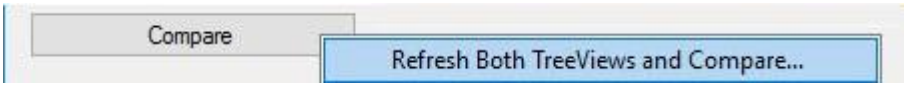


Figure 55

The Compare operation is designed to display objects that are different from their counterparts in another SSIS Catalog instance, and objects that are present in one SSIS Catalog instance and missing from another.

4 SCRIPT CATALOG ARTIFACTS

SSIS Catalog Compare provides scripting functionality for an entire SSIS Catalog instance. Individual scripts are generally categorized into the following SSIS Catalog artifact categories:

1. Folders
2. Projects
3. Literals
4. Environments
5. References

4.1 GENERATE CATALOG SCRIPT

To generate a Catalog script, right-click the Catalog node (named SSISDB) and then click "Generate All Catalog Scripts" as shown in Figure 55:

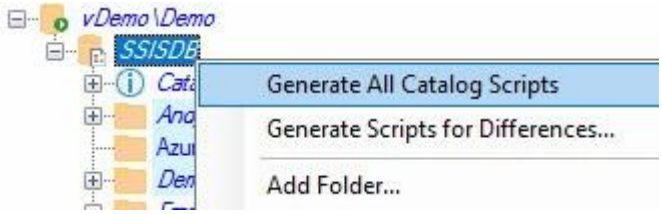


Figure 55

Scripts are generated for classes of SSIS Catalog artifacts, including:

- Folders
- Projects (ISPAC files)
- Package Parameter Literals
- Package Connection Literals
- Project Parameter Literals
- Project Connection Literals
- Environments (includes Environment Variables)
- Package References (includes Reference Mappings)
- Package Connection References (includes Reference Mappings)
- Project References (includes Reference Mappings)
- Project Connection References (includes Reference Mappings)

In the file system, scripts a generated in a folder structure that mimics the SSIS Catalog structure as shown in Figure 56:

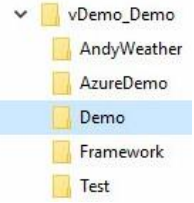


Figure 56

A file-system-friendly instance name is the topmost directory, representing the SSIS Catalog. The SSIS Catalog file system directory contains sub-directories – one for each SSIS Catalog Folder. Each Catalog Folder sub-directory contains scripts and ISPAC files that represent SSIS Catalog artifacts, as shown in Figure 57:

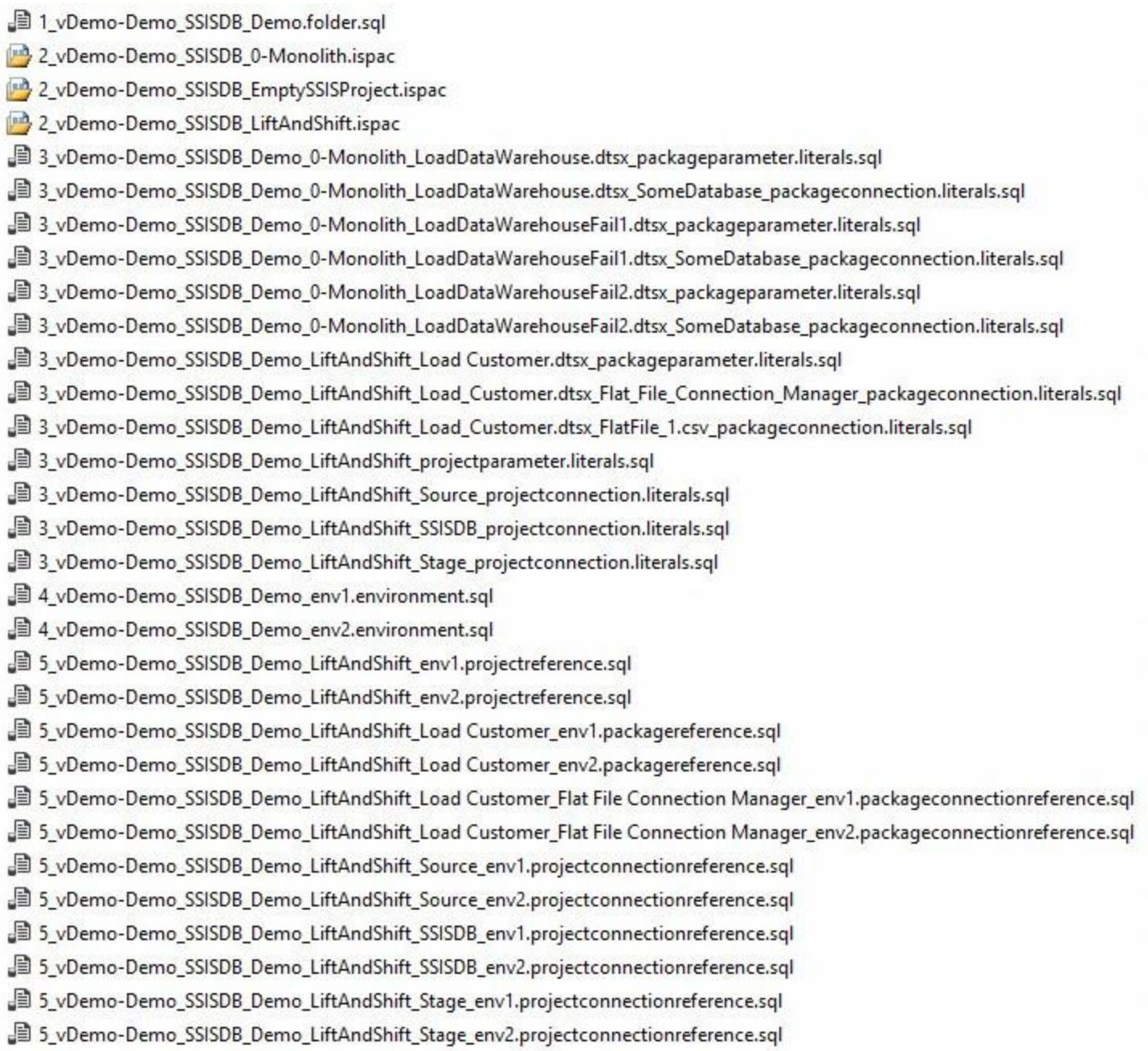


Figure 57

4.1.1 File Naming Convention

The Transact-SQL files are named using the following naming convention:

<*Precedence*>_<*SQL Server Source Instance Friendly Name*>_<*Path*>.<*Artifact Type*>.sql

The ISPAC files are named using the following naming convention:

2_<*SQL Server Source Instance Friendly Name*>_<*SSISDB*>_<*SSIS Project Name*>.ispac

The Transact-SQL and ISPAC files generated by SSIS Catalog Compare are *idempotent* – meaning they can be safely re-executed and produce the same result. More on this later…

4.1.1.1 Precedence

Scripts and ISPAC files are prefixed with a number that indicates precedence. For example, The Demo Catalog Folder must exist prior to the deployment of the LiftAndShift.ispac SSIS Project deployment to that Catalog Folder. So the Catalog Folder script is numbered "1" in its file name: 1_vDemo-Demo_SSISDB_Demo.folder.sql. The LiftAndShift ISPAC SSIS Project deployment file is numbered "2" in its file name: 2_vDemo-Demo_SSISDB_LiftAndShift.ispac. If executed in this order, the Demo Catalog Folder will be created first and will be ready for the deployment of the LiftAndShift SSIS project when the ISPAC file is executed.

4.2 GENERATE FOLDER SCRIPT

To generate a Catalog Folder script, right-click the folder and click "Generate Folder Script" as shown in Figure 58:

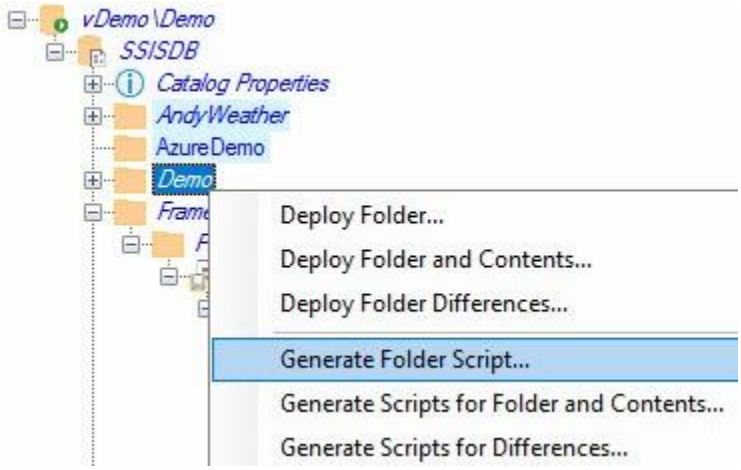


Figure 58

The "Browse For Folder" dialog displays. Select (or create) the file system folder where you wish to store the Catalog Folder script as shown in Figure 59:

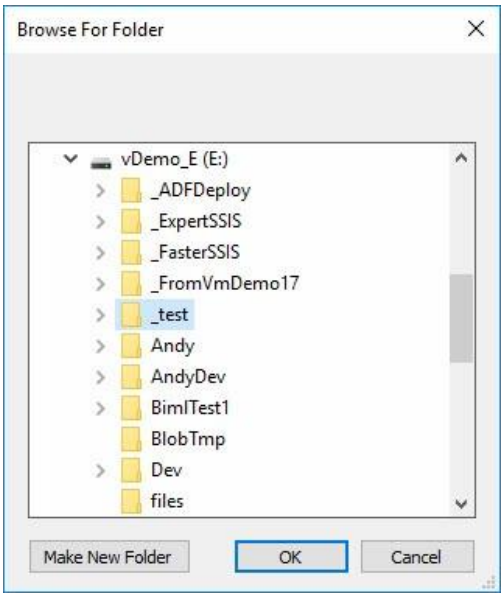


Figure 59

Inside the file system directory you selected, another file system directory is created with a file-system-friendly of the SQL Server instance which hosts the SSIS Catalog. Inside this folder a sub-directory is created with the same name as the SSIS Catalog Folder. Inside this folder the folder script is generated as shown in Figure 60:

Figure 60

The script contents appear similar to that shown in Figure 61:

```
/*
Script Name: E:\_test\20180823\vDemo_Demo\Demo\1_vDemo-Demo_SSISDB_Demo.folder.sql

Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/23/2018 10:29:05 AM
Generated From: CatalogBase v3.2.2.1
 executing on: VDEMO

*/
Use SSISDB
go

print 'Script Name: E:\_test\20180823\vDemo_Demo\Demo\1_vDemo-Demo_SSISDB_Demo.folder.sql

Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/23/2018 10:29:05 AM
Generated From: CatalogBase v3.2.2.1
 executing on: VDEMO'

print ''
print '------------------------------------------------------------'
print 'Deployed to Instance: ' + @@servername
print 'Deploy Date: ' + Convert(varchar,GetDate(), 101) + ' ' + Convert(varchar,GetDate(), 108)
print 'Deployed By: ' + original_login()
print '------------------------------------------------------------'
print ''
-- SSISDB\Demo


print 'Folder SSISDB\Demo'
If Not Exists(Select *
From SSISDB.[catalog].folders
Where name = N'Demo')
begin
print ' - Creating Demo folder'
declare @folder_id bigint
Exec SSISDB.[catalog].create_folder
   @folder_name = N'Demo'
, @folder_id = @folder_id OUTPUT
print ' - Demo folder created'
 end
else
 begin
  print ' - Demo folder already exists.'
 end
 print ' - Setting Demo folder description to '
 Exec SSISDB.[catalog].set_folder_description
       @folder_name = N'Demo'
     , @folder_description=N''
 print ' - Demo folder description set to '
go
```

Figure 61

When executed, the folder script either creates the SSIS Catalog Folder or informs the person executing the script that the SSIS Catalog Folder already exists. If the SSIS Catalog Folder is created, script execution generates output similar to that shown in Figure 62:

```
Script Name: E:\_test\20180823\vDemo_Demo\Demo\1_vDemo-Demo_SSISDB_Demo.folder.sql

Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/23/2018 10:29:05 AM
Generated From: CatalogBase v3.2.2.1
 executing on: VDEMO

------------------------------------------------------------
Deployed to Instance: VDEMO\QA
Deploy Date: 08/23/2018 10:41:50
Deployed By: VDEMO\A. Ray Leonard
------------------------------------------------------------

Folder SSISDB\Demo
 - Creating Demo folder
 - Demo folder created
 - Setting Demo folder description to
 - Demo folder description set to
```

Figure 62

> *Note: The statements returned in the Messages tab of SQL Server Management Studio (SSMS) are designed to be copied and stored. The authors recommend enterprises use a ticketing system to manage and track the deployment of enterprise scripts. Before closing a ticket to create a Catalog Folder, the deploying agent is advised to copy the contents of the Messages tab and paste them into the Notes section of the ticket for auditing purposes.*

After executing the folder script in the target instance, click the Refresh button in SSIS Catalog Compare to observe the updated SSIS Catalog state of the target SSIS Catalog instance as shown in Figure 63:

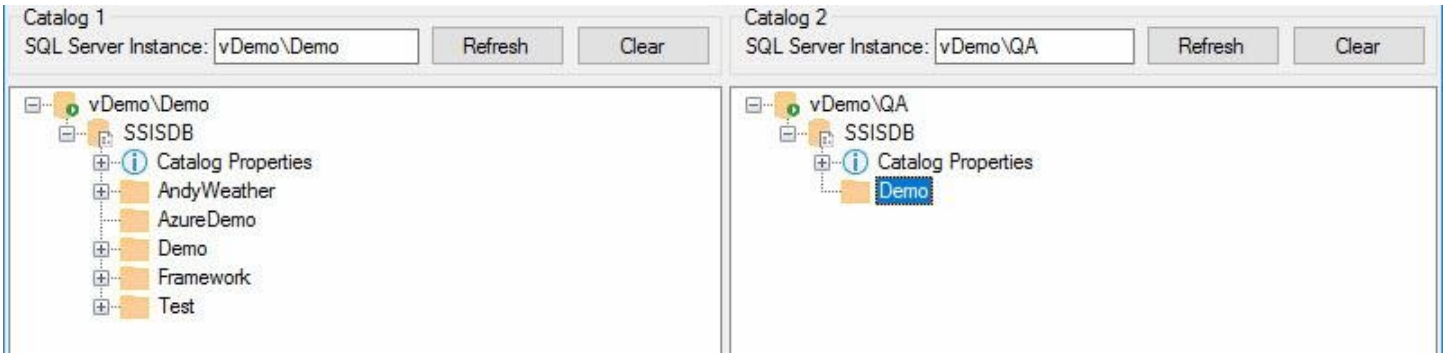


Figure 63

4.3 Generate Project ISPAC File

To generate a Project ISPAC file, right-click the project and click “Export ISPAC” as shown in Figure 64:

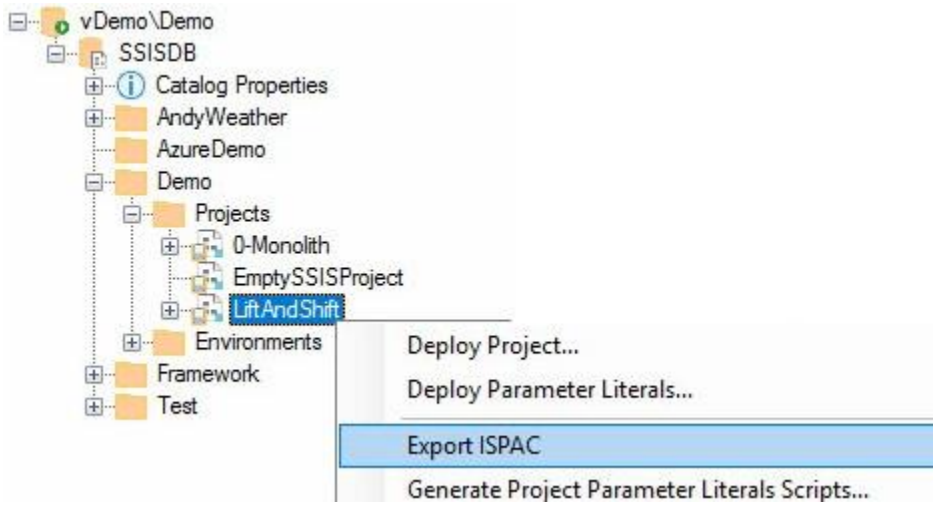


Figure 64

The "Browse For Folder" dialog displays. Select (or create) the file system folder where you wish to store the Project ISPAC file as shown in Figure 65:

Figure 65

Inside the file system folder you selected, another file system folder is created with the same name as the SSIS Catalog Folder. Inside this folder the project ISPAC file is generated as shown in Figure 66:

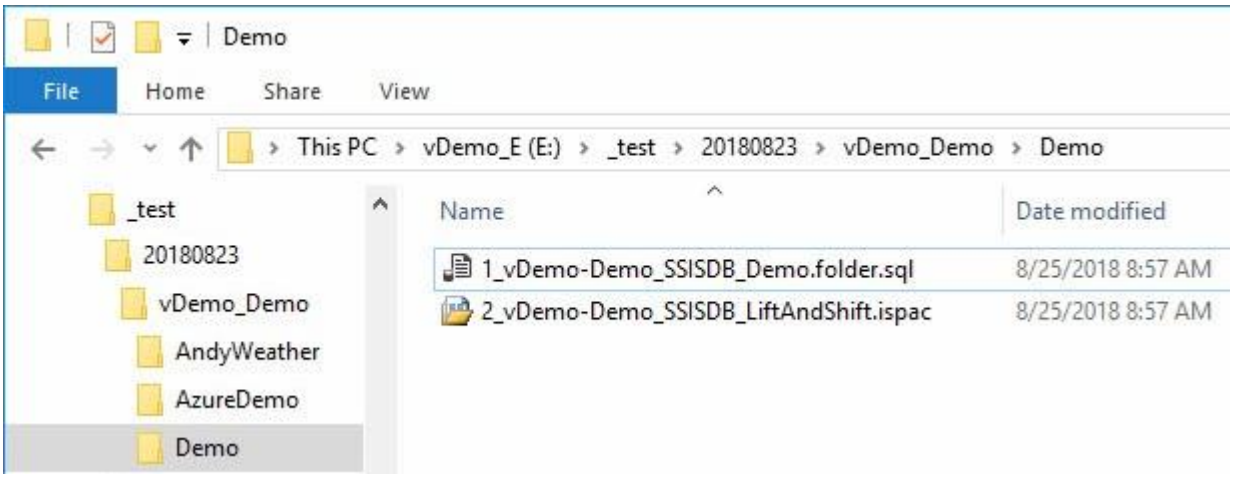


Figure 66

Note the Folder script is regenerated when the ISPAC file is generated. Scripts and ISPAC files for all dependencies are generated (or regenerated) by default when dependent scripts or ISPAC files are generated.

When executed, the ISPAC file starts the Integration Services Deployment Wizard and will appear similar to that shown in Figure 67:

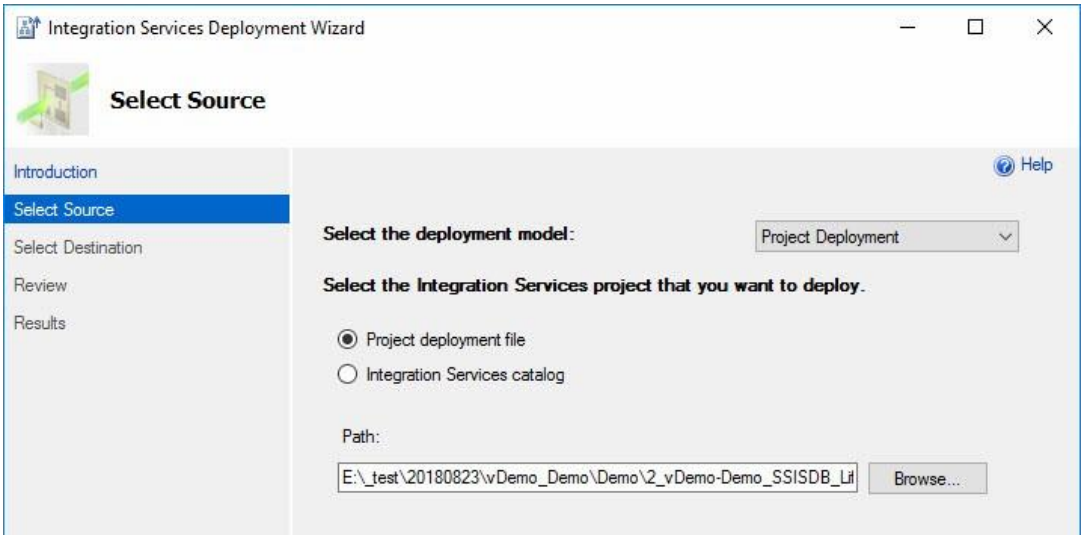


Figure 67

After executing the ISPAC file in the target instance, click the Refresh button in SSIS Catalog Compare to observe the updated SSIS Catalog state of the target SSIS Catalog instance as shown in Figure 68:

Figure 68

4.4 SURFACING CONNECTIONS

Changing the way Connections are surfaced and managed was a major focus of SSIS Catalog Compare version 3. The changes were first shared as part of SSIS Catalog Browser – a free utility from DILM Suite for viewing the contents of an SSIS Catalog – because building the surfacing mechanisms for the SSIS Catalog was the easiest part of the job. For this reason, SSIS Catalog Browser will almost always be ahead of SSIS Catalog Compare in surfacing functionality.

4.4.1 Connections Virtual Folders

SSIS Catalog Compare now surfaces Connections virtual folders at the Project and Package level as shown in Figure 69:

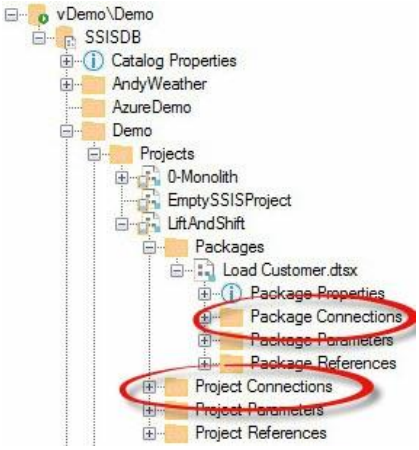


Figure 69

Expanded, the Connection virtual folders respectively contain nodes that represent SSIS Project and Package connections as shown in Figure 70:

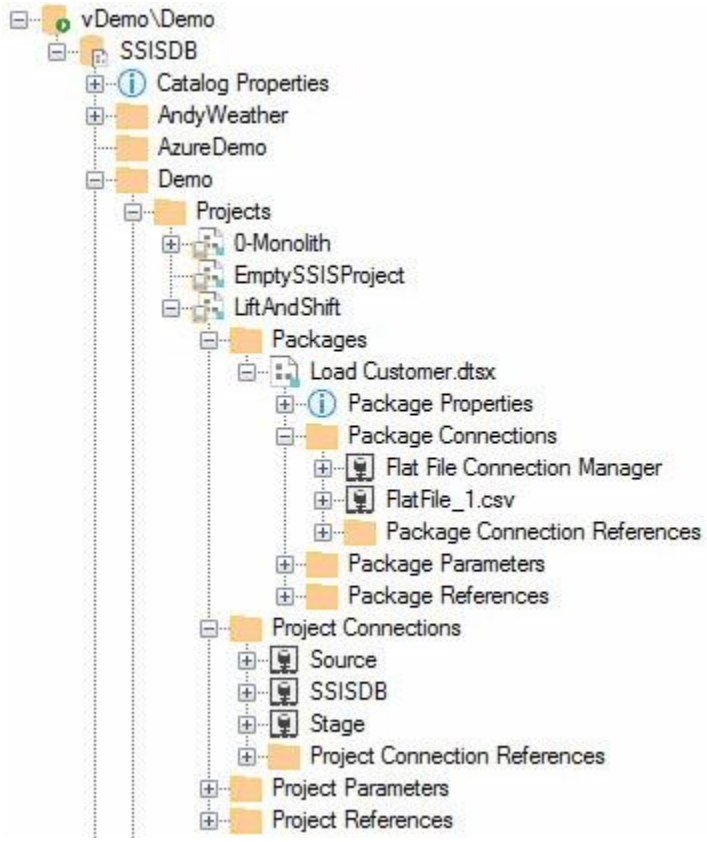


Figure 70

Each connection node, in turn, surfaces Connection properties as shown in Figure 71:

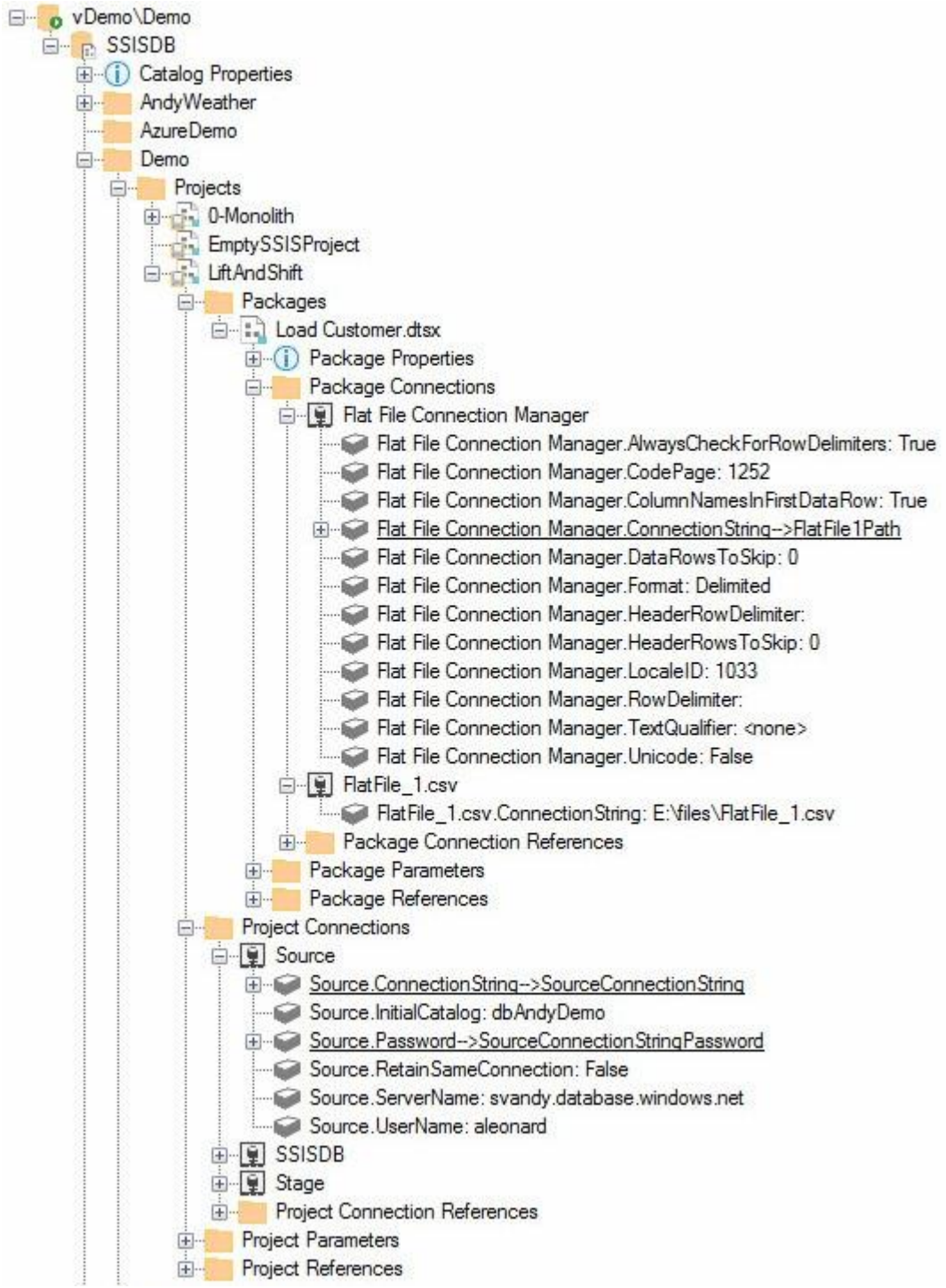


Figure 71

Different connection manager providers surface different property collections. *But* the SSIS Catalog treats connection manager properties *exactly* like parameters. Don't believe me? Query the SSISDB.catalog.object_parameters view as shown in Figure 72:

```
Select *
From [catalog].object_parameters
```

100 %

Results | Messages

	parameter_id	project_id	object_type	object_name	parameter_name	data_type	required	sensitive
1	1	1	20	LiftAndShift	ProjectParameter	String	0	0
2	2	1	20	LiftAndShift	ProjectDefaultParameter	Int32	0	0
3	3	1	20	LiftAndShift	ProjectLiteralParameter	String	0	0
4	4	1	20	LiftAndShift	CM.Stage.ConnectionString	String	0	0
5	5	1	20	LiftAndShift	CM.Stage.InitialCatalog	String	0	0
6	6	1	20	LiftAndShift	CM.Stage.Password	String	0	1
7	7	1	20	LiftAndShift	CM.Stage.RetainSameConnection	Boolean	0	0
8	8	1	20	LiftAndShift	CM.Stage.ServerName	String	0	0
9	9	1	20	LiftAndShift	CM.Stage.UserName	String	0	0
10	10	1	20	LiftAndShift	CM.Source.ConnectionString	String	0	0
11	11	1	20	LiftAndShift	CM.Source.InitialCatalog	String	0	0
12	12	1	20	LiftAndShift	CM.Source.Password	String	0	1
13	13	1	20	LiftAndShift	CM.Source.RetainSameConnection	Boolean	0	0
14	14	1	20	LiftAndShift	CM.Source.ServerName	String	0	0
15	15	1	20	LiftAndShift	CM.Source.UserName	String	0	0
16	16	1	20	LiftAndShift	CM.SSISDB.ConnectionString	String	0	0
17	17	1	20	LiftAndShift	CM.SSISDB.InitialCatalog	String	0	0
18	18	1	20	LiftAndShift	CM.SSISDB.Password	String	0	1
19	19	1	20	LiftAndShift	CM.SSISDB.RetainSameConnection	Boolean	0	0
20	20	1	20	LiftAndShift	CM.SSISDB.ServerName	String	0	0
21	21	1	20	LiftAndShift	CM.SSISDB.UserName	String	0	0
22	22	1	30	Load Customer.dtsx	AdditionalParameter	String	0	0
23	23	1	30	Load Customer.dtsx	PackageDefaultParameter	Int32	0	0
24	24	1	30	Load Customer.dtsx	PackageLiteralParameter	String	0	0
25	25	1	30	Load Customer.dtsx	PackageParameter	String	0	0
26	26	1	30	Load Customer.dtsx	CM.Flat File Connection Manager.AlwaysCheckForRowDelimiters	Boolean	0	0
27	27	1	30	Load Customer.dtsx	CM.Flat File Connection Manager.CodePage	Int32	0	0
28	28	1	30	Load Customer.dtsx	CM.Flat File Connection Manager.ColumnNamesInFirstDataRow	Boolean	0	0
29	29	1	30	Load Customer.dtsx	CM.Flat File Connection Manager.ConnectionString	String	0	0
30	30	1	30	Load Customer.dtsx	CM.Flat File Connection Manager.DataRowsToSkip	Int32	0	0
31	31	1	30	Load Customer.dtsx	CM.Flat File Connection Manager.Format	String	0	0

Figure 72

As shown in Figure 72, Project parameters shown at the top of the results with object_type 20 include Project-level parameters *and* project connection manager properties. The same can be seen for Package parameters and connection manager properties with object_type 30, found lower in the results. Connection manager properties are stored in the SSIS Catalog in the same location as parameters (the internal.object_parameters table).

Connection manager properties are identified with the prefix "CM.".

SSIS Catalog Compare surfaces connection manager properties apart from parameters to surface a more-accurate visualization of the SSIS Catalog.

4.4.2 Generate Project Connection Literals Script

There are a couple ways to generate project connection literals scripts:

- Right-click the project and click "Generate Project Connection Parameter Literals Script" as shown in Figure 73:

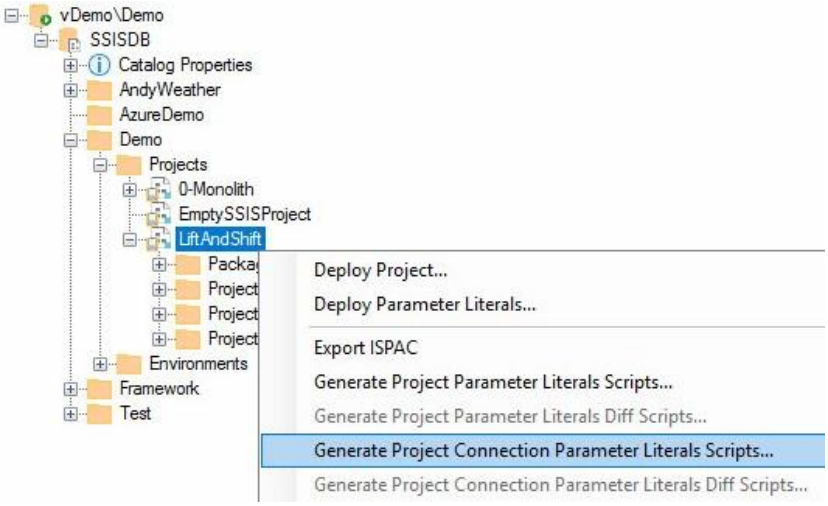


Figure 73

- Right-click a project connection and then click "Generate Connection Literals Script" as shown in Figure 74:

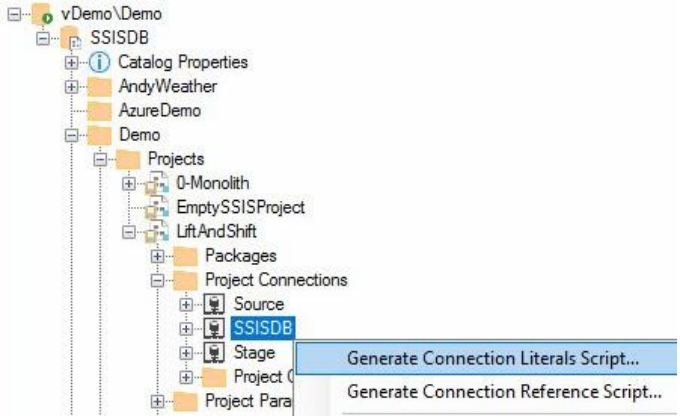


Figure 74

The next step for both methods is the Browse for Folder dialog show in Figure 75:

Figure 75

The major difference between these methods is the first method generates scripts for *all* Project connections – plus dependencies (the project's ISPAC file and the Catalog Folder script) – while the second method generates *only* the script for the selected Project connection.

4.4.2.1 Anatomy of a Connection Literals Script

A Connection Literals script is automatically generated from several methods in SSIS Catalog Compare's Catalog object named CatalogBase.

4.4.2.1.1 Declarations and Header Documentation

The script begins with a declaration of Transact-SQL parameters that will be used to provide literal overrides. These parameters are placed at the top of the script for easy access by release management personnel, DBAs, DevOps, or other specialists responsible for deployment and deployment testing – as shown in Figure 76:

```
-- PROJECT CONNECTION PARAMETER LITERAL VALUES --

-- LiftAndShift\CM_SSISDB_ConnectionString_0 Project Connection Parameter Lit
Declare @ProjectParameter_CM_SSISDB_ConnectionString_0 sql_variant = N'Data S

----------------------------

/*
Script Name: E:\_test\20180827\vDemo_Demo\Demo\3_vDemo-Demo_SSISDB_Demo_LiftA
Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Project Name: LiftAndShift
Project Connection Name: SSISDB
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/27/2018 6:54:43 AM
Generated From: CatalogBase v3.2.2.3
 executing on: VDEMO
*/
```

Figure 76

The next section provides feedback for the professional deploying the script. The feedback includes the same information contained in the head documentation, followed by deployment feedback as seen in Figure 77:

```
print 'Script Name: E:\_test\20180827\vDemo_Demo\Demo\3_vDemo-Demo_SSISDB_Demo_LiftAndShift_SSISDB_projectconnection.literals.sql
Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Project Name: LiftAndShift
Project Connection Name: SSISDB
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/27/2018 6:54:43 AM
Generated From: CatalogBase v3.2.2.3
 executing on: VDEMO
'
print ''
print '------------------------------------------------------------'
print 'Deployed to Instance: ' + @@servername
print 'Deploy Date: ' + Convert(varchar,GetDate(), 101) + ' ' + Convert(varchar,GetDate(), 108)
print 'Deployed By: ' + original_login()
print '------------------------------------------------------------'
print ''
```

Figure 77

When executed, this documentation section returns messages similar to that shown in Figure 78:

```
Script Name: E:\_test\20180827\vDemo_Demo\Demo\3_vDemo-Demo_SSISDB_Demo_LiftAndShift_SSISDB_projectconnection.literals.sql
Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Project Name: LiftAndShift
Project Connection Name: SSISDB
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/27/2018 6:54:43 AM
Generated From: CatalogBase v3.2.2.3
 executing on: VDEMO

------------------------------------------------------------
Deployed to Instance: VDEMO\QA
Deploy Date: 08/27/2018 10:46:41
Deployed By: VDEMO\A. Ray Leonard
------------------------------------------------------------
```

Figure 78

As mentioned otherwise in this document, these messages are intended to be copied and stored in the Notes field of a ticketing system in a DevOps enterprise. Note the detail contained herein:

- Script Name – the path to the file used to perform the operation.
- Generated From – the SQL Server instance of the SSIS Catalog host from which the script was generated.
- Catalog Name – redundant at present because all SSIS Catalogs are named "SSISDB."
- Folder Name – the name of the SSIS Catalog Folder that contains the scripted artifact.
- Project Name – the name of the SSIS Project that contains the scripted artifact.
- Project Connection Name – the name of the SSIS Project Connection.
- Generated By – the name of the enterprise account used to generate the artifact's script.
    - Note: SSIS Catalog Compare respects the security model of the SSIS Catalog. Windows Authentication is required to perform many SSIS Catalog operations.
- Generated Date – the date and time the script was generated.
- Generated From – the version of CatalogBase used in the generation of the artifact script.
    - Executing On – the name of the machine on which CatalogBase was running.
- Deployed to Instance – the SQL Server instance hosting the target SSIS Catalog.
- Deploy Date – the date and time the deployment script was executed.
- Deploy By – the enterprise account used to deploy the artifact script.

4.4.2.1.2 Script Support Declarations

The next section of the artifact script is the declaration of parameters used to support the remained of the script's operations. An example is shown in Figure 79:

```
40  declare @object_type smallint = 20 -- project
41  declare @folder_name nvarchar(128) = N'Demo'
42  declare @project_name nvarchar(128) = N'LiftAndShift'
43  declare @object_name nvarchar(260) = N'LiftAndShift'
44  declare @value_type char(1) = 'V'
45  declare @validation_status char(1) = 'N'
46  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0 nvarchar(128) = N'CM.SSISDB.ConnectionString'
47  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0_parameter_data_type nvarchar(128) = N'String'
48  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0_required bit = 0
49  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0_sensitive bit = 0
50  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0_description nvarchar(1024) = N''
51  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0_base_data_type nvarchar(128) = N'nvarchar'
52  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0_default_value sql_variant = N'Data Source=vDemo\Demo;Initial Catalog=SSI
53  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0_design_default_value sql_variant = N'Data Source=vDemo\Test;Initial Cata
54  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0_sensitive_default_value varbinary(max) = NULL
55  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0_value_set bit = 1
56  declare @ProjectParameterName_CM_SSISDB_ConnectionString_0_referenced_variable_name nvarchar(128) = N''
```

Figure 79

4.4.2.1.3 Status and Conditions Checks

The next section of the artifact script checks for the existence of required artifacts like Catalog Folders and Projects. An example of checks for the existence of a Catalog Folder and an SSIS Project is shown in Figure 80:

```
declare @ErrMsg varchar(1000)
declare @folderID bigint = (Select folder_id
                            From SSISDB.[catalog].folders
                            Where name = @folder_name)
print 'Check for folder: ' + @folder_name
If (@folderID Is NULL)
  begin
    set @ErrMsg = ' - Folder ' + @folder_name + ' does not exist.'
    raisError(@ErrMsg, 16, 1)
    return
  end
Else
  begin
    print ' - Folder ' + @folder_name + ' exists.'
  end
print ''

print 'Check for project: ' + @folder_name + '\' + @project_name + ''
declare @projectID bigint = (Select project_id
                             From SSISDB.[catalog].projects
                             Where name = N'' + @project_name + ''
                               And folder_id = @folderID)
declare @objectVersionLSN bigint = (Select object_version_lsn
                                    From SSISDB.[catalog].projects
                                    Where project_id = @projectID)
If (@projectID Is NULL)
  begin
    set @ErrMsg = ' - Project ' + @folder_name + '\' + @project_name + ' does not exist.'
    raisError(@ErrMsg, 16, 1)
    return
  end
Else
  begin
    print ' - Project ' + @folder_name + '\' + @project_name + ' exists.'
  end
print ''
```

Figure 80

An example of messages generated by this portion of the script are shown in Figure 81:

```
Check for folder: Demo
 - Folder Demo exists.

Check for project: Demo\LiftAndShift
 - Project Demo\LiftAndShift exists.
```

Figure 81

If required preceding artifacts do *not* exist in the target SSIS Catalog, an error message is generated – similar to that seen in Figure 82:

```
Check for folder: Demo
 - Folder Demo exists.

Check for project: Demo\LiftAndShift
Msg 50000, Level 16, State 1, Line 87
 - Project Demo\LiftAndShift does not exist.
```

Figure 82

4.4.2.1.4 Connections Properties Reset

Unlike scripts we've examined to date, Connections Literals scripts *reset* all related properties (parameters) for a connection manager that are not overridden via Reference Mapping. The portion of the script that manages clearing connection property parameter values is shown in Figure 83:

```
-- Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal
print 'Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal'
-- Check for Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal
print ' - Check for Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal'
If Exists(Select *
          From SSISDB.[internal].[object_parameters] op
          Where op.[object_type] = 20
            And op.[object_name] = N'LiftAndShift'
            And op.parameter_name = N'CM.SSISDB.ConnectionString' )
 begin
  -- Clearing Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal
  print ' - Clearing Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal'
  Exec [SSISDB].[catalog].[clear_object_parameter_value]
      @object_type = 20 -- project
    , @parameter_name = N'CM.SSISDB.ConnectionString'
    , @object_name = N'LiftAndShift'
    , @project_name = N'LiftAndShift'
    , @folder_name = N'Demo'
  print ' - Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal cleared'
 end
Else
 begin
  print ' - Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal does not exist.'
 end
print ''
```

Figure 83

The results of the execution of this portion of the artifact script are shown in Figure 84:

```
Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal
 - Check for Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal
 - Clearing Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal
 - Demo\LiftAndShift\SSISDB\CM.SSISDB.ConnectionString Project Connection Parameter Literal cleared
```

Figure 84

4.4.2.1.5 Connections Properties Literal Override

The final section of the Connection Property Literal script contains the literal override. This section of the script is laden with existence checks and conditionals, as shown in Figure 85:

```
-- Demo\LiftAndShift\CM.SSISDB.ConnectionString Project Connection Parameter Literal
print ' - Add or Update ' + @folder_name + '\' + @project_name + '\CM.SSISDB.Connecti
If Exists(Select *
          From SSISDB.[catalog].[object_parameters] op
          Join SSISDB.[catalog].projects p
            On p.project_id = op.project_id
          Join SSISDB.[catalog].folders f
            On f.folder_id = p.folder_id
          Where f.[name] = @folder_name
            And p.[name] = @project_name
            And op.[object_name] = @object_name
            And op.[parameter_name] = N'CM.SSISDB.ConnectionString')
 begin
  If ((@ProjectParameter_CM_SSISDB_ConnectionString_0 Is Not NULL) And(@ProjectParame
   begin
    print ' - Updating ' + @folder_name + '\' + @project_name + '\CM.SSISDB.Connectio
    Exec [SSISDB].[catalog].[set_object_parameter_value]
        @object_type = @object_type
      , @parameter_name = @ProjectParameterName_CM_SSISDB_ConnectionString_0
      , @object_name = @object_name
      , @folder_name = @folder_name
      , @project_name = @project_name
      , @parameter_value = @ProjectParameter_CM_SSISDB_ConnectionString_0
      , @value_type = @value_type
    print ' - ' + @folder_name + '\' + @project_name + '\CM.SSISDB.ConnectionString P
    if (@ProjectParameterName_CM_SSISDB_ConnectionString_0_sensitive = 0)
     begin
      print ' - ' + @folder_name + '\' + @project_name + '\CM.SSISDB.ConnectionString
     end
    else
     begin
      print ' - ' + @folder_name + '\' + @project_name + '\CM.SSISDB.ConnectionString
     end
   end
  Else
   begin
    Set @ErrMsg = ' - NOT UPDATED! ' + @folder_name + '\' + @project_name + '\CM.SSIS
    RaisError(@ErrMsg, 16, 1)
   end
 end
Else
 begin
    print ' - ' + @folder_name + '\' + @project_name + '\CM.SSISDB.ConnectionString P
 end
print ''
```

Figure 85

The results of executing this section of the script are shown in Figure 86:

```
- Add or Update Demo\LiftAndShift\CM.SSISDB.ConnectionString Project Connection Parameter Literal
- Updating Demo\LiftAndShift\CM.SSISDB.ConnectionString Project Connection Parameter Literal
- Demo\LiftAndShift\CM.SSISDB.ConnectionString Project Connection Parameter Literal updated
- Demo\LiftAndShift\CM.SSISDB.ConnectionString Project Connection Parameter Value set to: Data Sour
```

Figure 86

As you may glean from this analysis of one script generated for Project Connection Literals management, the Transact-SQL for scripting SSIS Catalog artifacts is rigorous, containing several existence and error-condition checks prior to performing any updates. The script is designed to be *idempotent*, as well, meaning the script will succeed and the results will be repeatable and predictable each time the script is executed – and that the script itself is re-executable.

4.4.3 Generate Project Parameter Literals Script

To generate a Project Literals script, right-click the project and click "Generate Parameter Literals Scripts" as shown in Figure 87:

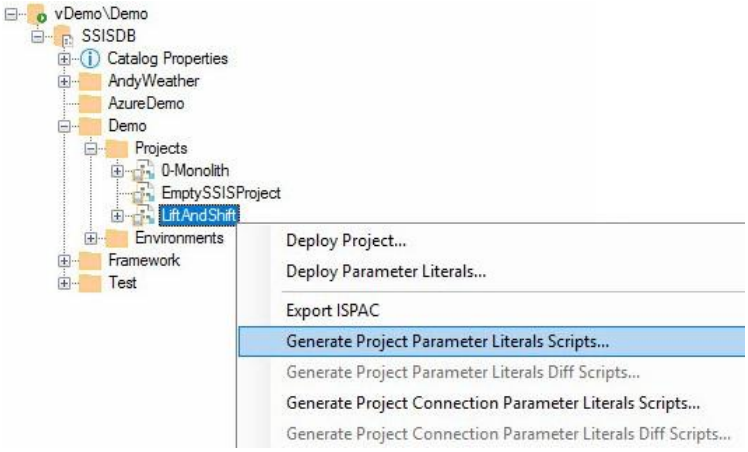


Figure 87

The "Browse For Folder" dialog displays. Select (or create) the file system folder where you wish to store the Catalog Folder script as shown in Figure 88:

Figure 88

Inside the file system folder you selected, another file system folder is created with the same name as the SSIS Catalog Folder. Inside this folder the Project Parameter Literals script is generated as shown in Figure 89:

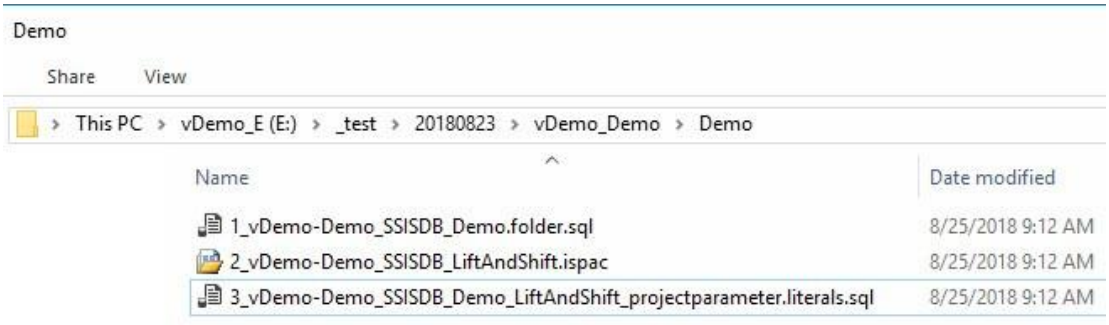


Figure 89

As before, the dependent Catalog artifacts – the Demo Catalog and the LiftAndShift SSIS Project in this case – are regenerated.

The script contents appear similar to those shown in Figure 90:

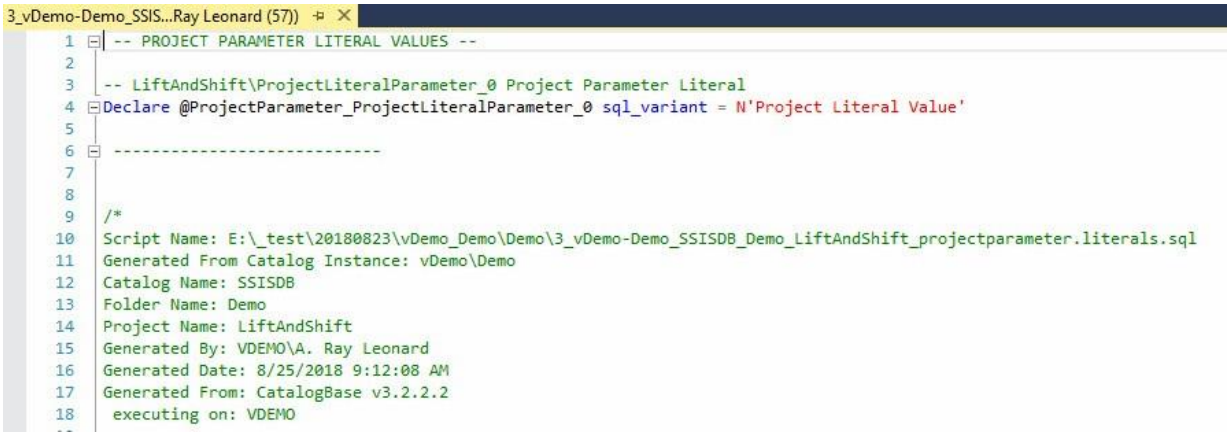


Figure 90

When executed, the folder script either configures the Project Parameter Literals in the Catalog as shown in Figure 91:

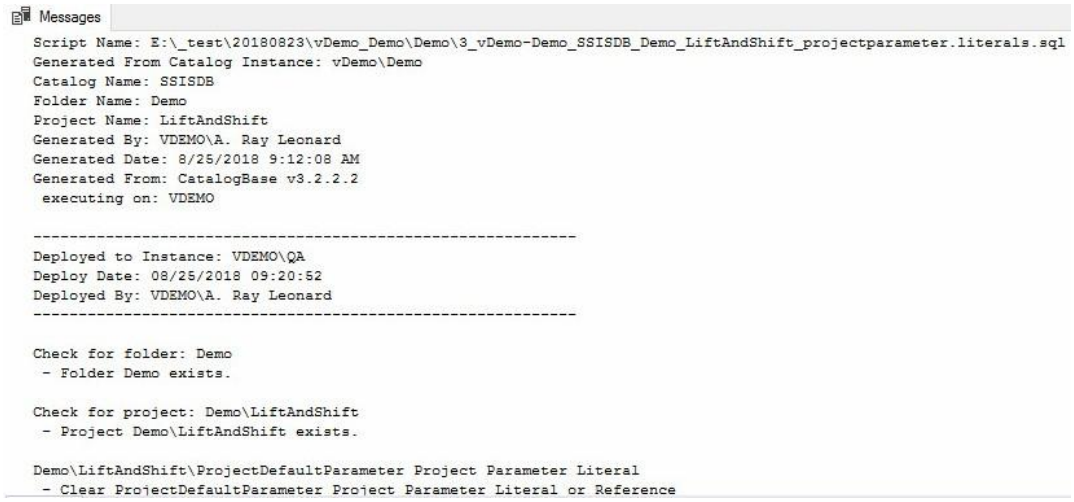


Figure 91

> *Note: The statements returned in the Messages tab of SQL Server Management Studio (SSMS) are designed to be copied and stored. The authors recommend enterprises use a ticketing system to manage and track the deployment of enterprise scripts. Before closing a ticket to create Project Parameter Literals, the deploying agent is advised to copy the contents of the Messages tab and paste them into the Notes section of the ticket for auditing purposes.*

For a detailed overview of the anatomy of artifact scripts, please see the section titled Anatomy of a Connections Literal Script.

After executing the project literals script in the target instance, click the Refresh button in SSIS Catalog Compare to observe the updated SSIS Catalog state of the target SSIS Catalog instance as shown in Figure 92:

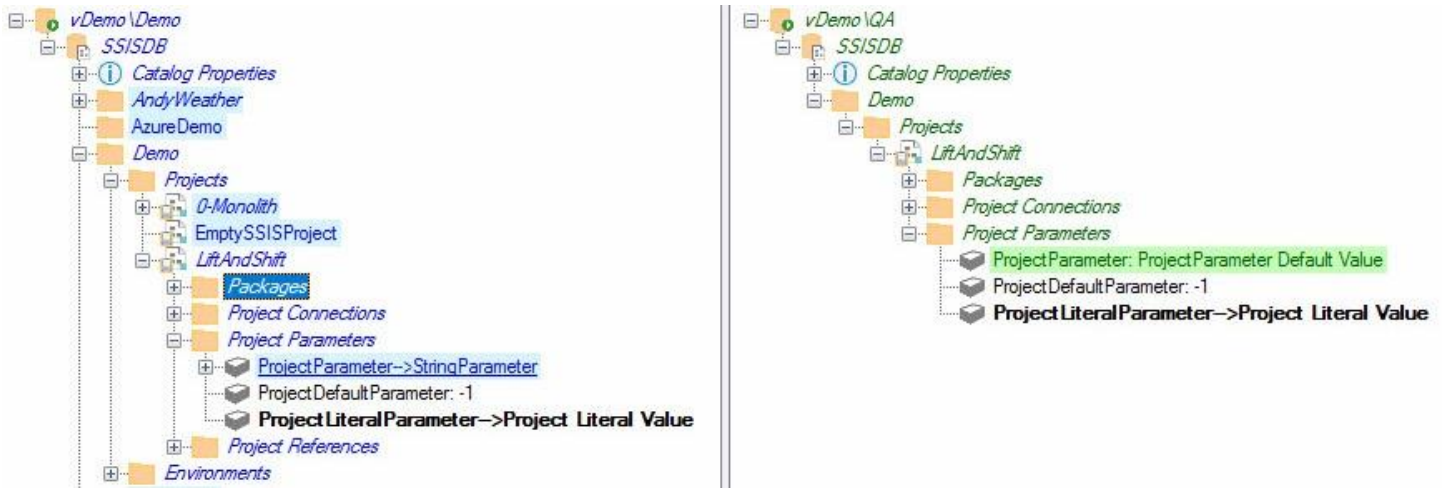


Figure 92

4.4.4 Generate Package Connections and Package Literals Scripts

Scripts for Package Connections Literals and Package Literals are generated in much the same way as their Project counterparts. The biggest difference is scope.

As with Project Connections Literals scripts, there are two ways to generate a Package Connections Literal Scripts:

- Right-click the package and click "Generate Project Connections Parameter Literals Script" as shown in Figure 93:

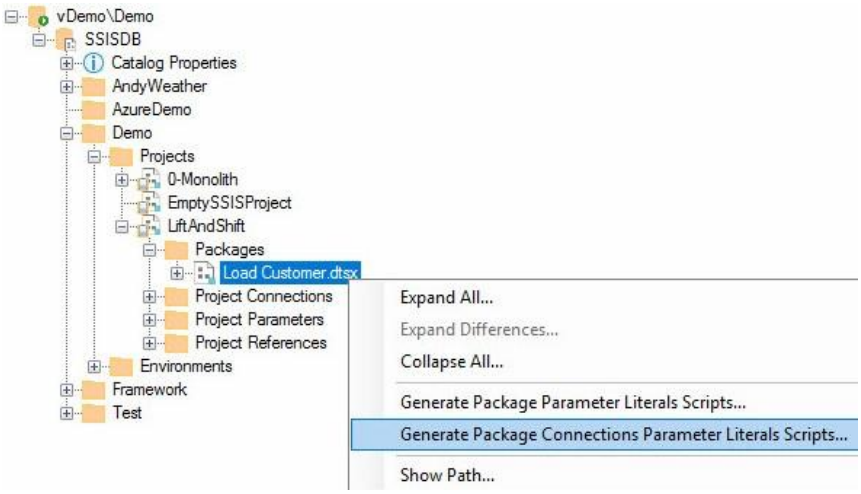


Figure 93

- Right-click a package connection and then click "Generate Connection Literals Script" as shown in Figure 94:

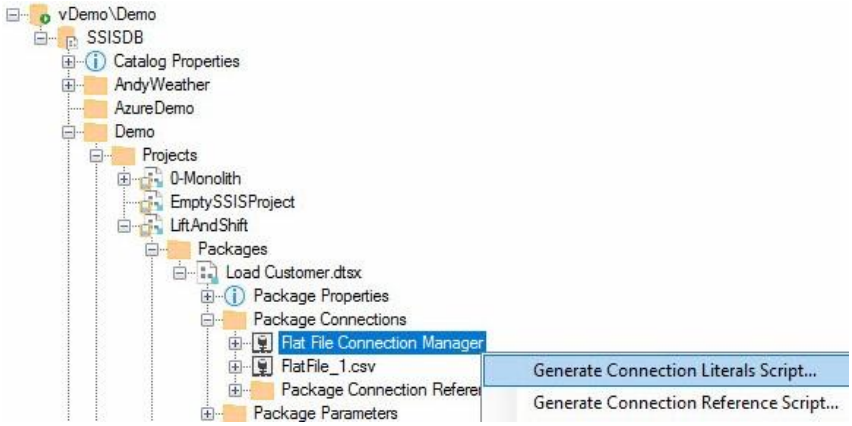


Figure 94

The next step for both methods is the Browse for Folder dialog show in Figure 95:

Figure 95

The major difference between these methods is the first method generates scripts for *all* Package connections – plus dependencies (the project's ISPAC file and the Catalog Folder script) – while the second method generates *only* the script for the selected Package connection.

For details please see the section titled Anatomy of a Connection Literals Script.

4.5 Generate Environment Script

To generate a Catalog Environment script, right-click the environment and click "Generate Environment Script" as shown in Figure 96:

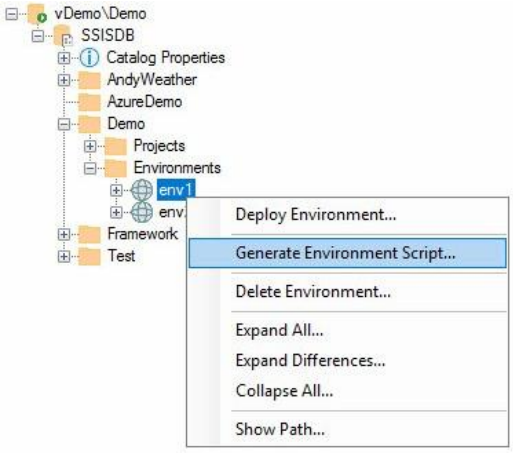


Figure 96

The "Browse For Folder" dialog displays. Select (or create) the file system folder where you wish to store the Catalog Environment script as shown in Figure 97:

Figure 97

Inside the file system folder you selected, another file system folder is created with the same name as the SSIS Catalog Folder. Inside this folder the Catalog Environment script is generated as shown in Figure 98:

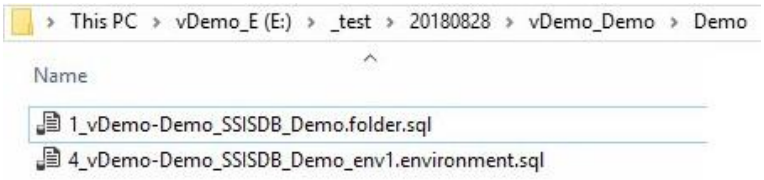
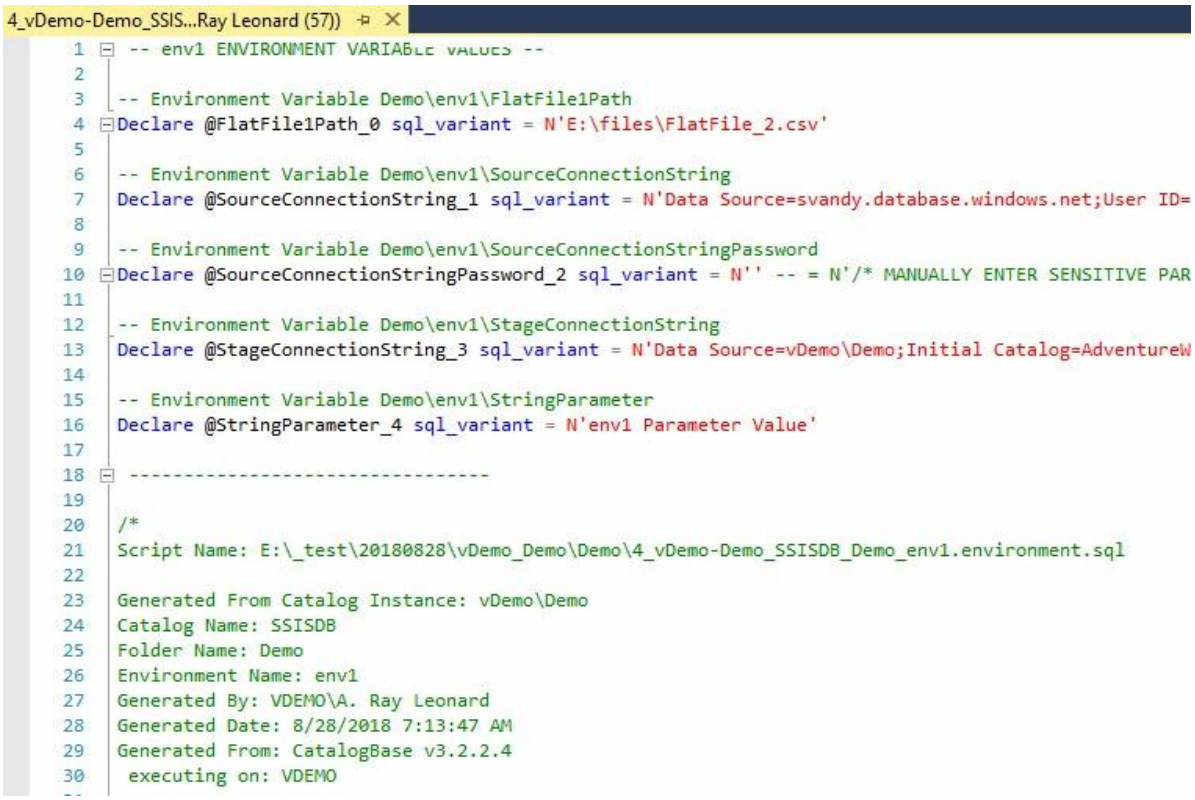


Figure 98

The script contents appear similar to that shown in Figure 99:



```
4_vDemo-Demo_SSIS...Ray Leonard (57))
 1  -- env1 ENVIRONMENT VARIABLE VALUES --
 2
 3  -- Environment Variable Demo\env1\FlatFile1Path
 4  Declare @FlatFile1Path_0 sql_variant = N'E:\files\FlatFile_2.csv'
 5
 6  -- Environment Variable Demo\env1\SourceConnectionString
 7  Declare @SourceConnectionString_1 sql_variant = N'Data Source=svandy.database.windows.net;User ID=
 8
 9  -- Environment Variable Demo\env1\SourceConnectionStringPassword
10  Declare @SourceConnectionStringPassword_2 sql_variant = N'' -- = N'/* MANUALLY ENTER SENSITIVE PAR
11
12  -- Environment Variable Demo\env1\StageConnectionString
13  Declare @StageConnectionString_3 sql_variant = N'Data Source=vDemo\Demo;Initial Catalog=AdventureW
14
15  -- Environment Variable Demo\env1\StringParameter
16  Declare @StringParameter_4 sql_variant = N'env1 Parameter Value'
17
18  ---------------------------------
19
20  /*
21  Script Name: E:\_test\20180828\vDemo_Demo\Demo\4_vDemo-Demo_SSISDB_Demo_env1.environment.sql
22
23  Generated From Catalog Instance: vDemo\Demo
24  Catalog Name: SSISDB
25  Folder Name: Demo
26  Environment Name: env1
27  Generated By: VDEMO\A. Ray Leonard
28  Generated Date: 8/28/2018 7:13:47 AM
29  Generated From: CatalogBase v3.2.2.4
30   executing on: VDEMO
```

Figure 99

When executed, the Catalog Environment script either creates the Catalog Environment or informs the person executing the script that the Catalog Environment already exists. If Environment Variables exist, they are dropped and recreated. If the Catalog Environment is created, script execution generates output similar to that shown in Figure 100:

```
Messages
Script Name: E:\_test\20180828\vDemo_Demo\Demo\4_vDemo-Demo_SSISDB_Demo_env1.environment.sql

Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Environment Name: env1
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/28/2018 7:13:47 AM
Generated From: CatalogBase v3.2.2.4
 executing on: VDEMO

------------------------------------------------------------
Deployed to Instance: VDEMO\QA
Deploy Date: 08/28/2018 07:18:02
Deployed By: VDEMO\A. Ray Leonard
------------------------------------------------------------

Check for folder: Demo
 - Demo folder exists.

Environment SSISDB\Demo\env1
 - Creating Environment SSISDB\Demo\env1
 - Environment SSISDB\Demo\env1 created
```

Figure 100

Note: The statements returned in the Messages tab of SQL Server Management Studio (SSMS) are designed to be copied and stored. The authors recommend enterprises use a ticketing system to manage and track the deployment of enterprise scripts. Before closing a ticket to create a Catalog Environment, the deploying agent is advised to copy the contents of the Messages tab and paste them into the Notes section of the ticket for auditing purposes.

4.5.1 Anatomy of an Environment Script

A Catalog Environment script is automatically generated from several methods in SSIS Catalog Compare's Catalog object named CatalogBase.

4.5.1.1.1 Declarations and Header Documentation

The script begins with a declaration of Transact-SQL parameters that support Catalog Environment Variables contained within the Catalog Environment. These parameters are placed at the top of the script for easy access by release management personnel, DBAs, DevOps, or other specialists responsible for deployment and deployment testing – as shown in Figure 101:

```
-- env1 ENVIRONMENT VARIABLE VALUES --

-- Environment Variable Demo\env1\FlatFile1Path
Declare @FlatFile1Path_0 sql_variant = N'E:\files\FlatFile_2.csv'

-- Environment Variable Demo\env1\SourceConnectionString
Declare @SourceConnectionString_1 sql_variant = N'Data Source=svandy.database.windows.net;User

-- Environment Variable Demo\env1\SourceConnectionStringPassword
Declare @SourceConnectionStringPassword_2 sql_variant = N'' -- = N'/* MANUALLY ENTER SENSITIVE

-- Environment Variable Demo\env1\StageConnectionString
Declare @StageConnectionString_3 sql_variant = N'Data Source=vDemo\Demo;Initial Catalog=Adventu

-- Environment Variable Demo\env1\StringParameter
Declare @StringParameter_4 sql_variant = N'env1 Parameter Value'
```

Figure 101

Script documentation follows and is recorded as both Transact-SQL documentation and then printed so it will be part of the output found in the Messages window, shown in Figure 102:

```
/*
Script Name: E:\_test\20180828\vDemo_Demo\Demo\4_vDemo-Demo_SSISDB_Demo_env1.environment.sql

Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Environment Name: env1
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/28/2018 7:13:47 AM
Generated From: CatalogBase v3.2.2.4
 executing on: VDEMO

*/

print 'Script Name: E:\_test\20180828\vDemo_Demo\Demo\4_vDemo-Demo_SSISDB_Demo_env1.environment.sql

Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Environment Name: env1
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/28/2018 7:13:47 AM
Generated From: CatalogBase v3.2.2.4
 executing on: VDEMO
'
```

Figure 102

When executed, this portion of the script outputs messages suitable for copying and pasting into the Notes field of a ticket used by enterprise DevOps teams, as shown in Figure 103:

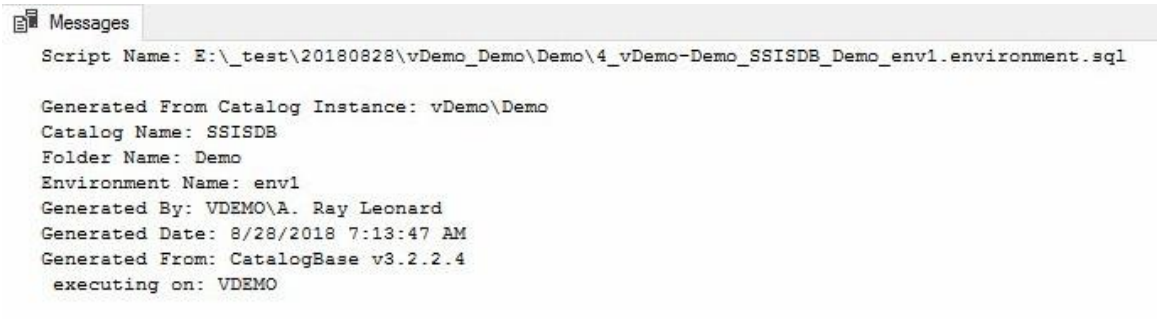


Figure 103

The last piece of the script header is the deployment output message, for which the script is shown in Figure 104:

```
45  print ''
46  print '------------------------------------------------------------'
47  print 'Deployed to Instance: ' + @@servername
48  print 'Deploy Date: ' + Convert(varchar,GetDate(), 101) + ' ' + Convert(varchar,GetDate(), 108)
49  print 'Deployed By: ' + original_login()
50  print '------------------------------------------------------------'
51  print ''
```

Figure 104

When executed, this portion of the script produces output similar to that shown in Figure 105:

```
------------------------------------------------------------
Deployed to Instance: VDEMO\QA
Deploy Date: 08/28/2018 07:18:02
Deployed By: VDEMO\A. Ray Leonard
------------------------------------------------------------
```

Figure 105

4.5.1.1.2 Status and Conditions Checks

The next section of the artifact script checks for the existence of the Catalog Folder which is the only prerequisite for a Catalog Environment. An example is shown in Figure 106:

```
52  declare @ErrMsg varchar(100)
53  print 'Check for folder: Demo '
54  If Not Exists(Select name
55                From SSISDB.[catalog].folders
56                Where name = N'Demo')
57   begin
58    set @ErrMsg = ' - Demo does not exist.'
59    raisError(@ErrMsg, 16, 1)
60    return
61   end
62  Else
63   begin
64    print ' - Demo folder exists.'
65   end
66  print ''
```

Figure 106

When executed, this portion of the script produces a message similar to that shown in Figure 107:

```
Check for folder: Demo
 - Demo folder exists.
```

Figure 107

4.5.1.1.3 Catalog Environment Check / Creation

The next portion of the script checks for the existence of the Catalog Environment and creates it if it does not exist. If the environment exists, the script informs the individual executing the script as shown in Figure 108:

```
68   -- Environment SSISDB\Demo\env1
69  print 'Environment SSISDB\Demo\env1'
70  If Not Exists(Select *
71              From SSISDB.[catalog].environments e
72              Join SSISDB.[catalog].folders f
73                On f.folder_id = e.folder_id
74              Where e.name = N'env1'
75                And f.name = N'Demo')
76
77  begin
78    print ' - Creating Environment SSISDB\Demo\env1'
79
80    Exec SSISDB.[catalog].create_environment
81        @environment_name=N'env1'
82      , @folder_name=N'Demo'
83    print ' - Environment SSISDB\Demo\env1 created'
84  end
85  else
86   print ' - Environment SSISDB\Demo\env1 already exists.'
87  print ''
```

Figure 108

If the script creates the Environment, the output appears similar to that shown in Figure 109:

```
Environment SSISDB\Demo\env1
 - Creating Environment SSISDB\Demo\env1
 - Environment SSISDB\Demo\env1 created
```

Figure 109

If the Environment exists, the user is informed via output message:

```
Environment SSISDB\Demo\env1
 - Environment SSISDB\Demo\env1 already exists.
```

Figure 110

4.5.1.1.4 Environment Variables

The final portion of the script checks for the existence of the Environment Variables and responds accordingly. This is a three-step process:

1. Drop the Environment Variable if it exists.
2. Create the Environment Variable.
3. Set the Environment Variable value.

If the Environment Variable exists the script drops it. Why? SSIS Catalog Compare wants to be sure the environment variable is created with the proper data type and initial values.

The next step is creation of the Environment Variable.

Finally, the Environment Variable value is set. This is somewhat redundant as the value of the Environment Variable is initialized when the Environment Variable is created in the previous step.

An example of the Transact-SQL for this portion of the script is shown in Figure 111:

```
-- Environment Variable Demo\env1\StringParameter
print 'Environment Variable Demo\env1\StringParameter'
If Exists(Select *
          From SSISDB.[catalog].environment_variables ev
       Join SSISDB.[catalog].environments e
On e.environment_id = ev.environment_id
   Join SSISDB.[catalog].folders f
On f.folder_id = e.folder_id
   Where e.name = N'env1'
And ev.name = N'StringParameter'
And f.name = N'Demo')
 begin
  print ' - Dropping Environment Variable Demo\env1\StringParameter'
  Exec SSISDB.[catalog].delete_environment_variable
    @variable_name=N'StringParameter'
   , @environment_name=N'env1'
   , @folder_name=N'Demo'
  print ' - Environment Variable Demo\env1\StringParameter dropped'
 end
print ' - Creating Environment Variable Demo\env1\StringParameter'
Exec SSISDB.[catalog].create_environment_variable
      @variable_name = N'StringParameter'
    , @sensitive = 0
    , @environment_name = N'env1'
    , @folder_name = N'Demo'
    , @value = @StringParameter_4
    , @data_type = N'String'

print ' - Environment Variable Demo\env1\StringParameter created'

-- Set Environment Variable Demo\env1\StringParameter
print ' - Set Environment Variable Demo\env1\StringParameter value'
Exec SSISDB.[catalog].set_environment_variable_value
      @variable_name = N'StringParameter'
    , @environment_name = N'env1'
    , @folder_name = N'Demo'
    , @value = @StringParameter_4
print ' - Environment Variable Demo\env1\StringParameter value set'
print ''
```

Figure 111

After executing this portion of the script, messages similar to those shown in Figure 112 are displayed in the Messages output:

```
Environment Variable Demo\env1\StringParameter
 - Creating Environment Variable Demo\env1\StringParameter
 - Environment Variable Demo\env1\StringParameter created
 - Set Environment Variable Demo\env1\StringParameter value
 - Environment Variable Demo\env1\StringParameter value set
```

Figure 112

If the SSIS Catalog Environment Variable was first dropped, the output messages appear as shown in Figure 113:

```
Environment Variable Demo\env1\StringParameter
 - Dropping Environment Variable Demo\env1\StringParameter
 - Environment Variable Demo\env1\StringParameter dropped
 - Creating Environment Variable Demo\env1\StringParameter
 - Environment Variable Demo\env1\StringParameter created
 - Set Environment Variable Demo\env1\StringParameter value
 - Environment Variable Demo\env1\StringParameter value set
```

Figure 113

4.5.2 Catalog Environments, Post-Script-Execution

After executing the Catalog Environment script in the target instance, click the Refresh button in SSIS Catalog Compare to observe the updated SSIS Catalog state of the target SSIS Catalog instance as shown in Figure 114:

Figure 114

4.6 Generate Project and Package Reference Script

To generate a Project Reference script, right-click the project reference and click “Generate Reference Script” as shown in Figure 115:

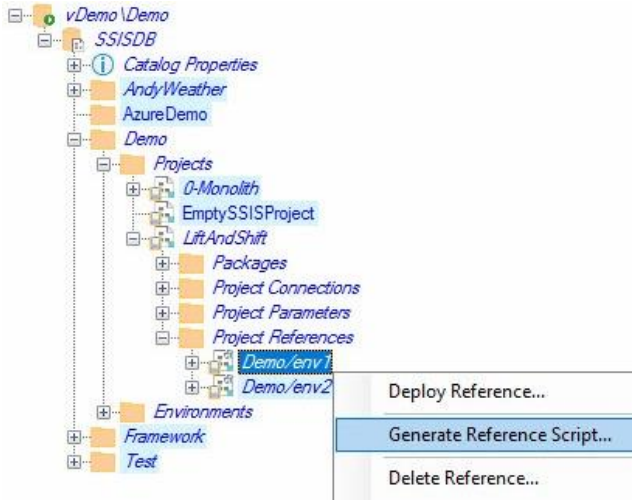


Figure 115

The “Browse For Folder” dialog displays. Select (or create) the file system folder where you wish to store the Catalog Environment script as shown in Figure 116:

Figure 116

Inside the file system folder you selected, another file system folder is created with the same name as the SSIS Catalog Folder. Inside this folder the Project Reference script is generated as shown in Figure 117:

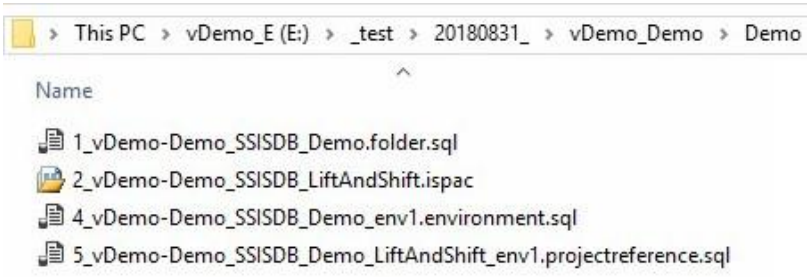


Figure 117

Dependent artifacts – the Catalog Folder, the SSIS Project ISPAC file, and the Catalog Environment that is referenced – are scripted, as well.

4.6.1 Anatomy of a Reference Script

A Catalog Reference script is automatically generated from several methods in SSIS Catalog Compare's Catalog object named CatalogBase.

4.6.1.1.1 Header Documentation

The script begins with a header documentation similar to that shown in Figure 118:

```
/*
Script Name: E:\_test\20180831_\vDemo_Demo\Demo\5_vDemo-Demo_SSISDB_Demo_LiftAndShift_env1.projectreference.sql

Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Project Name: LiftAndShift
Reference Name: Demo/env1
Environment Name: env1
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/31/2018 10:56:15 AM
Generated From: CatalogBase v3.2.2.5
 executing on: VDEMO

*/
 -- SSISDB\Demo\LiftAndShift\[.|env1]
print 'Script Name: E:\_test\20180831_\vDemo_Demo\Demo\5_vDemo-Demo_SSISDB_Demo_LiftAndShift_env1.projectreference.sql

Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Project Name: LiftAndShift
Reference Name: Demo/env1
Environment Name: env1
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/31/2018 10:56:15 AM
Generated From: CatalogBase v3.2.2.5
 executing on: VDEMO
'
print ''
print '------------------------------------------------------------'
print 'Deployed to Instance: ' + @@servername
print 'Deploy Date: ' + Convert(varchar,GetDate(), 101) + ' ' + Convert(varchar,GetDate(), 108)
print 'Deployed By: ' + original_login()
print '------------------------------------------------------------'
print ''
```

Figure 118

When executed, the Project Reference script header documentation portion appears similar to that shown in Figure 119:

```
Script Name: E:\_test\20180831_\vDemo_Demo\Demo\5_vDemo-Demo_SSISDB_Demo_LiftAndShift_env1.projectreference.sql

Generated From Catalog Instance: vDemo\Demo
Catalog Name: SSISDB
Folder Name: Demo
Project Name: LiftAndShift
Reference Name: Demo/env1
Environment Name: env1
Generated By: VDEMO\A. Ray Leonard
Generated Date: 8/31/2018 10:56:15 AM
Generated From: CatalogBase v3.2.2.5
 executing on: VDEMO

------------------------------------------------------------
Deployed to Instance: VDEMO\QA
Deploy Date: 08/31/2018 11:04:08
Deployed By: VDEMO\A. Ray Leonard
------------------------------------------------------------
```

Figure 119

> *Note: The statements returned in the Messages tab of SQL Server Management Studio (SSMS) are designed to be copied and stored. The authors recommend enterprises use a ticketing system to manage and track the deployment of enterprise scripts. Before closing a ticket to create a Catalog Reference, the deploying agent is advised to copy the contents of the Messages tab and paste them into the Notes section of the ticket for auditing purposes.*

4.6.1.1.2 Status and Conditions Checks

The next portion of the Catalog Reference script tests for the existence of prerequisite artifacts, as shown in Figure 120:

```
declare @ErrMsg varchar(100)
print 'Check for folder: Demo '
If Not Exists(Select name
              From SSISDB.[catalog].folders
              Where name = N'Demo')
 begin
  set @ErrMsg = ' - Demo does not exist.'
  raisError(@ErrMsg, 16, 1)
  return
 end
Else
 begin
  print ' - Demo folder exists.'
 end
print ''

print 'Check for project: LiftAndShift '
If Not Exists(Select name
              From SSISDB.[catalog].projects
              Where name = N'LiftAndShift')
 begin
  set @ErrMsg = ' - LiftAndShift project does not exist.'
  raisError(@ErrMsg, 16, 1)
  return
 end
Else
 begin
  print ' - LiftAndShift project exists.'
 end
print ''


print 'Check for environment: env1 '
If Not Exists(Select name
              From SSISDB.[catalog].environments
              Where name = N'env1')
 begin
  set @ErrMsg = ' - env1 environment does not exist.'
  raisError(@ErrMsg, 16, 1)
  return
 end
Else
 begin
  print ' - env1 environment exists.'
 end
print ''
```

Figure 120

Prerequisites for a Project Reference include:

- Catalog Folder
- SSIS Project
- Catalog Environment

When executed, this portion of the script returns status messages for prerequisites similar to those shown in Figure 121:

Figure 121

4.6.1.1.3 Create the Reference

The next portion of the Reference Script creates the Reference which is a relationship between an SSIS Catalog Environment and an SSIS Project (or Package). An example of this portion of the script is shown in Figure 122:

```
print 'Reference SSISDB\Demo\LiftAndShift\[.|env1]'
If Not Exists(Select *
              From SSISDB.[catalog].environment_references er
              Join SSISDB.[catalog].projects cp
                On cp.project_id = er.project_id
              Join SSISDB.[catalog].folders cf
                On cf.folder_id = cp.folder_id
              Where cf.name = N'Demo'
                And cp.name = N'LiftAndShift'
                And er.environment_name = N'env1'
                And er.environment_folder_name Is NULL)
 begin
  print ' - Creating Reference SSISDB\Demo\LiftAndShift\[.|env1]'
  Declare @reference_id bigint
  Exec [SSISDB].[catalog].[create_environment_reference]
       @environment_name = N'env1'
     , @reference_id = @reference_id OUTPUT
     , @project_name = N'LiftAndShift'
     , @folder_name = N'Demo'
     , @environment_folder_name = NULL
     , @reference_type = 'R'
  print ' - Reference SSISDB\Demo\LiftAndShift\[.|env1] created'
 end
else
 print ' - Reference SSISDB\Demo\LiftAndShift\[.|env1] already exists.'
print ''
```

Figure 122

Once this portion of the script is executed, a message similar to that shown in Figure 123 is returned if the reference is created:

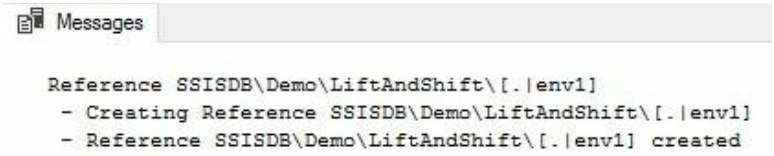


Figure 123

If the script detects the reference already exists, a message similar to that shown in Figure 124 is returned:

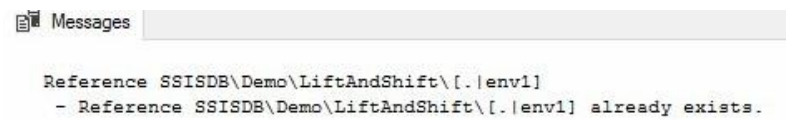


Figure 124

4.6.1.1.4 Clear the Parameter Value

The next portion of the Reference script clears the parameter value as shown in Figure 125:

```
-- Reference Project Parameter Mapping StringParameter-->ProjectParameter
print ' - Reference Project Parameter Mapping StringParameter-->ProjectParameter'
print ' - Clear Reference Project Parameter Mapping StringParameter-->ProjectParameter'
  Exec [SSISDB].[catalog].[clear_object_parameter_value]
      @object_type = 20 -- project
    , @object_name = N'LiftAndShift'
    , @parameter_name = N'ProjectParameter'
    , @folder_name = N'Demo'
    , @project_name = N'LiftAndShift'

print ' - Reference Project Parameter Mapping StringParameter-->ProjectParameter cleared'
print ''
```

Figure 125

The messages generated by this portion of the References script appear similar to that shown in Figure 126:

```
Messages

 - Reference Project Parameter Mapping StringParameter-->ProjectParameter
 - Clear Reference Project Parameter Mapping StringParameter-->ProjectParameter
 - Reference Project Parameter Mapping StringParameter-->ProjectParameter cleared
```

Figure 126

4.6.1.1.5 Set the Parameter Value

The final portion of the script builds the *Reference Mapping* – the relationship between a Catalog Environment Variable and a Parameter that the Environment Variable value will override at execution-time. This portion of the script is shown in Figure 127:

```
print ' - Add or Update Reference Project Parameter Mapping StringParameter-->ProjectParameter'
Exec [SSISDB].[catalog].[set_object_parameter_value]

     @object_type = 20 -- project
   , @parameter_name = N'ProjectParameter'
   , @object_name = N'LiftAndShift'
   , @folder_name = N'Demo'
   , @project_name = N'LiftAndShift'
   , @value_type = R
   , @parameter_value = N'StringParameter'

print ' - Reference Project Parameter Mapping StringParameter-->ProjectParameter added / updated'
print ''
```

Figure 127

When executed, this portion of the script generates a message similar to that shown in Figure 128:

```
Messages

 - Add or Update Reference Project Parameter Mapping StringParameter-->ProjectParameter
 - Reference Project Parameter Mapping StringParameter-->ProjectParameter added / updated
```

Figure 128

After executing the Project Reference script in the target instance, click the Refresh button in SSIS Catalog Compare to observe the updated SSIS Catalog state of the target SSIS Catalog instance as shown in Figure 129:

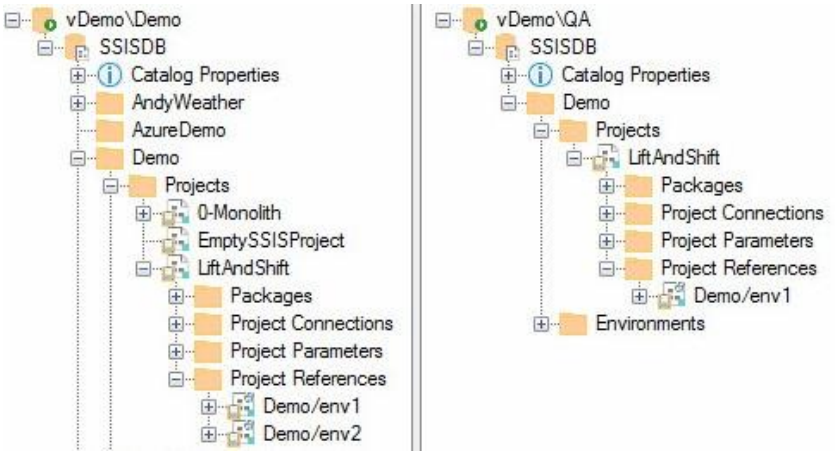

vDemo\Demo
SSISDB
Catalog Properties
AndyWeather
AzureDemo
Demo
Projects
0-Monolith
EmptySSISProject
LiftAndShift
Packages
Project Connections
Project Parameters
Project References
Demo/env1
Demo/env2
vDemo\QA
SSISDB
Catalog Properties
Demo
Projects
LiftAndShift
Packages
Project Connections
Project Parameters
Project References
Demo/env1
Environments

Figure 129

5 DEPLOY CATALOG ARTIFACTS

SSIS Catalog Compare provides deployment functionality for eleven SSIS Catalog artifacts:

1. Folders
2. Projects
3. Project Literals
4. Project Connection Literals
5. Package Literals
6. Package Connection Literals
7. Environments
8. Project References
9. Project Connection References
10. Package References
11. Package Connection References

5.1 DEPLOYING FOLDERS

There are three options for deploying Catalog Folders from one SSIS Catalog to another:

1. Deploy Folder
2. Deploy Folder and Contents
3. Deploy Folder Differences

5.1.1 Deploy Folder

To deploy a Catalog Folder, right-click the Folder and click "Deploy Folder" as shown in Figure 130:

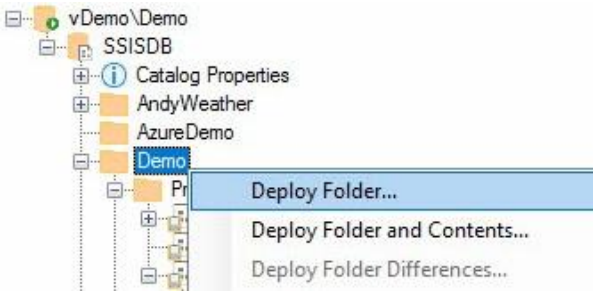


Figure 130

The Confirm Deploy Folder dialog displays as shown in Figure 131:

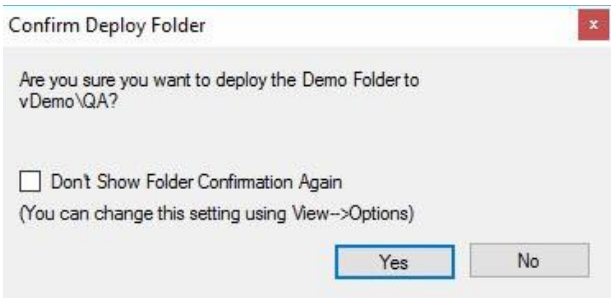


Figure 131

(…unless View→Options “Don’t Show Folder Deploy Confirmation Dialog” is selected as shown in Figure 132:

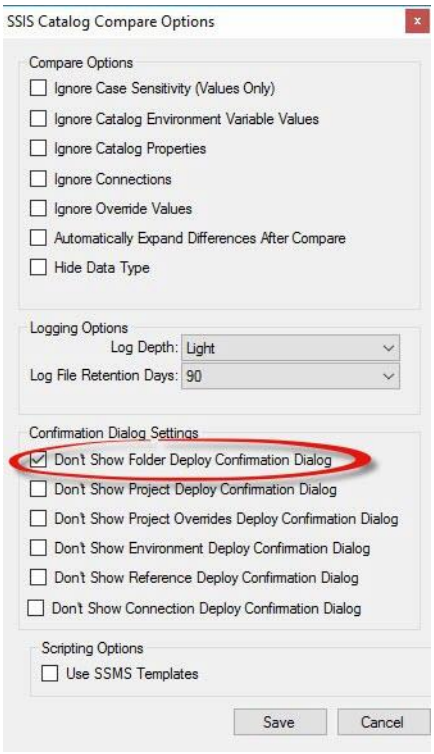


Figure 132

Once deployment is complete, the target treeview refreshes to display the updated target SSIS Catalog contents as shown in Figure 133:

Figure 133

5.1.2 Deploy Folder and Contents

If you examine Figure 133, you will note the Demo folder deployed to vDemo\QA is empty. That's because we deployed *only* the Demo folder in the previous step. What if we want to deploy the folder and its contents? There's an option for that as shown in Figure 134:

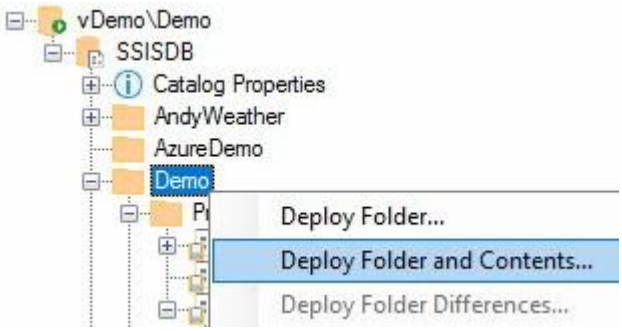


Figure 134

Catalog Compare prompts you, to be sure this is what you intend as shown:

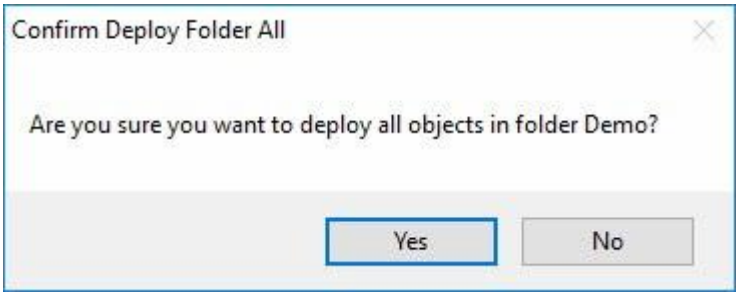


Figure 135

If you click the Yes button, another prompt seeks details regarding options. You can:

- Yes == Replace *all* contents of the folder in the target Catalog (if it exists)
- No == Overwrite the contents of the folder in the target Catalog (if they exist, Append if they do not exist)
- Cancel == cancels the current operation.

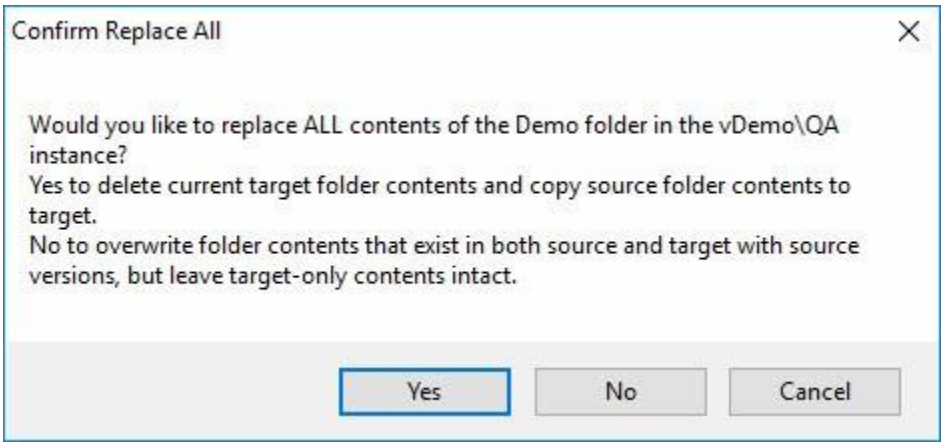


Figure 136

To make sure the folder and contents were deployed successfully, execute a Compare operation by clicking the Compare button. The results of the Compare operation appear in Figure 137:

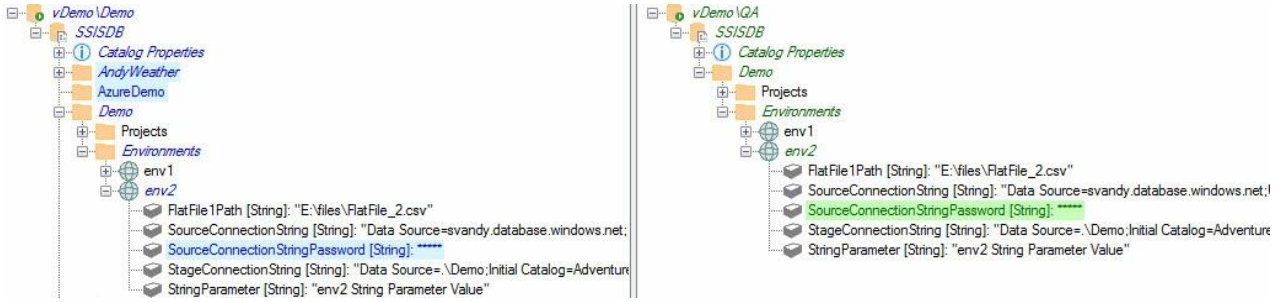


Figure 137

5.1.2.1 A Note About Sensitive Values

Why is there a difference detected between sensitive Catalog Environment Variables? SSIS Catalog Compare will never move sensitive values from one SSIS Catalog to another. SSIS Catalog Compare never "writes down" a value flagged as sensitive in an SSIS Catalog.

This is by design. Release management personnel are required to manually enter sensitive values.

5.1.3 Deploy Folder Differences

After a Compare operation, the option to Deploy Folder Differences is enabled as shown in Figure 138:

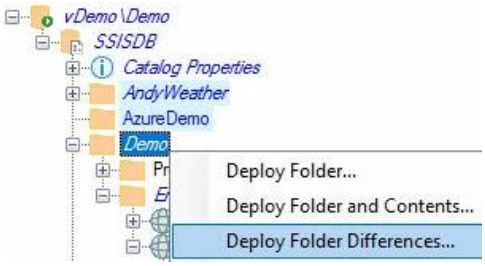


Figure 138

The user is prompted as shown in Figure 139:

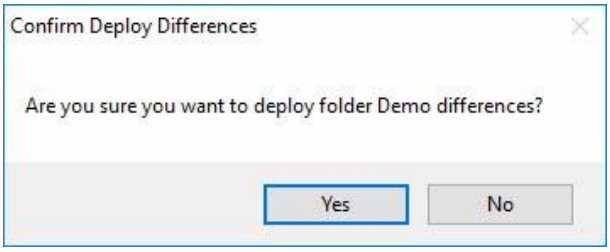


Figure 139

If the user clicks the Yes button, detected differences are deployed. Please note, sensitive values are never deployed (see A Note About Sensitive Values).

5.2 DEPLOY PROJECT

To deploy a Project, right-click the Project and click "Deploy Project" as shown in Figure 140:

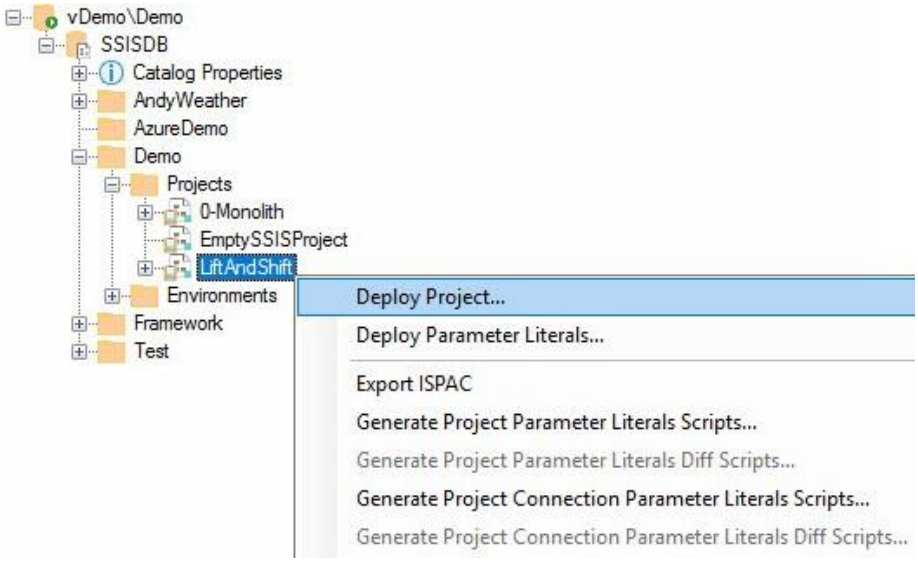


Figure 140

The "Browse For Folder" dialog displays as shown in Figure 141:

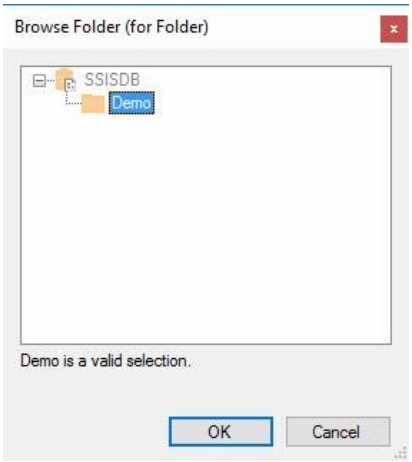


Figure 141

The Confirm Deploy Project dialog displays as shown in Figure 142:

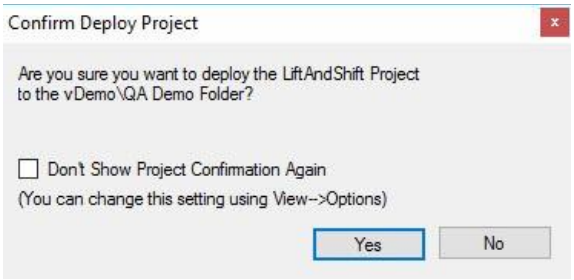


Figure 142

(…unless View→Options "Don't Show Project Deploy Confirmation Dialog" is selected as shown in Figure 143:

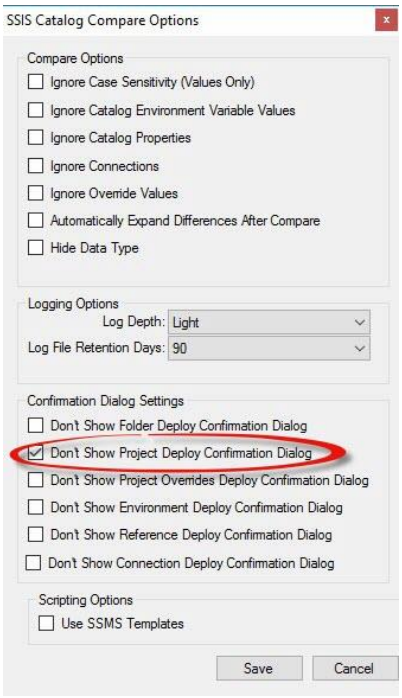


Figure 143

Once deployment is complete, the target treeview refreshes to display the updated target SSIS Catalog contents as shown in Figure 144:

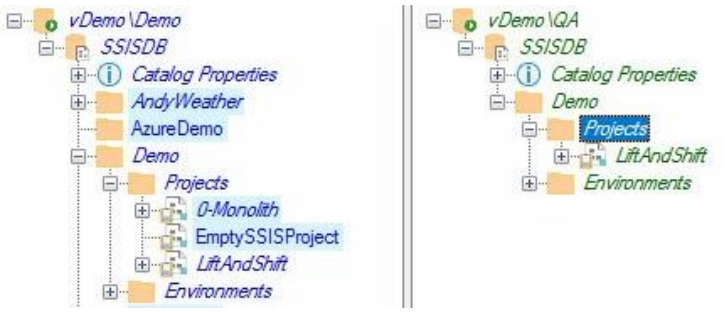


Figure 144

5.3 Deploy Project Parameter Literals

To deploy Project Literals, right-click the Project and click "Deploy Parameter Literals" as shown in Figure 145:

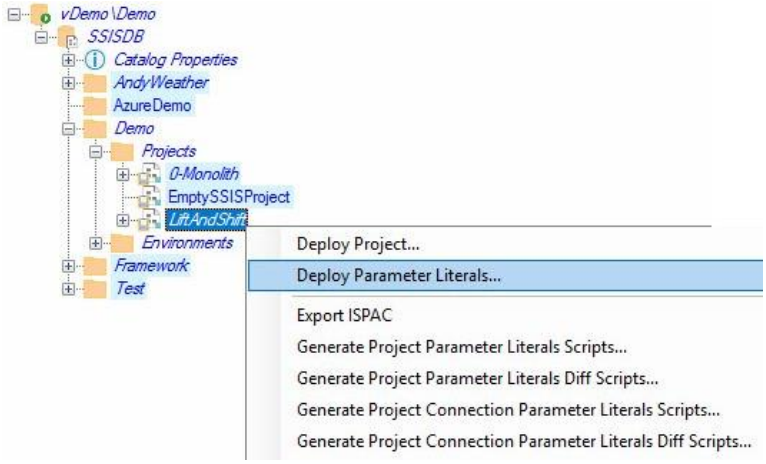


Figure 145

The "Browse For Folder" dialog displays as shown in Figure 146:

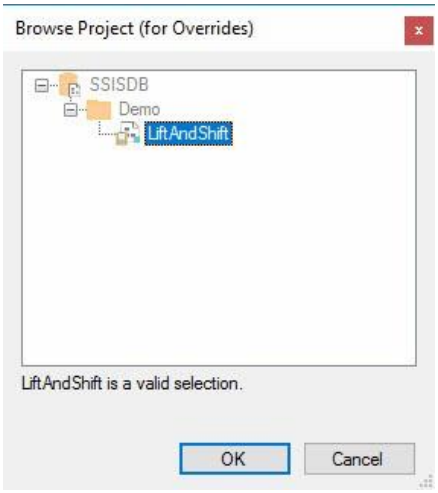


Figure 146

The Confirm Deploy Project Parameter Overrides dialog displays as shown in Figure 147:

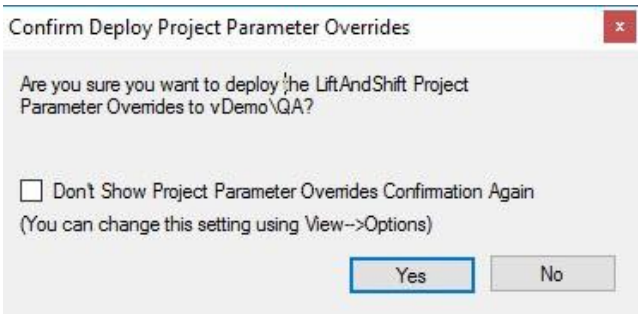


Figure 147

(…unless View→Options "Don't Show Project Overrides Deploy Confirmation Dialog" is selected as shown in Figure 148:

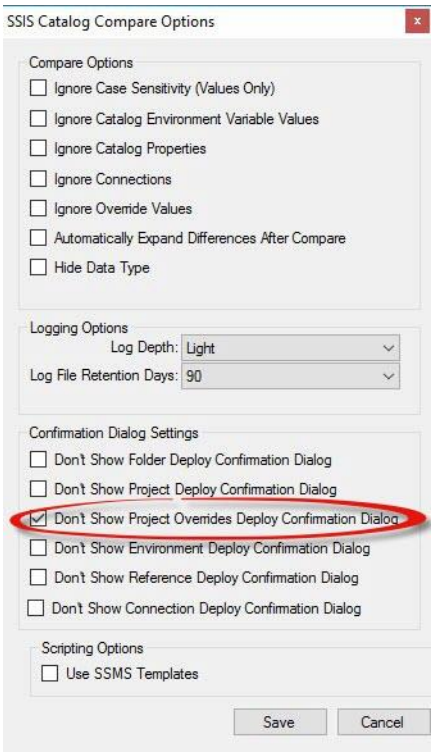


Figure 148

Once deployment is complete, the target treeview refreshes to display the updated target SSIS Catalog contents as shown in Figure 149:

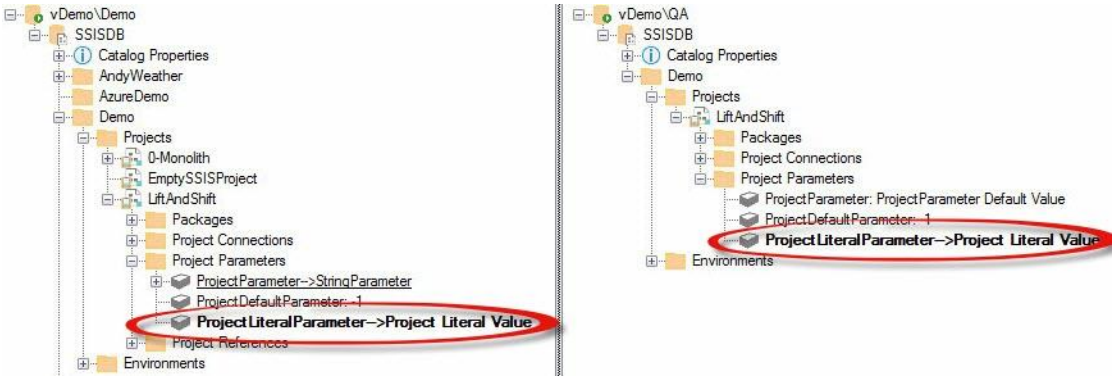


Figure 149

5.4 Deploy Environment

To deploy a Catalog Environment, right-click the Environment and click "Deploy Environment" as shown in Figure 150:

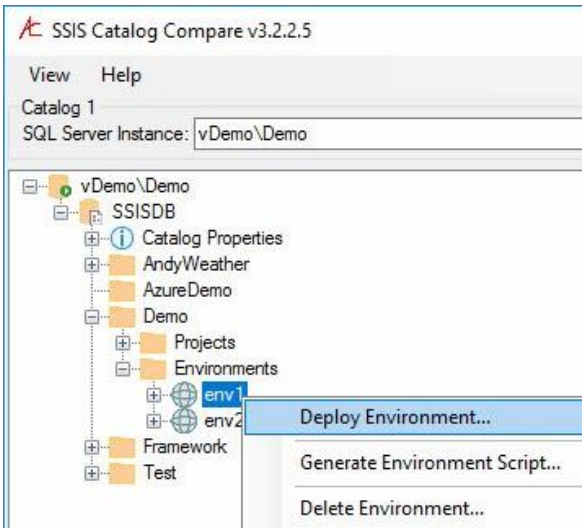


Figure 150

The "Browse For Folder" dialog displays as shown in Figure 151:

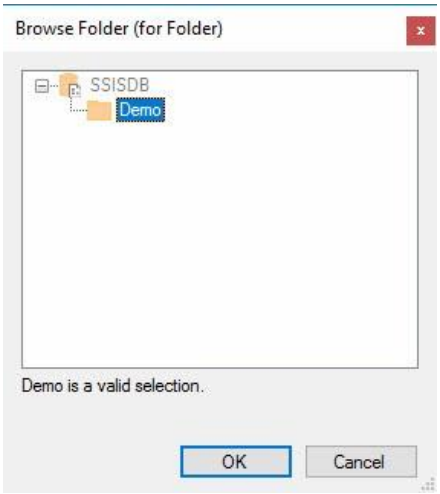


Figure 151

The Confirm Deploy Environment dialog displays as shown in Figure 152:

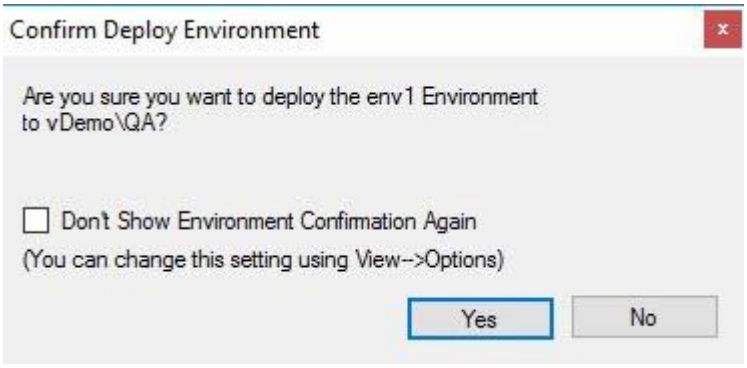


Figure 152

…unless View→Options "Don't Show Environment Deploy Confirmation Dialog" is selected as shown in Figure 153:

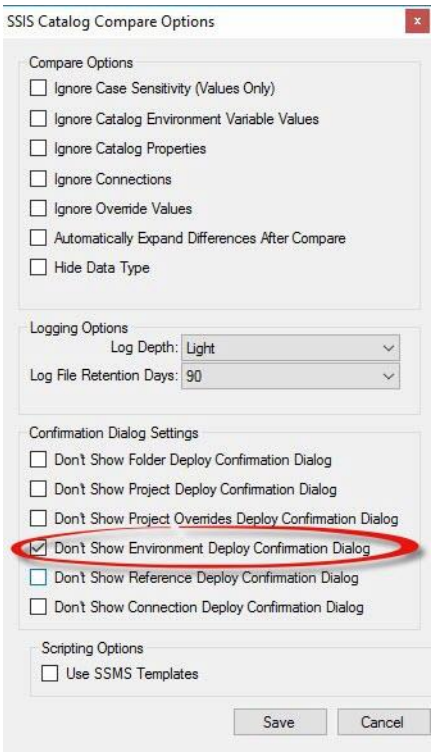


Figure 153

Once deployment is complete, the target treeview refreshes to display the updated target SSIS Catalog contents as shown in Figure 154:

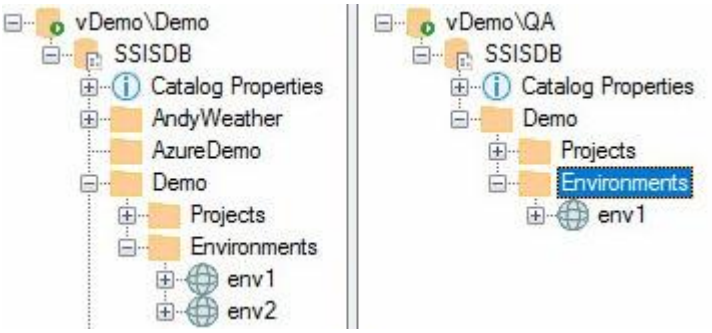


Figure 154

5.5 Deploy Project Reference

To deploy a Project Reference, right-click the Reference and click "Deploy Reference" as shown in Figure 155:

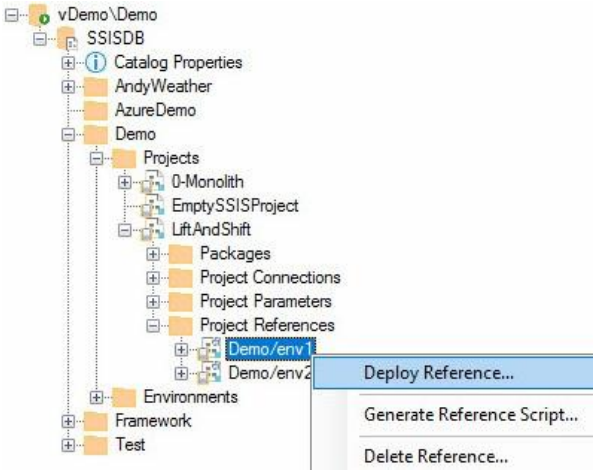


Figure 155

The "Browse Project (for Reference)" dialog displays as shown in Figure 156:

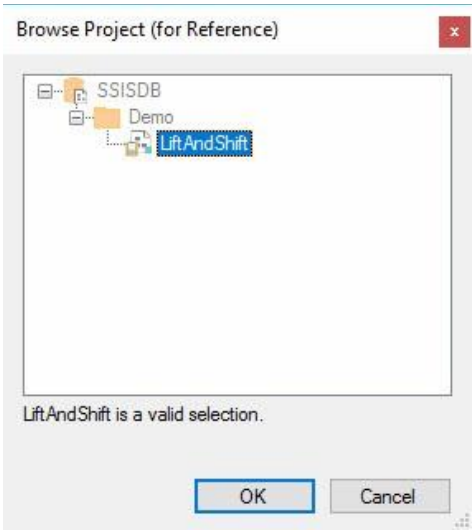


Figure 156

The "Browse Environment (for Reference)" dialog displays as shown in Figure 157:

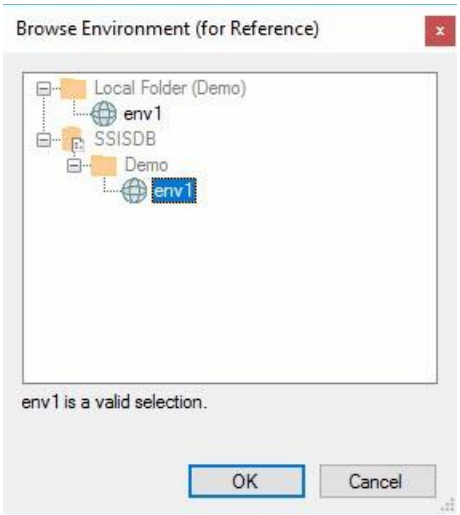


Figure 157

The Confirm Deploy Reference dialog displays as shown in Figure 158:

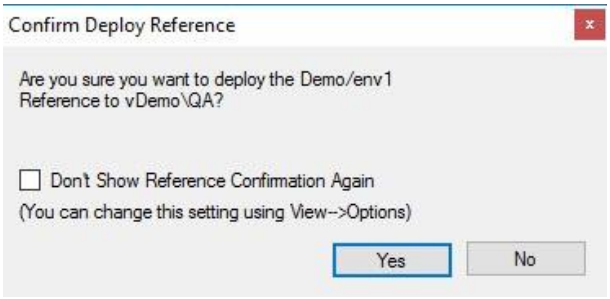


Figure 158

(…unless View→Options "Don't Show Reference Deploy Confirmation Dialog" is selected as shown in Figure 159:

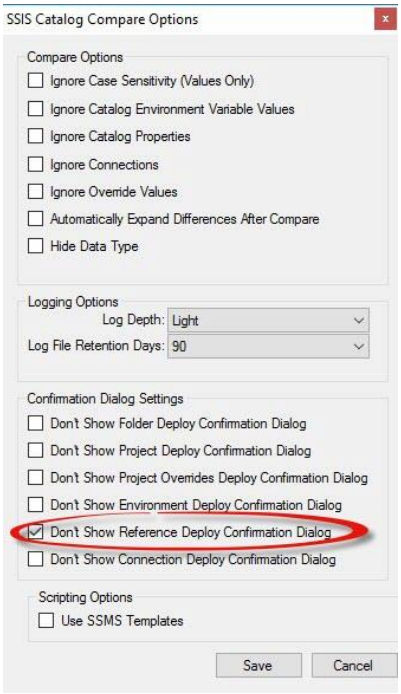


Figure 159

Once deployment is complete, the target treeview refreshes to display the updated target SSIS Catalog contents as shown in Figure 160:

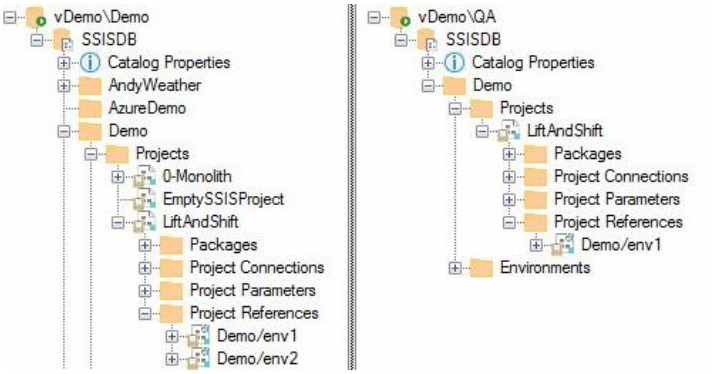


Figure 160

5.6 Deploy Project Connection Reference

To deploy a Project Connection Reference, right-click a Project Connection Reference node and click Deploy Reference as shown in Figure 161:

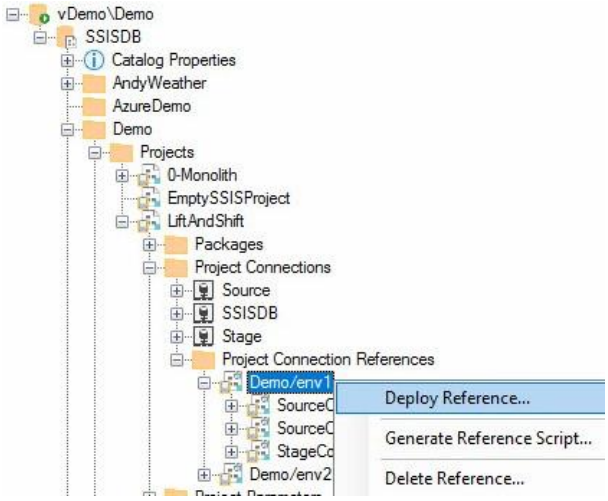


Figure 161

The user is prompted to select a target project as shown in Figure 162:

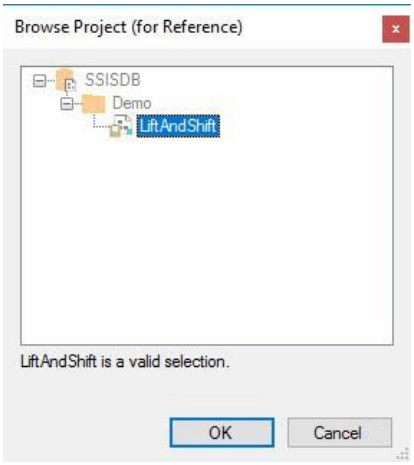


Figure 162

The user is prompted to select a target environment as shown in Figure 163:

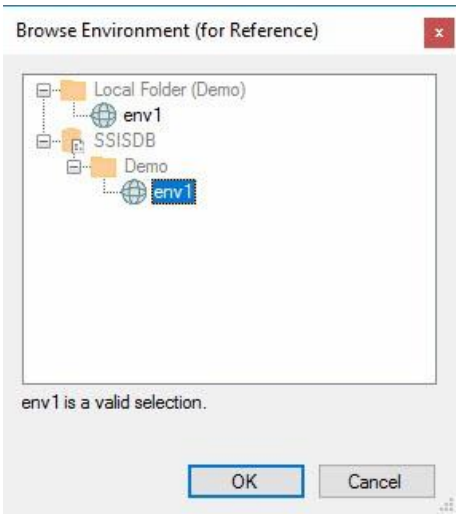


Figure 163

The user is asked to confirm the overwrite of the Reference (if the Reference exists), as shown in Figure 164:

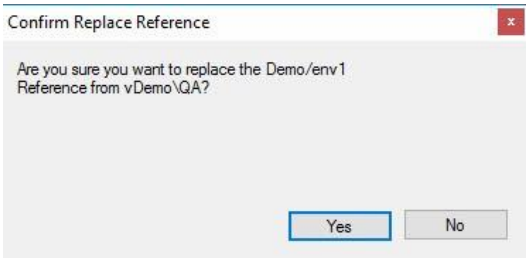


Figure 164

The user is asked to confirm deployment of the Reference (if the Reference exists), as shown in Figure 165:

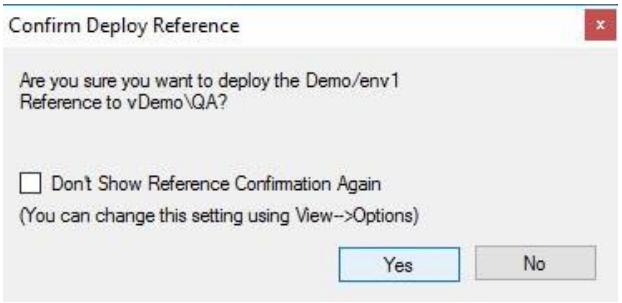


Figure 165

…unless View→Options "Don't Show Reference Deploy Confirmation Dialog" is selected as shown in Figure 166:

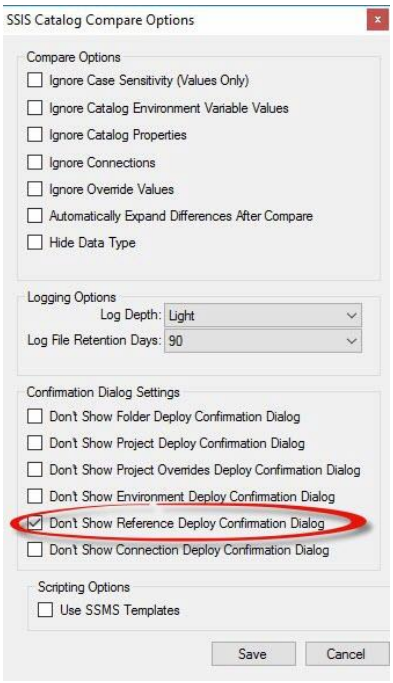


Figure 166

Once deployed, the Project Reference is surfaced and displayed as shown in Figure 167:

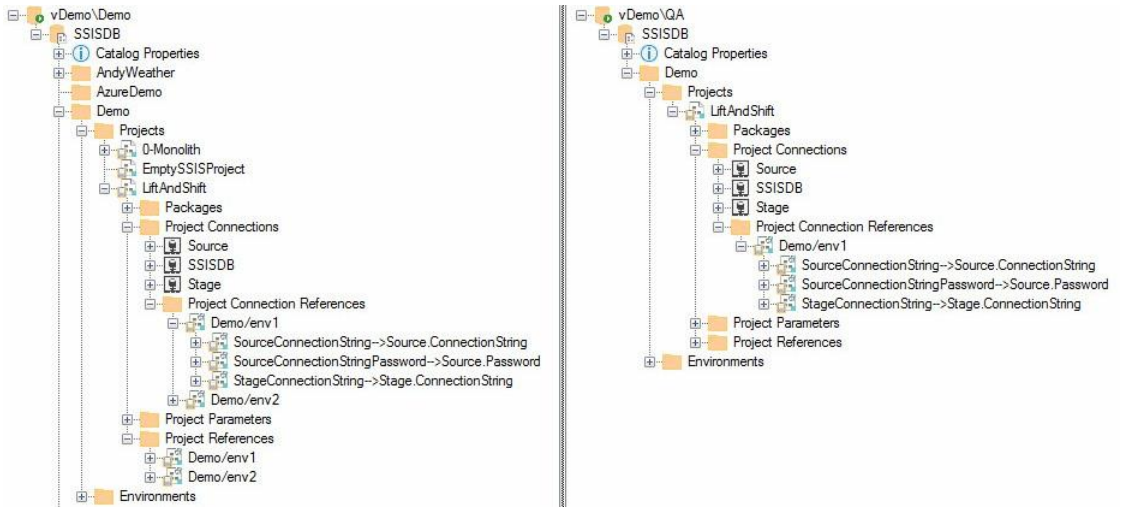


Figure 167

5.7 Deploy Package Reference

To deploy a Package Reference, right-click the Reference and click "Deploy Reference" as shown in Figure 168:

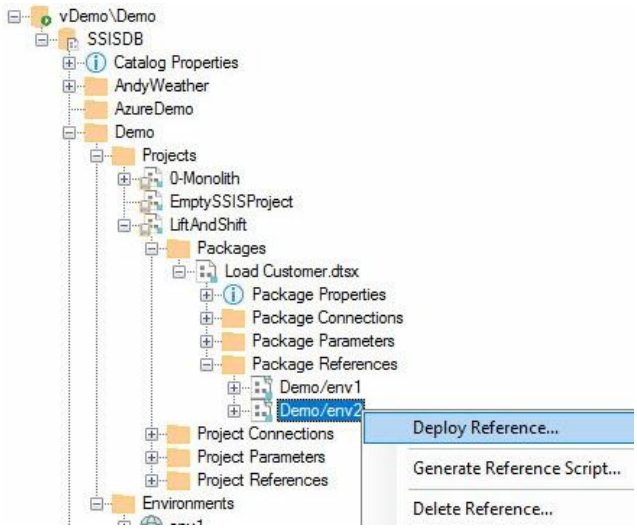


Figure 168

The "Browse Package (for Reference)" dialog displays as shown in Figure 169:

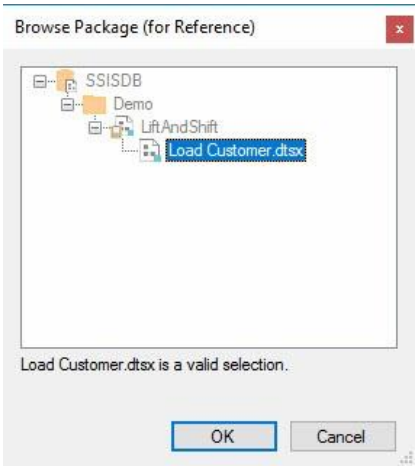


Figure 169

The "Browse Environment (for Reference)" dialog displays as shown in Figure 170:

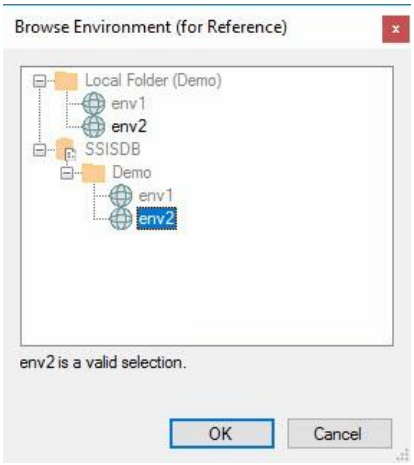


Figure 170

The Confirm Deploy Reference dialog displays as shown in Figure 171:

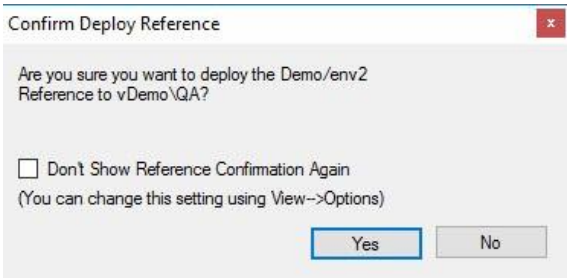


Figure 171

(…unless View→Options "Don't Show Reference Deploy Confirmation Dialog" is selected as shown in Figure 172:

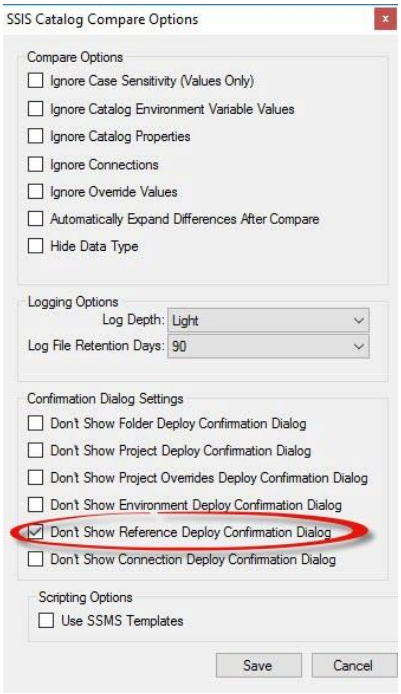


Figure 172

Once deployment is complete, the target treeview refreshes to display the updated target SSIS Catalog contents as shown in Figure 173:

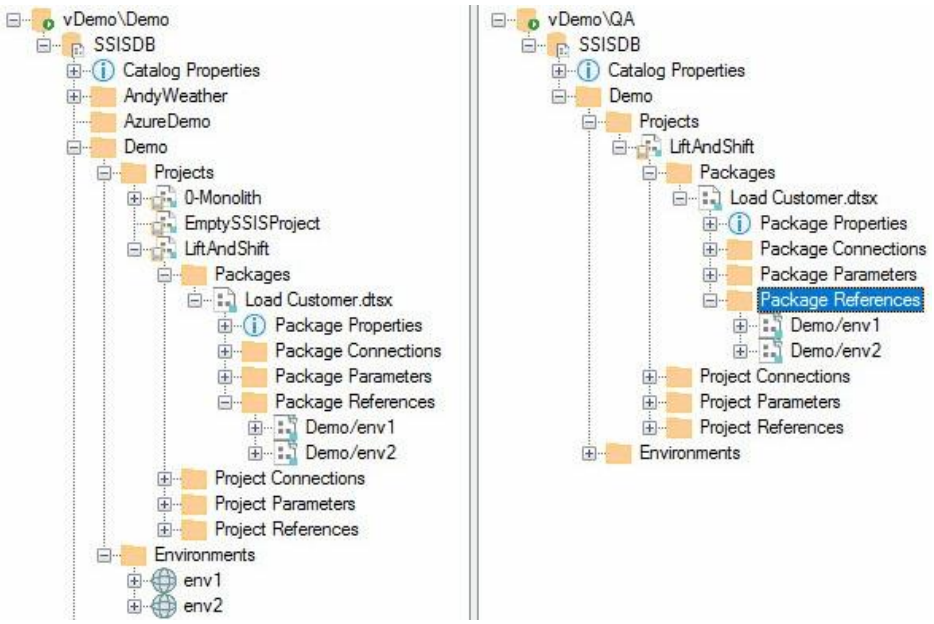


Figure 173

5.8 Deploy Package Connection Reference

To deploy a Package Connection Reference, right-click a Package Connection Reference node and click Deploy Reference as shown in Figure 174:

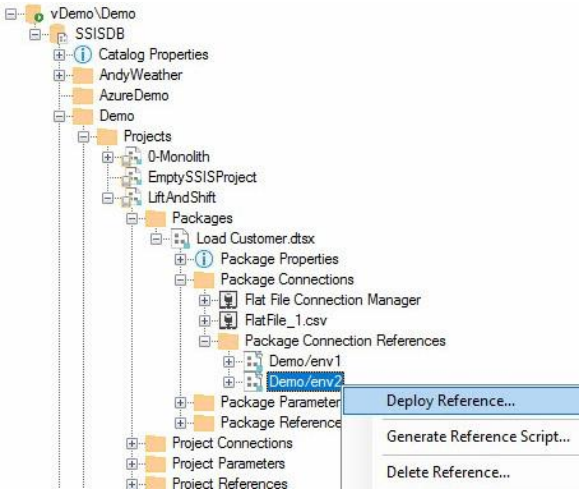


Figure 174

The user is prompted to select a target package as shown in Figure 175:

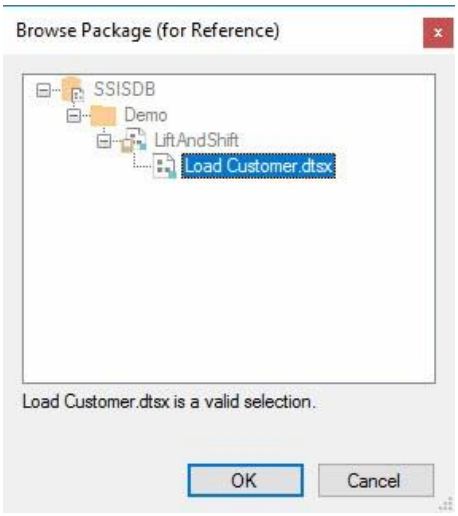


Figure 175

The user is prompted to select a target environment as shown in Figure 176:

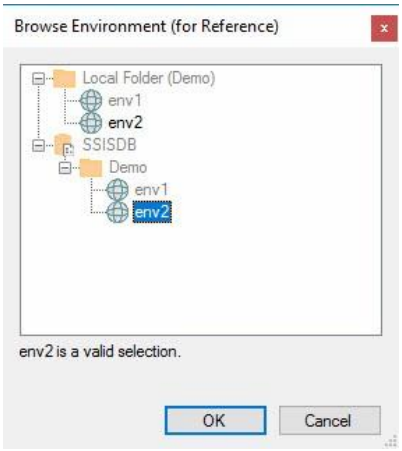


Figure 176

The user is asked to confirm the overwrite of the Reference (if the Reference exists), as shown in Figure 177:

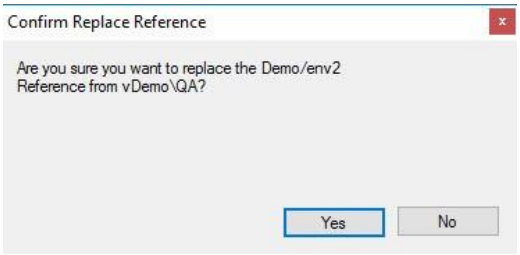


Figure 177

The user is asked to confirm deployment of the Reference (if the Reference exists), as shown in Figure 178:

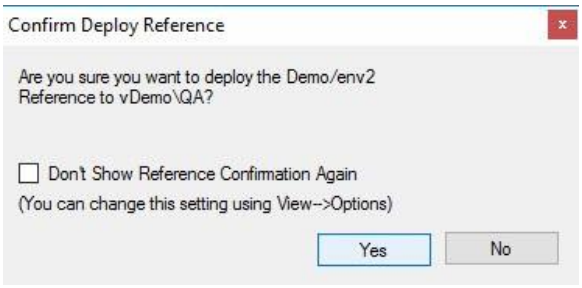


Figure 178

…unless View→Options "Don't Show Reference Deploy Confirmation Dialog" is selected as shown in Figure 179:

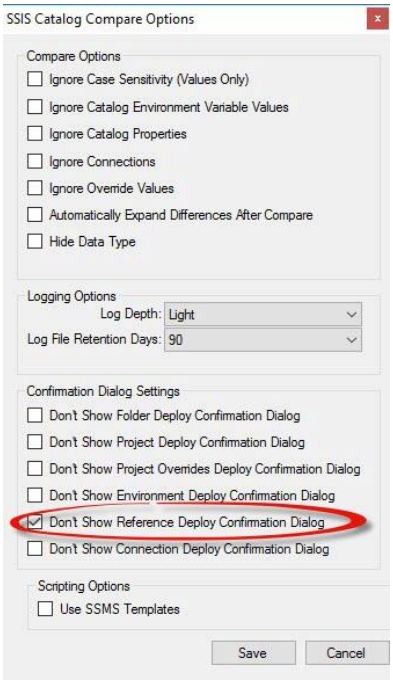


Figure 179

Once deployed, the Project Reference is surfaced and displayed as shown in Figure 180:

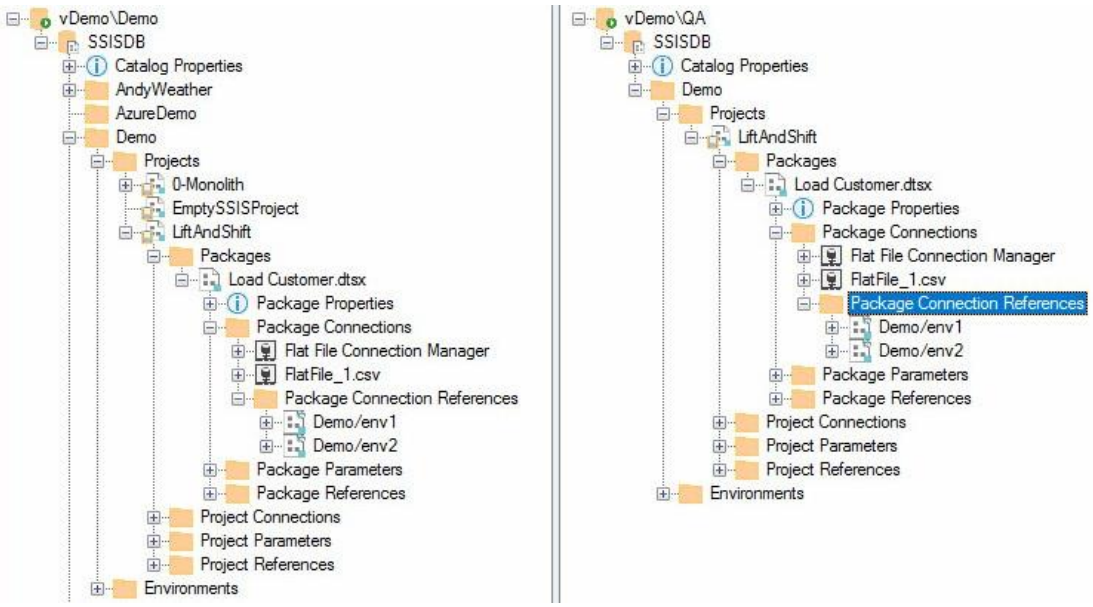


Figure 180

6 DELETE CATALOG ARTIFACTS

SSIS Catalog Compare provides delete functionality for eleven SSIS Catalog artifacts:

1. Folders
2. Projects
3. Environments
4. Project References
5. Project Connection References
6. Package References
7. Package Connection References

6.1 DELETE FOLDER

To delete a Catalog folder, right-click the folder and click Delete Folder as shown in Figure 181:

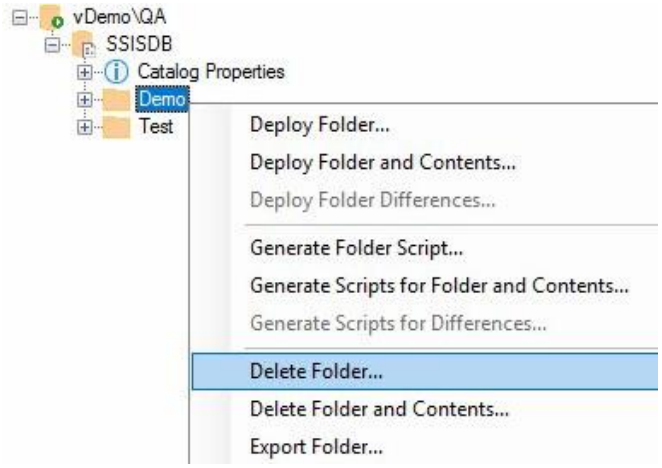


Figure 181

The user is prompted to confirm folder deletion as shown in Figure 182:

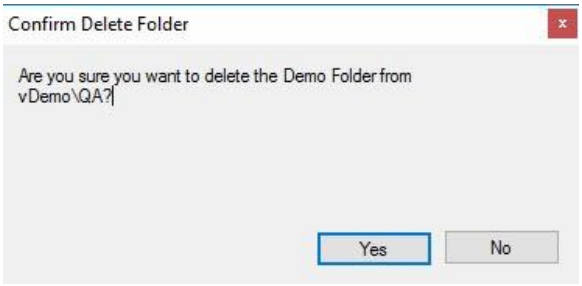


Figure 182

If the folder contains additional Catalog artifacts, the user is prompted *again* to confirm the deletion of the folder *and* all contents as shown in Figure 183:

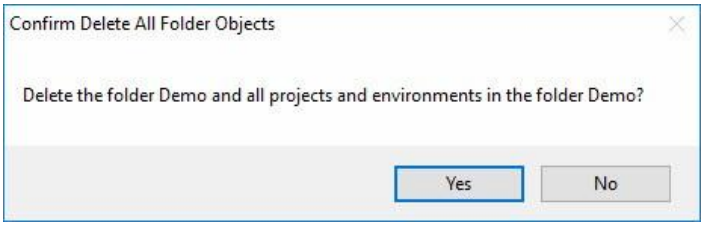


Figure 183

If the folder contents include SSIS Catalog environments that participate in references, the user must confirm the deletion of these Catalog Environments.

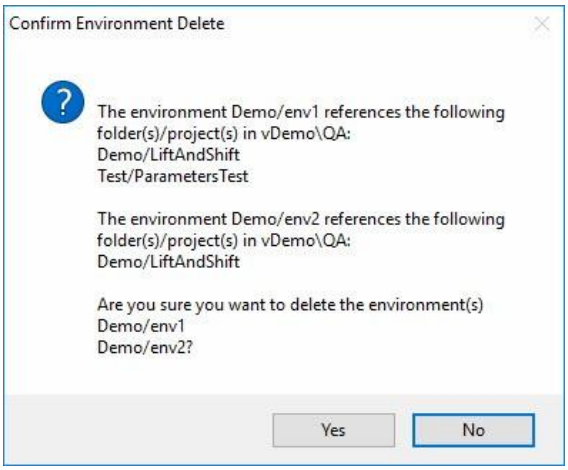


Figure 184

6.1.1 Some Important Notes on References and Environments

SSIS Catalog Compare informs the user if they are about to delete an SSIS Catalog Environment that participates in a reference.

Catalog Environments may exist in any SSIS Catalog Folder.

References define a relationship between an SSIS Catalog Environment and an SSIS Project or Package.

The SSIS Catalog Environment *is not* required to reside in the same SSIS Catalog Folder.

The SSIS Catalog permits the deletion of Catalog Environments that participate in references.

SSIS Catalog Compare identifies references for which the Catalog Environment does not exist as *Broken References*. A broken reference is shown in Figure 185:

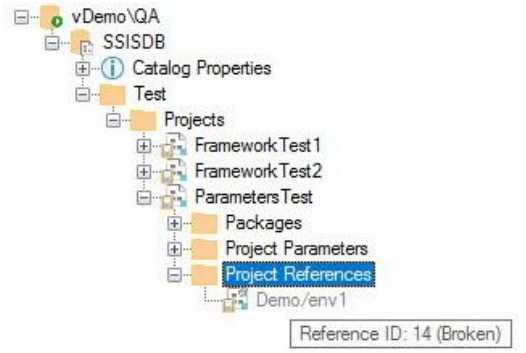


Figure 185

Broken references can leave data engineers and operations folks in a serious bind. Avoid broken references at all costs.

6.2 DELETE SSIS PROJECT

To delete an SSIS Project using SSIS Catalog Compare, right-click an SSIS Project and click Delete Project as shown in Figure 186:

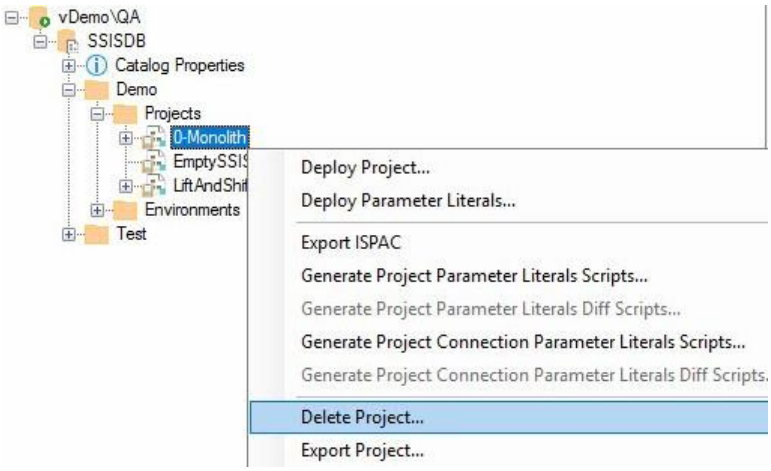


Figure 186

The user is prompted to confirm the deletion of the SSIS Project as shown in Figure 187:

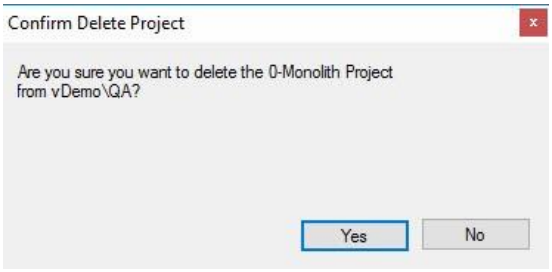


Figure 187

If the user clicks Yes, the project is deleted as shown in Figure 188:

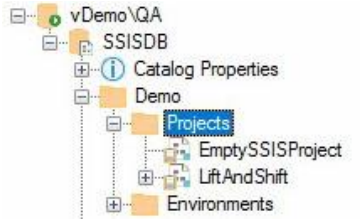


Figure 188

6.3 DELETE ENVIRONMENT

To delete an SSIS Environment using SSIS Catalog Compare, right-click an SSIS Catalog Environment and click Delete Environment as shown in Figure 189:

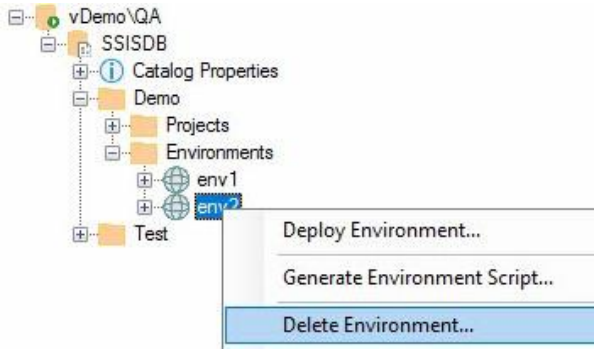


Figure 189

The user is prompted to confirm the deletion of the SSIS Environment as shown in Figure 190:

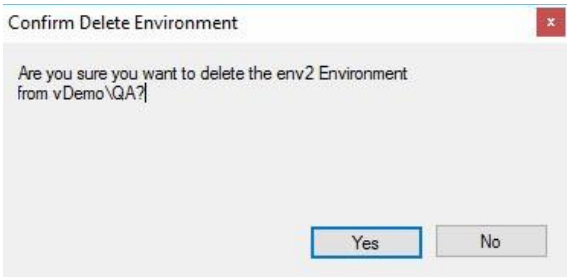


Figure 190

If the user clicks Yes and the Environment participates in a Reference, the user is prompted with Reference metadata and asked to re-confirm the Delete operation as shown in Figure 191:

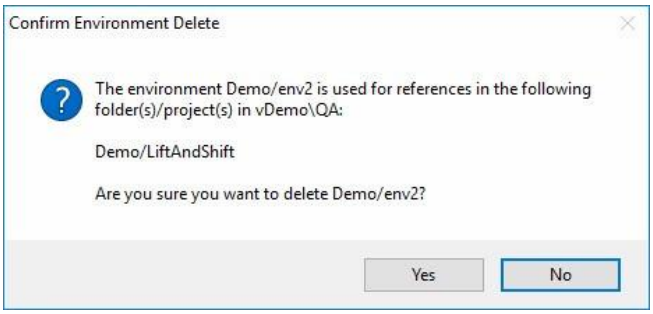


Figure 191

If the user clicks Yes (again), the environment is deleted as shown in Figure 192:

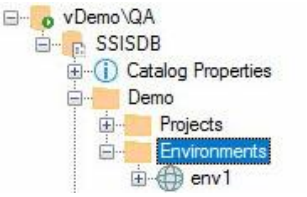


Figure 192

6.4 Delete Project References

6.4.1 A Note About Deleting References

Starting with version 3, SSIS Catalog Compare includes a feature called *Values Everywhere*. Values Everywhere means SSIS Catalog Environment Variable values are surfaced beneath nodes related to their consumption in a Reference Mapping. This means the same reference may appear beneath any of the following nodes:

- Project Reference
- Project Connection Reference
- Package Reference
- Package Connection Reference
- *Any* parameter (or Connection property)

When deleting References it is most important users realize that Values Everywhere is a construct of SSIS Catalog Compare and *does not* reflect the physical storage of the parameters, values, or reference mappings.

Therefore, when a reference is deleted from any location, it is removed from *all locations* in an SSIS Catalog Project.

To delete an SSIS Project Reference using SSIS Catalog Compare, right-click an SSIS Project Reference and click Delete Reference as shown in Figure 193:

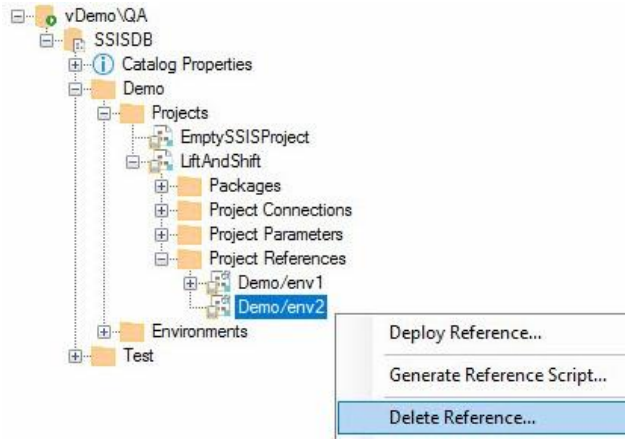


Figure 193

The user is prompted to confirm the deletion of the Reference as shown in Figure 194:

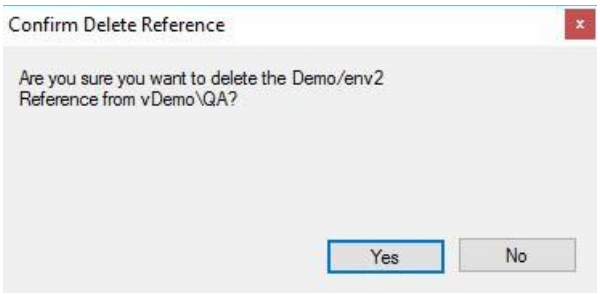


Figure 194

If the user clicks Yes, the reference is deleted as shown in Figure 195:

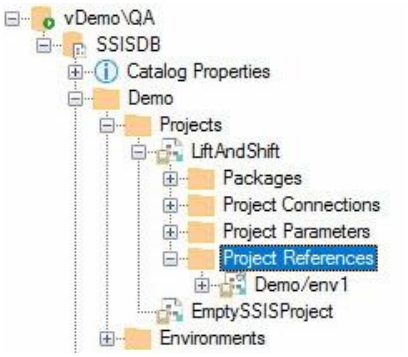


Figure 195

6.5 Delete Project Connection Reference

To delete an SSIS Project Connection Reference using SSIS Catalog Compare, right-click an SSIS Project Connection Reference and click Delete Reference as shown in Figure 196:

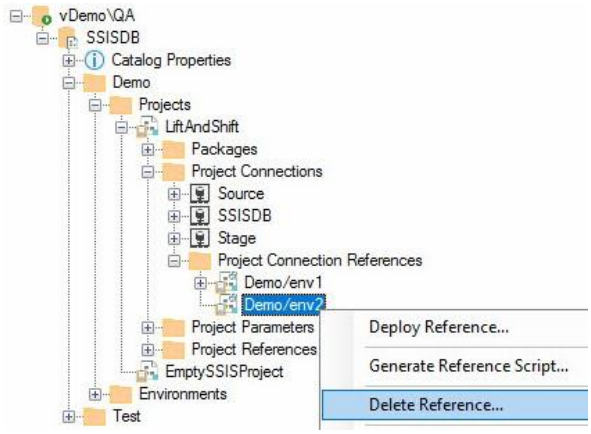


Figure 196

The user is prompted to confirm the deletion of the Connection Reference as shown in Figure 197:

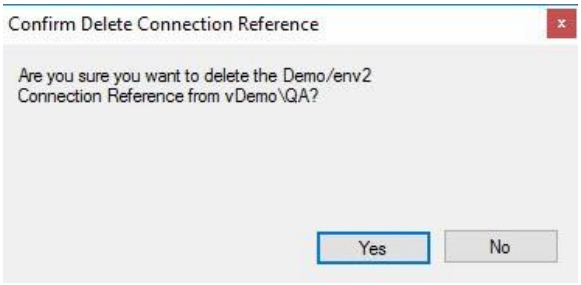


Figure 197

If the user clicks Yes, the connection reference is deleted as shown in Figure 198:

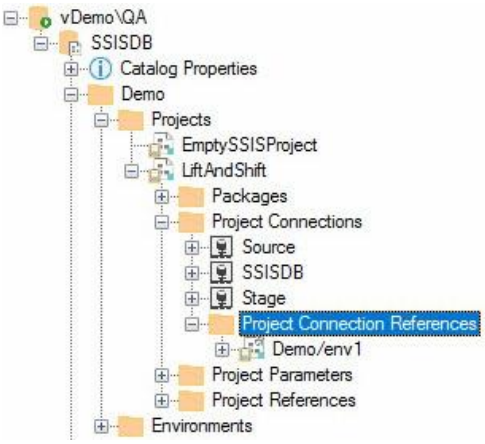


Figure 198

Please see A Note About Deleting References.

6.6 Delete Package Reference

To delete an SSIS Package Reference using SSIS Catalog Compare, right-click an SSIS Package Reference and click Delete Reference as shown in Figure 199:

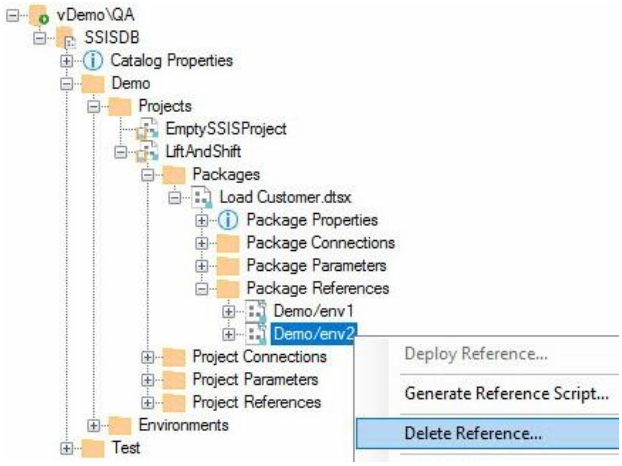


Figure 199

The user is prompted to confirm the deletion of the Reference as shown in Figure 200:

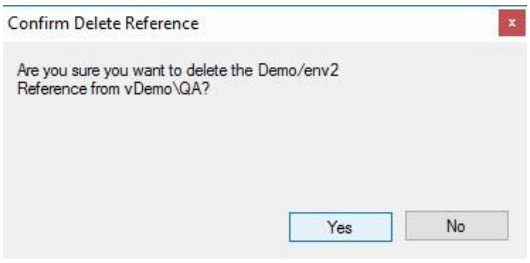


Figure 200

If the user clicks Yes, the reference is deleted as shown in Figure 201:

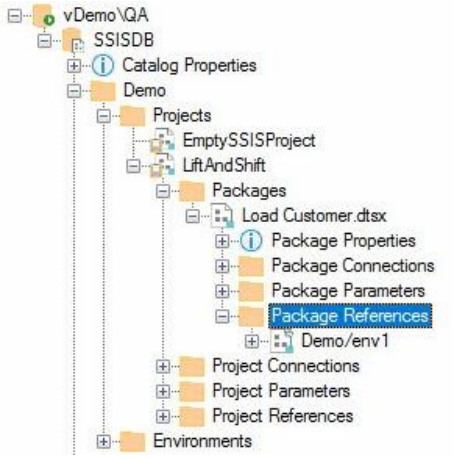


Figure 201

6.7 Delete Package Connection Reference

To delete an SSIS Package Connection Reference using SSIS Catalog Compare, right-click an SSIS Package Connection Reference and click Delete Reference as shown in Figure 202:

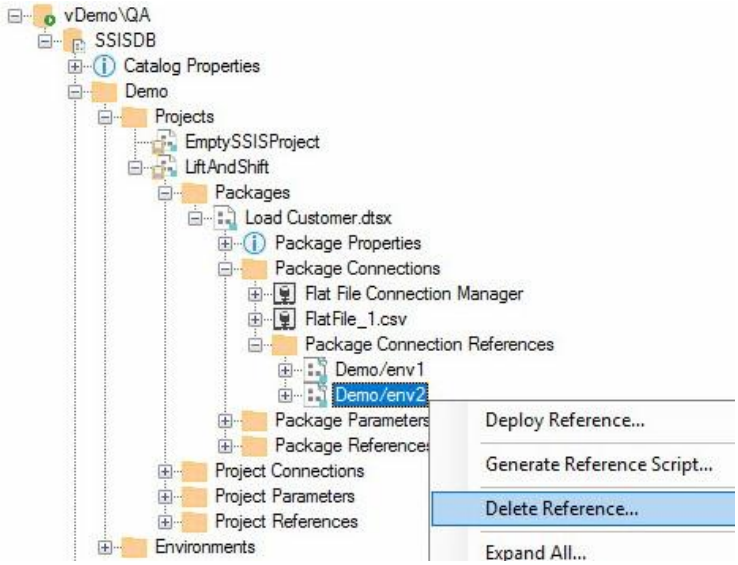


Figure 202

The user is prompted to confirm the deletion of the Connection Reference as shown in Figure 203:

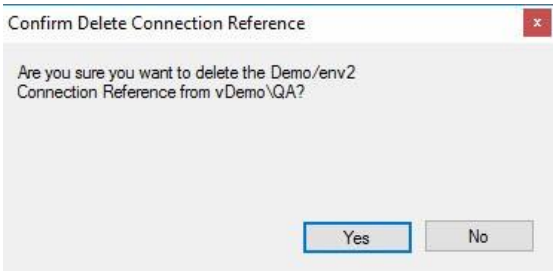


Figure 203

If the user clicks Yes, the connection reference is deleted as shown in Figure 204:

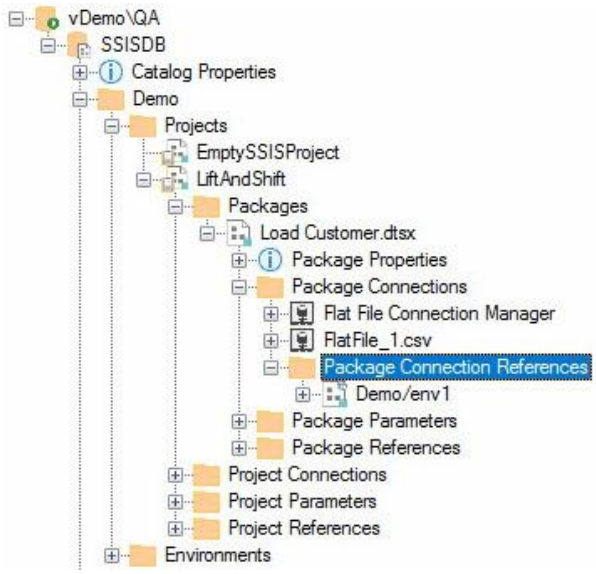


Figure 204

Please see A Note About Deleting References.

7 End User License Agreement

Andy Leonard Consulting

End User Software License Agreement

NOTICE: THIS AGREEMENT CONTAINS IMPORTANT TERMS AND CONDITIONS THAT REQUIRE YOUR COMPLIANCE. CAREFULLY READ THE ENTIRE DOCUMENT.

This Agreement describes the license terms between Andy Leonard Consulting ("Andy Leonard Consulting") and you (referred to in this Agreement alternately as "you", and "Customer") for use of the accompanying software ("Software") on a trial, permanent or subscription basis. To access and use this Software, you must accept the terms of this Agreement by clicking on acceptance prompt below the window in which this document appears. If you have already paid for a license to the Software and choose not to accept this Agreement, do not proceed to the Software, and contact Andy Leonard Consulting (andy@andyleonardconsulting.com) to obtain a refund. THIS AGREEMENT IS SUBJECT TO ARBITRATION.

1. LICENSE GRANT.

a) TRIAL LICENSE. Upon accepting this Agreement, Customer will be entitled to a one-time trial license at no charge. Unless otherwise agreed, the term of each trial license will be expired at a pre-determined interval. All of the terms and conditions of this Agreement apply with equal force to the trial license. Each trial license will expire at the end of the trial period, at which time Customer must either purchase a full license, or stop using the Software, and delete all copies of the Software and any portions of it that may have been downloaded or installed by Customer, its officers, employees, agents, representatives, or any other person that Customer allows to access the Software. Customers will not be entitled to take advantage of more than one free trial period with respect to the Software directly or through any other person or entity without the knowledge and express permission of Andy Leonard Consulting. During the term of a Trial License, Customers will receive the same access to Updates, Support and Maintenance as they would receive under a Full License.

b) FULL LICENSE. Customer may purchase a full license for use of the Software upon the terms and conditions set forth in this Agreement. If a full license is purchased, Customer will receive a license "key" consisting of a code with which to access the Software on a permanent or subscription basis.

(i) Permanent License. A Permanent License entitles Customer to perpetual use of the version of the Software as it exists on the day the License commences. A Permanent License includes all revisions, improvements, upgrades, patches, enhancements, fixes, and modifications offered by Andy Leonard Consulting (the "Updates") at no additional charge for twelve months following the commencement of the license. A Permanent License does not include any rights to subsequent Updates, or any releases of new versions of the Software or other existing or subsequently developed software of any kind that is separate from and in addition to the Software. Andy Leonard Consulting will have unlimited discretion as to whether to issue a software product as an Update of the Software, as a new version of the Software, or as a new version or Update of totally separate software.

(ii) Subscription License. If Customer purchases a Subscription License, Customer will pay a periodic license fee to access the Software. Each Subscription License will expire immediately upon the end of the last period for which payments have been received. A Subscription License includes free Updates, as offered by Andy Leonard Consulting

during the term of the license. A Subscription License does not give Customer any rights to releases of new versions of the Software or other existing or subsequently developed software of any kind that is separate from and in addition to the Software. Andy Leonard Consulting will have unlimited discretion as to whether it issues a software product as an Update of the Software, as a new version of the Software, or as a new version or Update of totally separate software.

c) CUSTOMER DEPLOYMENT OPTION. Customers who wish to make the Software available to their End Users on shared devices, or manage the assignment to and transfer of End User authorizations may select the customer deployment option when installing the Software (the "Customer Deployment Option"). The Customer Deployment Option may be elected for Permanent or Subscription Licenses.

2. SCOPE OF LICENSE.

Every license granted under this Agreement is non-exclusive and may not be assigned without the express written

consent of Andy Leonard Consulting. Each license commences when the Customer accepts this Agreement.

a) END USERS.

i) Access. Each license granted under this Agreement includes the right to allow one individual to access and use the Software in connection with the conduct of Customer's business operations (each such individual being referred to herein as an "End User"). The number of licenses required is based on the number of authorized End Users and not the number of End Users concurrently accessing the system.

ii) Transfer. End User status may be transferred from one individual to another only in connection with permanent changes in job descriptions and general work assignments. End User status cannot not be shared or rotated or temporarily transferred between or among individuals to avoid purchasing a license for each End User.

iii) Registration and Acceptance. Each End User must be registered with Andy Leonard Consulting or a Customer who has elected the Customer Deployment Option. All use of the Software is subject to the terms of this Agreement. Customer may only give access to the Software to End Users who have been fully informed of their obligations under this Agreement and have agreed to be bound by this Agreement.

iv) Joint Responsibility. Both Customer and each of its End Users will be fully liable for any violation of this Agreement by the End User.

b) AUTHORIZED COPIES. Each permanent license granted under this Agreement includes the right to install two separate copies of the Software, or access a copy of the Software on a Andy Leonard Consulting website, provided that only one copy may be in use at any time. The Software may also be installed on shared devices that provide (i) virtualization or (ii) multiplexing capabilities that pool connections or share hardware across multiple users. Each such installation shall constitute one of the two authorized copies for each End User with access to the Software on the shared device. Remote access of the Software on any other device without copying or downloading the Software does not constitute a separate copy.

c) BACKUP COPY. Each Customer may create one backup copy of the Software exclusively for use in installing the Software on other devices consistent with this Agreement or replacing another licensed copy that is destroyed or

becomes unusable. The backup copy will not be counted against the Customer's authorized number of copies if its use is limited to Software installation.

d) USAGE LIMITATIONS. The Software is licensed, not sold. Customer may use the Software in combination

with Customer's other systems, but only as permitted in this Agreement. Customer may not attempt to

circumvent or work around any features or technical limitations of the software. Andy Leonard Consulting reserves all rights not

expressly licensed under this Agreement. Specifically, and without limitation, Customer is not permitted to:

· modify or create derivative works of the Software;

· emulate, clone, re-create, reverse engineer, decompile, disassemble or otherwise attempt to discover or replicate the binary code, source code design features, graphics, logos or any copyrighted or trademarked text or images included in the Software, or reduce any or all of the Software's code to human readable form;

· assign, transfer, rent, lease, lend, sublicense or otherwise transfer the license key, or any of Customer's rights in the Software without the express written consent of Andy Leonard Consulting;

· publish, redistribute or resell the Software or make it available for others to copy or purchase in whole or in part;

· include the Software or any of its parts or components as part of any other software, service, or product;

· create any copies of the Software except as expressly permitted in this Agreement;

· access application programmer interfaces that are not documented as End User accessible on Andy Leonard Consulting's website.

· delete any claims, markings or notices of patent, trademark or copyright ownership or registrations.

· use the Software in any act that is a violation of governing laws.

· install or access the Software, either directly or through commands, data or instructions from or to a computer that is not part of Customer's internal network.

· make the Software accessible to the public, or to any individuals who are not authorized End Users.

· use the Software in any way that would make the number of users uncountable.

· use the software for commercial or non-commercial software hosting services.

· use the software for Software as a Service (SaaS).

3. INFORMATION COLLECTION AND RECORDKEEPING.

The Software is designed to track and report various information to Andy Leonard Consulting and/or Customer's systems related to Software installation, authorized End Users, Licensed Usage Instances and Software performance and effectiveness. Customer and End Users agree to the collection of such information, and will not do anything to interrupt or interfere with it. Customer will register each licensed installation of a Software copy, and each authorized End User with Andy Leonard Consulting, unless Customer elected the Customer Deployment Option when installing the Software. Customers who elect the Customer Deployment Option must assume responsibility for tracking each licensed installation of a Software copy and each authorized End User. Customer agrees to keep such records for at least one calendar year, and provide them to Andy Leonard Consulting upon request in order to verify license compliance and Software improvement. Customer will only be required to provide data for one such request every 30 days. If Customer does not elect the Customer Deployment Option, the same information must be provided directly to Andy Leonard Consulting.

4. UPDATES, SUPPORT AND MAINTENANCE.

a) AUTHORIZED UPDATES. Andy Leonard Consulting is the only authorized distributor of Updates. From time to time,

Andy Leonard Consulting may decide to issue Updates to correct deficiencies or errors in the Software that are reported to

Andy Leonard Consulting. Andy Leonard Consulting may not correct every deficiency or error that comes to its attention, and Andy Leonard Consulting will

have no obligation to issue any Updates. At its discretion, Andy Leonard Consulting may choose to include additional features

in Updates. These additional features may be subject to additional terms which Customer may be required to

separately review and accept as a condition to their use. From time to time, Andy Leonard Consulting may issue updates that it

considers critical to the performance of the Software or compliance with applicable laws. Failure to promptly

install such updates will terminate Customer's license, and result in deactivation of the software.

b) SUPPORT AND MAINTENANCE PLANS. Unless otherwise specified in a separate agreement, Andy Leonard Consulting will

not provide direct Customer support. Community supported forums will be provided at no charge through the

Andy Leonard Consulting website for current Software versions. Andy Leonard Consulting makes no guarantees with respect to responses to

support requests, but will make reasonable efforts to assist customers on the community support forums.

Customers purchasing a permanent license who wish to receive Updates beyond the first 12 months of their

license must purchase a maintenance plan. Information regarding the community support forums, maintenance plans and Updates can be obtained from a Andy Leonard Consulting sales representative, by submitting an e-mail to support@andyleonardconsulting.com.

5. LIMITED WARRANTY.

a) LIMITED ONE-YEAR WARRANTY. Andy Leonard Consulting represents and warrants that, to the best of its knowledge, the Software will function substantially in accordance with the documentation it provides to Customer for a period of 12 months following the commencement of each full license purchased by Customer, and that, to the best of its knowledge, the Software, as created by Andy Leonard Consulting, does not infringe or otherwise violate the intellectual property rights of any third party in the United States of America or in any other country where Andy Leonard Consulting offers it for sale. This warranty does not apply to defects, deficiencies and violations that were not known by Andy Leonard Consulting at the time this Agreement was entered. If any such defects, deficiencies or violations were or in the exercise of reasonable prudence should have been known by Andy Leonard Consulting prior to the execution of this Agreement by Customer, Andy Leonard Consulting will either (i) obtain a license to avoid the infringement, (ii) create an Update to solve the problem or avoid the infringement, (iii) replace the Software with a new version that solves the problem or avoids the infringement at no cost to Customer, and/or (iv) terminate Customer's license for use of the Software and refund any portion of the license fee that is reasonably allocable to the remaining term of the license. To take advantage of this warranty, Customer must request a remedy in writing during the 12 month period after it enters this Agreement via e-mail addressed to support@andyleonardconsulting.com. Andy Leonard Consulting may, but is not required to take such measures with respect to other defects, deficiencies and or violations. Customer agrees to cooperate with all efforts by Andy Leonard Consulting to effectuate any such measures, or, if necessary, at the request of Andy Leonard Consulting, terminate any pending license and delete and destroy all relevant Software copies.

In all other respects, and to the fullest extent permissible by law, the Software is sold "as is" and without any warranty by Andy Leonard Consulting whatsoever as to its use, performance or otherwise. Andy Leonard Consulting does not warrant that the Software will be free of all defects and deficiencies; that it will be error-free, function without interruptions, or other failures, that it will meet all of Customers performance expectations, regardless of whether those expectations have been made known to Andy Leonard Consulting, or that the Software will not violate some party's intellectual property rights in a relevant jurisdiction.

b) EXCLUSIONS. Andy Leonard Consulting does not warrant against defects, deficiencies, or intellectual property violations resulting from or related to problems with Customer's equipment or systems; incompatibility between Customer's equipment or systems and the Software; combinations of the Software by Customer with Customer's other software, equipment or systems; misuse; abuse; neglect; improper installation, use or maintenance; theft; vandalism; acts of God; acts of terrorism; power failures or surges; casualty; alteration; modification; defects, deficiencies or infringements not reported to Andy Leonard Consulting promptly after they become known to Customer; or failure to cooperate with efforts by Andy Leonard Consulting to solve a problem or avoid infringement.

c) EXCLUSION OF IMPLIED WARRANTIES. TO THE FULLEST EXTENT PERMISSIBLE BY LAW, Andy Leonard Consulting DISCLAIMS ALL IMPLIED WARRANTIES OF THE MERCHANTABILITY, WORKMANLIKE QUALITY OR FITNESS FOR AN INTENDED PURPOSE WITH RESPECT TO THE SOFTWARE. TO THE EXTENT THAT LOCAL LAW MAY NOT PERMIT SUCH

DISCLAIMERS, THE IMPLIED WARRANTY IS LIMITED TO THE DURATION OF THE EXPRESS WARRANTY (12 months). THE LAWS OF YOUR STATE OR JURISDICTION MAY NOT PERMIT SUCH DISCLAIMERS AND LIMITATIONS, SO YOUR RIGHTS MAY VARY.

d) LIMITED DAMAGES. Andy Leonard Consulting WILL NOT BE LIABLE TO CUSTOMER, ANY END USER OR ANY OTHER PARTY FOR ANY LOSS OF REVENUE OR PROFIT, BUSINESS INTERRUPTION, LOSS OR DAMAGE TO DATA, OR FOR ANY OTHER CONSEQUENTIAL, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES THAT MIGHT BE ASSERTED BASED UPON NEGLIGENCE, RECKLESSNESS, INTENTIONAL WRONGDOING, STRICT LIABILITY, BREACH OF CONTRACT, INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, OR ANY OTHER LEGAL OR EQUITABLE THEORY OF RELIEF THAT MAY BE ASSERTED WITH RESPECT TO THE SOFTWARE OR ANY TERMS OF THIS AGREEMENT, EVEN IF Andy Leonard Consulting MAY HAVE KNOWN OR HAD REASON TO KNOW OF THE POSSIBILITY THAT SUCH DAMAGES WOULD OR COULD RESULT. ANY DAMAGES THAT MAY BE ASSESSABLE AGAINST Andy Leonard Consulting IN CONNECTION WITH THIS AGREEMENT SHALL BE LIMITED TO $100.00. THE LAWS OF YOUR STATE OR JURISDICTION MAY NOT PERMIT THE EXCLUSION OF CONSEQUENTIAL DAMAGES OR OTHER LIMITATIONS, SO YOUR RIGHTS MAY VARY.

6. DISPUTE RESOLUTION.

a) APPLICABLE LAW: This Agreement shall in all respects be governed by the laws of the State of South Carolina. The parties to this Agreement specifically disclaim the U.N. Convention on Contracts for the International Sale of Goods.

b) ARBITRATION. To the fullest extent permissible under applicable laws, all disputes arising under or in connection with this Agreement that are not resolved through direct negotiation or mediation will be resolved by binding arbitration to be conducted in Greenville, South Carolina, USA under the rules of the American Arbitration Association (the “AAA”). All disputes will be arbitrated before a single arbitrator under AAA’s Expedited Procedure rules, subject to the consent of Andy Leonard Consulting which may be withheld regardless of the amount in controversy.

c) JURISDICTION AND VENUE. This Agreement is performable in Greenville County, South Carolina. Venue of any suit or other judicial proceeding hereunder shall be properly placed in Greenville County, South Carolina, and both parties waive any objection to venue of any such proceeding therein. All lawsuits or judicial proceedings of any kind brought by Customer or End User with respect to this Agreement or a license granted pursuant to this Agreement shall be brought in the state and federal courts of South Carolina.

7. TERMINATION. This Agreement remains in effect until terminated.

a) TERMINATION BY CUSTOMER. Customer may terminate the Agreement and/or any license(s) that are subject to its terms at any time by providing written notice to Andy Leonard Consulting of the termination by e-mail at support@andyleonardconsulting.com and by deleting and destroying all copies of the Software in its possession or control or the possession or control of its End Users, including any backup copies. Any fees paid prior to termination by Customer are not refundable, and termination shall not relieve Customer of any obligation to pay accrued fees or charges.

b) TERMINATION BY Andy Leonard Consulting.

i) Termination for Cause. If Customer or any of Customer's End Users fails to comply with this

Agreement, Andy Leonard Consulting may terminate this Agreement and/or any license(s) that are subject to its terms effective immediately, or if Andy Leonard Consulting asks Customer to take actions to remedy the violation, upon failure by Customer to do so. Upon termination of this Agreement by Andy Leonard Consulting, the Customer must delete and destroy all copies of the Software in its possession or control or the possession or control of its End Users, including any backup copies. Any fees paid prior to termination by Andy Leonard Consulting for cause are not refundable, and termination shall not relieve Customer of any obligation to pay accrued fees or charges.

ii) Termination without Cause. Andy Leonard Consulting may terminate this Agreement and/or any licenses that are subject to its terms for any reason or no reason, effective upon notice or such later date as Andy Leonard Consulting may indicate in the notice. In such cases, Andy Leonard Consulting will endeavor to provide reasonable notice prior to the termination, but will have no obligation to do so, and Andy Leonard Consulting will refund or provide a credit to Customer against outstanding payments for all pre-paid and unaccrued fees received prior to termination, on a pro-rata basis.

4. GENERAL PROVISIONS.

a) CONSTRUCTION: The section headings in this Agreement are for provided solely for convenience. They are

not to be used in interpreting this Agreement. The term "including" means "including without limitation."

b) SEVERABILITY: It is agreed that if the scope of any restriction herein expressed is too broad to permit

enforcement of such restriction to its full extent, then Employee agrees and consents to the enforcement of such restriction to the maximum extent permitted by applicable law.

c) ENTIRE AGREEMENT: This Agreement represents the sole and entire agreement with regard to the subject

matter hereof between the parties and supersedes any and all other agreements, written or oral, between them. No waiver or modification of any term of this Agreement shall be effective unless in writing and duly executed by the party to be charged therewith. In the event of any conflict between the terms of this Agreement and any other agreement entered between Andy Leonard Consulting and Customer, the terms of this Agreement will take precedence.

d) VIOLATIONS. Customer must promptly report any violation of this Agreement, and any violation of any third

party's intellectual property that comes to its attention, use its best efforts to avoid or limit any damages to Andy Leonard Consulting as a result of the violation, and cooperate fully with Andy Leonard Consulting in any investigation of the violation and enforcement of Andy Leonard Consulting's rights in connection with same.

e) EXPORT RESTRICTIONS. United States laws and regulation govern the export of the Software. Andy Leonard Consulting is not

aware of any restrictions imposed on export of the Software under such laws and regulations, but assumes no responsibility for compliance with them. Customer is responsible for compliance with any such restrictions, and all rights to use the Software under this Agreement are conditional on such compliance. Customer will indemnify, defend and hold harmless Andy Leonard Consulting from and against any claims, penalties, loss or damage arising out of a breach of said laws and regulations.

f) NON-WAIVER AND SEVERABILITY. Failure by Andy Leonard Consulting to enforce any term or provision of this Agreement

shall not constitute a waiver of the breach, or any future breach of the term or provision. If any terms or provisions of this Agreement are found by a court in any jurisdiction to be invalid or unenforceable, said terms and conditions will be deemed severed from the agreement in said jurisdiction and its remaining terms and conditions will remain in full force and effect to the fullest extent possible without frustrating its overall purpose and intent.

g) PRIVACY POLICY. Aggregate data about software usage created with information collected by Andy Leonard Consulting may

be made available to third parties by Andy Leonard Consulting, but no personal information or other information about Customers, Users or data entered into Andy Leonard Consulting equipment or systems will be disclosed to third parties. Provided, however, that Andy Leonard Consulting will be free to collect and use such information for purposes of monitoring compliance with this Agreement, and analyzing software performance, customer needs and opportunities to improve Andy Leonard Consulting products.

h) INTELLECTUAL PROPERTY RIGHTS. Andy Leonard Consulting retains ownership of all intellectual property rights of any kind

relating to the licensed software. The only intellectual property rights granted under this Agreement are a non-exclusive license for use of the licensed software upon the terms expressly stated herein. Nothing in this Agreement shall be deemed to grant or imply the grant of any additional intellectual property rights to Customer or any End User. Nor shall Andy Leonard Consulting be deemed to have agreed to hold in confidence any information disclosed to Andy Leonard Consulting or made available to Andy Leonard Consulting by Customer or any End User through use of the Software, or communications or feedback regarding customer support and software maintenance or to compensate Customer or any End User for the use thereof except as expressly stated herein.

i) AMENDMENT. This Agreement may be amended at any time by Andy Leonard Consulting for any or no reason with respect to

all terms of the Agreement for which a license fee has not already been paid, effective immediately upon posting the updated version on the Andy Leonard Consulting website for the software on which the prior version appeared. Andy Leonard Consulting will use reasonable efforts to notify existing Customers of the amendments, but shall have no obligation to do so, and the posted amendment will be effective regardless of whether such notice was accomplished or even attempted.

j) ASSIGNMENT. This Agreement and the license granted herein may be assigned by Customer only with the

written consent of Andy Leonard Consulting, which shall not be unreasonably withheld. Andy Leonard Consulting may assign this Agreement without prior approval or consent from Customer or any other party.

k) RELATIONSHIP OF PARTIES. This agreement is strictly a license agreement. Nothing stated in or implied by this

Agreement will be deemed to establish a joint venture, partnership, employment or agency agreement of any kind.

l) SURVIVAL OF TERMS. All of Andy Leonard Consulting's remedies for reverse engineering or other violations of this

Agreement, and any other terms which, by their nature, are intended to survive termination of any license granted herein will remain in full force and effect regardless of such termination.

m) NOTIFICATION OF COPYRIGHT INFRINGEMENT. It is the intention of Andy Leonard Consulting to take full advantage of all

rights and safe harbors afforded to a "service provider" as that term is used in the Digital Millenium Copyright Act as codified at Section 512 of Title 17 of the United States Code of Laws (the "DMCA"), and such other comparable laws and regulations as may be applicable in any jurisdiction throughout the world with respect to information stored, transmitted, located or accessed by a user on or with a system or network controlled or operated by or for Andy Leonard Consulting. In furtherance thereof, Andy Leonard Consulting reserves the right to remove or disable access to any material placed by a user on such system or network upon receipt of a notice of claim that the material infringes a copyright, or reason to know or believe such to be the case, and to terminate access to and use of such system or network for any repeat infringer. It is the policy of Andy Leonard Consulting to replace any such material upon receipt of a counter notification in accordance with the terms of such laws.

n) AGENT FOR NOTICE. All notices to Andy Leonard Consulting, including any notice of copyright infringement and any counter notice from the alleged infringer shall be effective upon receipt by Andy Leonard Consulting at the address or addresses specified at www.andyleonardconsulting.com/legal at the time the notice is given, via e-mail with return confirmation of receipt or an overnight courier service with national recognition in the United States. Notices to Licensees shall be effective upon delivery to any e-mail, physical or postal address designated by that Licensee for communications regarding this Agreement, or in the absence thereof, to any e-mail, physical or postal address that Andy Leonard Consulting knows or has reason to believe is currently used by Licensee.

_____