

White Paper



SSIS Package Configurations

Nitesh Rai

Abstract

Managing changes to environment dependent variables is common and critical in any ETL application. Especially during deployment of SSIS packages they need to be modified appropriately to ensure smooth deployment. SQL Server Integration Services (SSIS) has various methodologies of managing such configurations during deployment of the packages. They help developers to handle such environment related changes with elegance. This paper explains all the available package configuration methodologies and recommendations for each of them. It helps in selection of appropriate configuration method based on the requirement.

- Target Audience: SSIS developers, SQL Server developers, and SSIS project managers
- Pre-requisites: Basic knowledge of package development using visual studio or business intelligence development studio and transact SQL.

1. Introduction

SQL Server Integration Services (SSIS) is part of Microsoft Business Intelligence (MS – BI) suite. This is a platform for Extraction, Transformation, and Loading (ETL) offered by Microsoft which has raised the bar of its offering in BI space. A component developed by using features of SSIS is referred to as a package.

As part of software development life cycle (SDLC), the package can be shifted/transferred across various environments like development and unit testing, system testing, UAT and production. Most packages will have environment specific variables like connection string to a database or path to a flat file, or user defined variables etc. that would be impacted while moving the package across environments as part of deployment process. Hence, it is mandatory to change these environment dependent variables when the package is transferred across environments. Package configurations help in managing such changes without actually opening and editing the SSIS package in Business Intelligence Development Studio (BIDS). After deploying the package to a different machine (using SQL Server or file system deployment mode) it is mandatory to copy the related package configuration files on to that machine. If the package is scheduled to run through a SQL Agent job, the configuration file should be added while creating the job so that package will read the information from the configuration file. While executing the job, SQL Agent will take the design time values for connection strings if the package configuration file is not supplied.

This paper is intended to help developers understand various configuration modes available with SSIS in detail, their advantages and limitations. This white paper will enable users to decide upon the specific configuration mode that needs to be used based on the requirement.

2. Enabling Package Configuration

Package configuration should be enabled only when the package is completely developed. Following are the steps used to create an [SSIS project](#) and to add a configuration:

1. Select Project Types as [Business Intelligence Projects](#)
2. Select [Integration Services Project](#) as the template using Business Intelligence Development Studio or Visual Studio.
3. Develop the package and add Package Configuration by clicking [SSIS](#) menu and select [Package Configuration](#) as shown in fig. 1.

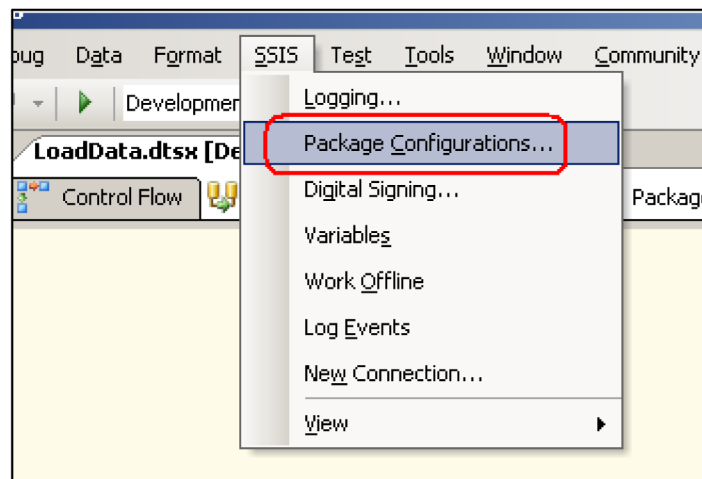


Figure 1: Adding Package Configuration

4. Click [Add](#) to add a configuration to the package and select the type of configuration required to use for the SSIS package as shown in fig.2.

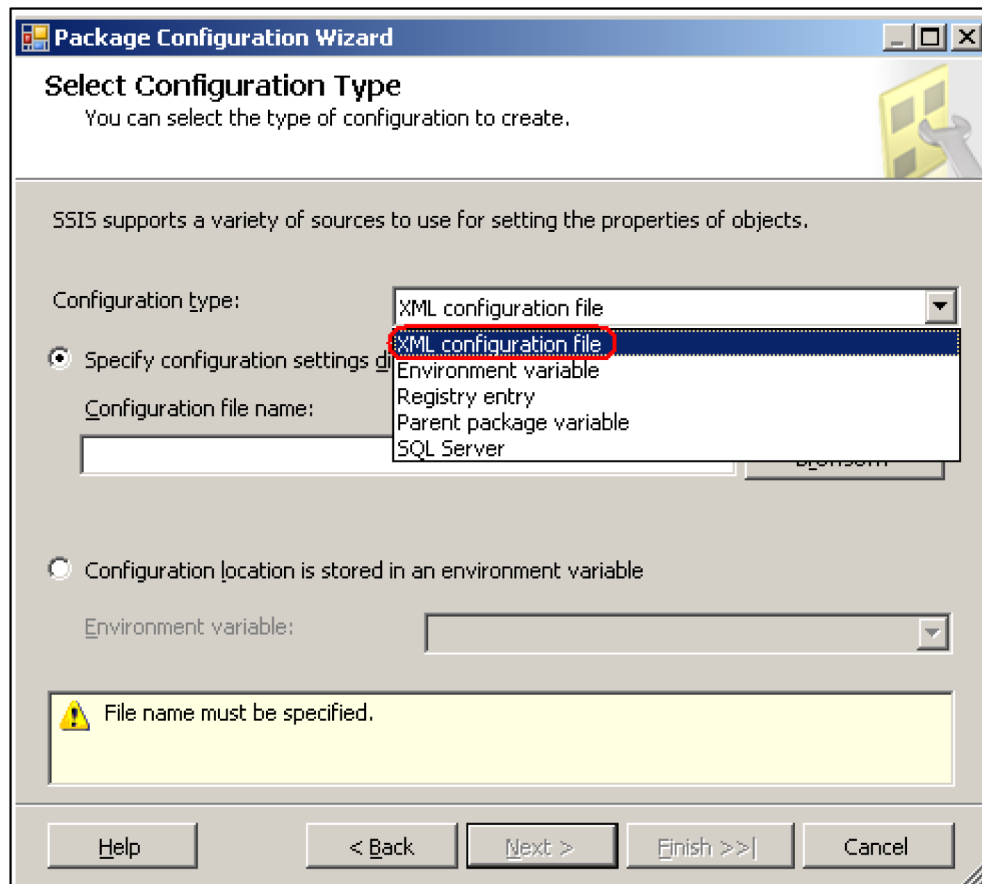


Figure 2: Selecting XML configuration file type

As shown in fig. 2, there are five ways to configure a package.

Before delving into the details of package configurations let's quickly see a small package driven by an XML Package Configuration File. Create an [SSIS](#) package, as shown in fig. 3, with the following tasks and components:

1. One data flow task.
2. One Flat File Source (located at C:\Source.txt).
3. An OLEDB Destination (database is TESTSSIS and table is MyTable) using connection manager DBCONN defined with NT authentication.

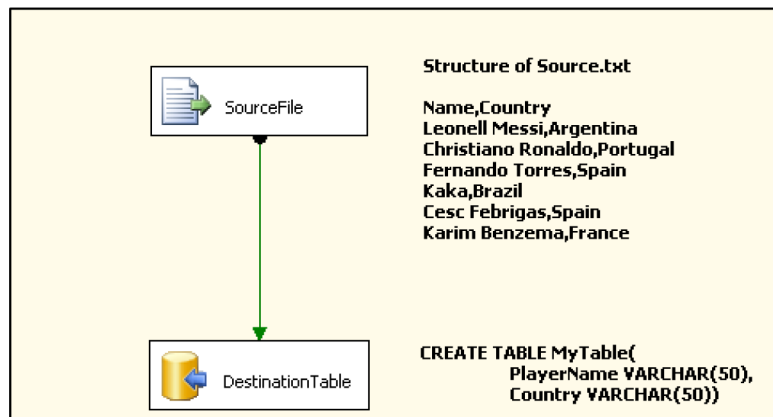


Figure 3 Data flow of the test package

This package is intended to get deployed on a machine where the target database is supposed to be [DestinationSSIS](#). In order to achieve this, follow the below mentioned steps to make the connection manager of the package configurable.

1. Enable the package configuration (fig. 2) by selecting the [Configuration Type](#) as [XML Configuration file](#).
2. Select the path where configuration file is to be created by supplying the path along with the filename or use [Browse](#) button to specify the path and name of the file. In this example C:\Sample is used as the target location and PKG_DTS.dtsconfig is the configuration file name as shown in fig. 4.

Note: The file name can also be given without using. However, from maintenance perspective it is the best practice to provide “.dtsconfig” extension.

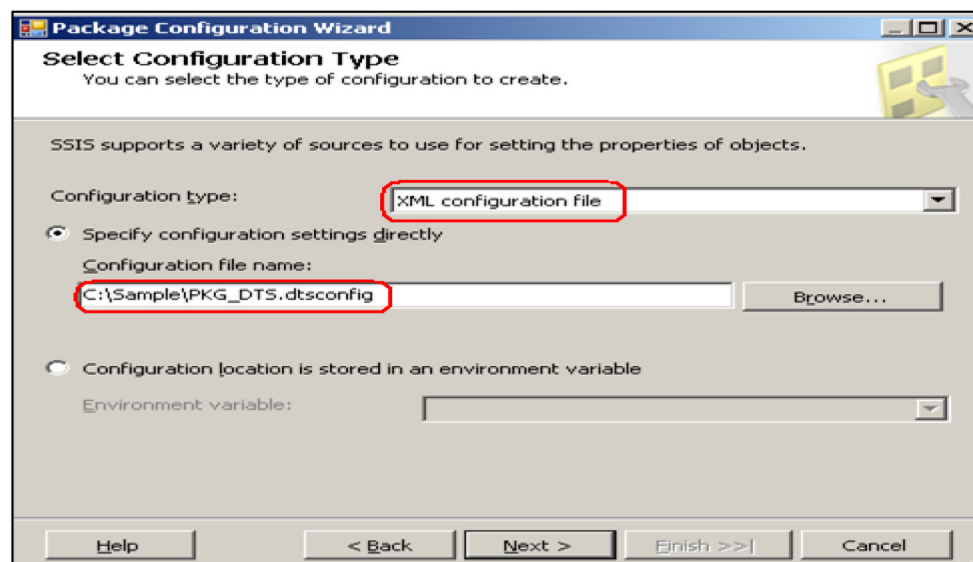


Figure 4: Configuration as XML configuration file- specify its path and name

- Click [Next](#) and select the package properties which should be made as configurable. In this case, [Database Connection String](#) should be made as configurable property as shown in fig. 5.

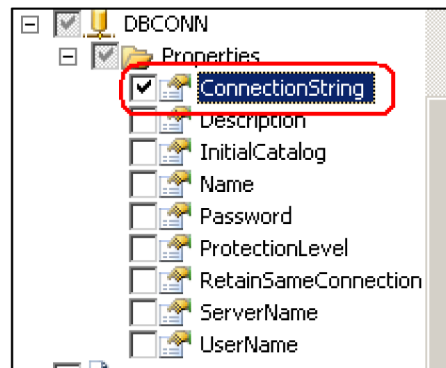


Figure 5: Selecting the configurable property

- Click [Next](#) and complete the wizard by clicking [Finish](#) button. A screenshot of the configuration file is shown in fig. 6.

```
</DTSConfigurationHeading>
  <Configuration ConfiguredType="Property" Path="\Package.Connections[DBCONN].
    Properties[ConnectionString]" ValueType="String">
    <ConfiguredValue>Data Source=ABC;
      Initial Catalog=TESTSSIS;
      Provider=SQLNCLI.1;
      Integrated Security=SSPI;
      Auto Translate=False;
    </ConfiguredValue>
  </Configuration>
</DTSConfiguration>
```

Figure 6: Original Configuration file

- Package configuration enables an SSIS package to read connection string from the configuration file for database connection manager. DBCONN is the connection manager name inside the SSIS package. In order to change the connection string inside the file it is required to change [Configured Value](#) part of the XML configuration file. For example, if the target server name is "XYZ" and database name is DESTINATIONSSIS then modify the configuration file as follows as shown in fig. 7.

```
<ConfiguredValue>DataSource = XYZ;
  Initial Catalog = DestinationSSIS;
  Provider = SQLNCLI.1;
  Integrated Security = SSPI;
  Auto Translate = False;
</ConfiguredValue>
```

Figure 7: New configuration file after edit

6. In case of SQL Authentication, there will be a new property added in the configuration file i.e., [User Id](#). While creating the connection manager using SQL Authentication, even password of that user also needs to be set. However, password is not captured by the package configuration wizard. It should be manually added inside the configuration file. After finishing the wizard, the configuration file looks as shown in fig. 8.

```
<ConfiguredValue>Data Source=ABC;  
    User ID= userid;  
    Initial Catalog=TESTSSIS;  
    Provider=SQLNCLI.1;  
    Auto Translate=False;  
</ConfiguredValue>
```

Figure 8: Original Configuration file

7. If the target server name is XYZ and database name is DESTINATIONSSIS, then the configuration file should be modified (note the manual addition of password) as shown in fig. 9.

```
<ConfiguredValue>Data Source=XYZ;  
    User ID= userid;  
    password=password;  
    Initial Catalog=DESTINATIONSSIS;  
    Provider=SQLNCLI.1;  
    Auto Translate=False;  
</ConfiguredValue>
```

Figure 9: New configuration file after edit

Since Password is sensitive information, there is a security concern while supplying the password to the configuration file. It can be avoided by using SQL Server configuration (discussed in the later sections in the paper) and protecting the sql server configuration table using role based security.

Note: When the package is deployed using deployment manifest file, the wizard will ask to select the installation folder for all the dependency files for the package (.dtsx and .dtsconfig). The default installation folder can be changed to a different location. By default the dependency files are stored in SSIS package store.

In case of SQL Server 2005 the default location is: <Drive>\Program Files\Microsoft SQL Server\90\DTS\Packages\<ProjectName>

For SQL Server 2008 the default location is: <Drive>\Program Files\Microsoft SQL Server\100\DTS\Packages\<Project Name>

Following section explains each of the configuration options in detail.

3. Types of Package Configurations

3.1 XML Configuration File

Selection of this option indicates to SSIS package that the configuration details are to be read from a XML file. There are two ways to indicate the path of configuration file location:

- **Direct Configuration**

Steps involved in configuring a **direct configuration** are:

1. In package configuration wizard, select the **Specify configuration settings directly** option button and give the path and name for the file as shown in fig. 10.

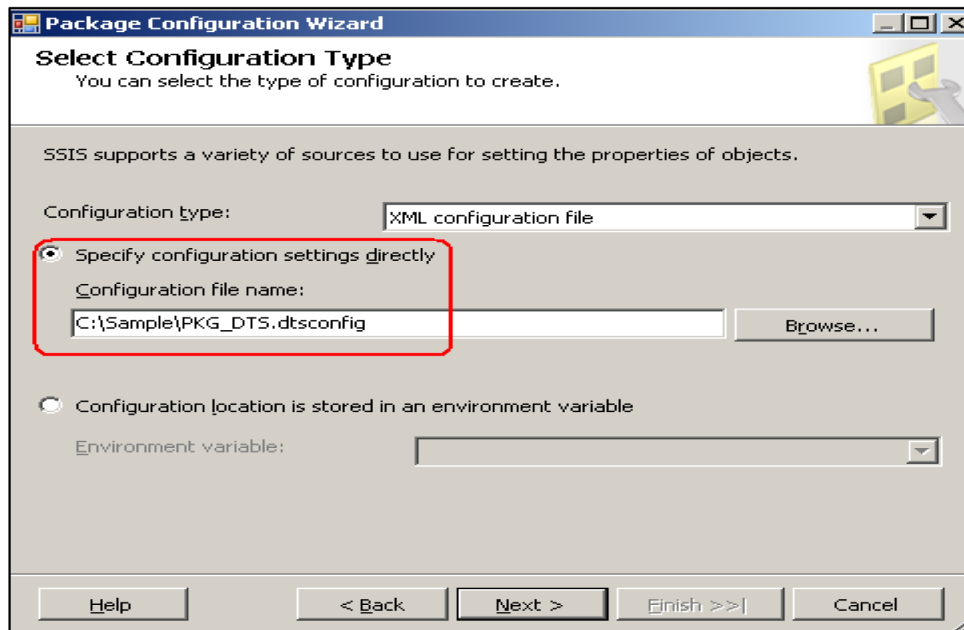


Figure 10: Selecting XML configuration file configuration type

2. Click **Next** and select all the properties that are configurable.

Wizard will script out those properties into an xml file and save it at the location mentioned along with the configuration file name. As we are explicitly defining the XML file inside the wizard it is known as Direct Configuration.

- **Indirect Configuration (using environment variable)**

Steps involved in configuring an **Indirect configuration** are as follows:

1. Create an environment variable and configure the path including file name.
2. Select **Configuration location is stored in an environment variable** from the option button as shown in fig. 11.
3. Select the environment variable from the drop-down option, which holds the location of the XML configuration file. For example, variable name is SSISDB and the value is C:\Sample\PKG_DTS.dtsconfig as shown in fig. 11.

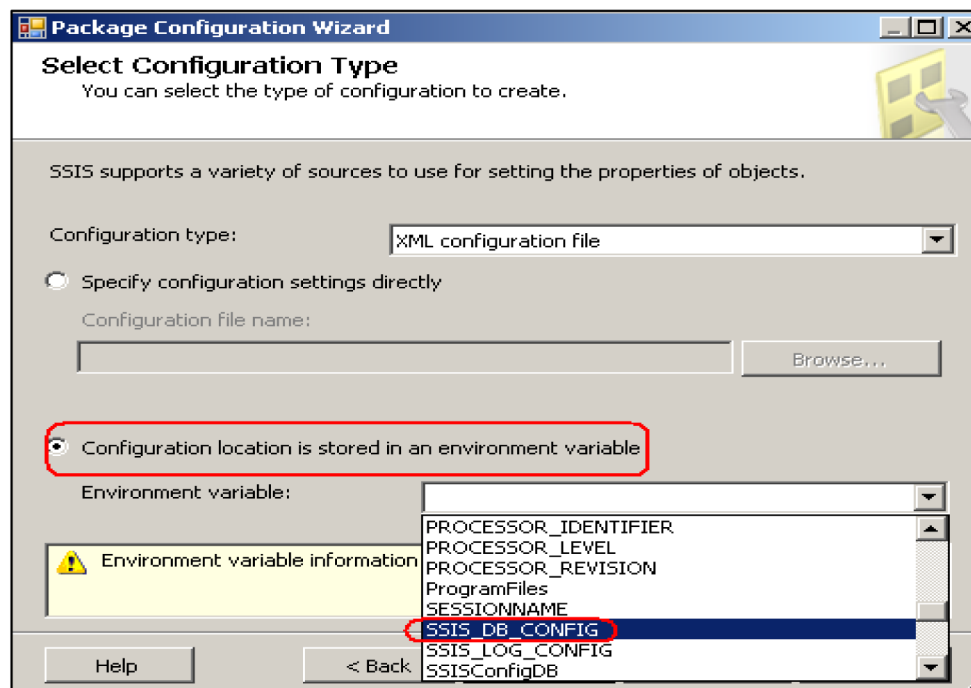


Figure 11: Selecting XML configuration file configuration type for indirect configuration

Comparison between Indirect and Direct configuration

Indirect Configuration	Direct Configuration
One configuration file for all the packages.	One configuration file per package.
Easy to handle and maintain single/one file.	Difficult to manage large number of configuration files.
The environment variable should be present on the target machine.	The configuration files could be copied on the target server.
Only the name of environment variable is saved inside the package and the path of configuration file is value of the environment variable. So, no need to explicitly add the configuration file while executing the package.	Design time configuration file path is saved inside the package which can be changed by explicitly adding the configuration file while executing the package on target machine.

Table 1: Indirect Vs Direct configuration

In case of XML configuration file (irrespective of direct or indirect configuration), one should be cautious when confidential information is supposed part of configuration content. For example, consider a package which connects to SQL Server using SQL Authentication and [User Id](#) and Password are included as configurable parameters in XML configuration file. There is no out of box mechanism to encrypt the xml configuration file to shield the Password. Hence it is not advisable to use XML configuration file in such scenarios.

3.2 Environment Variable

When this type of configuration is used in a package, the values of configurable properties are stored in the variables. In order to select the configuration type as **Environment Variable**, follow these steps:

1. Select **Environment variable** as configuration type as shown in fig. 12.

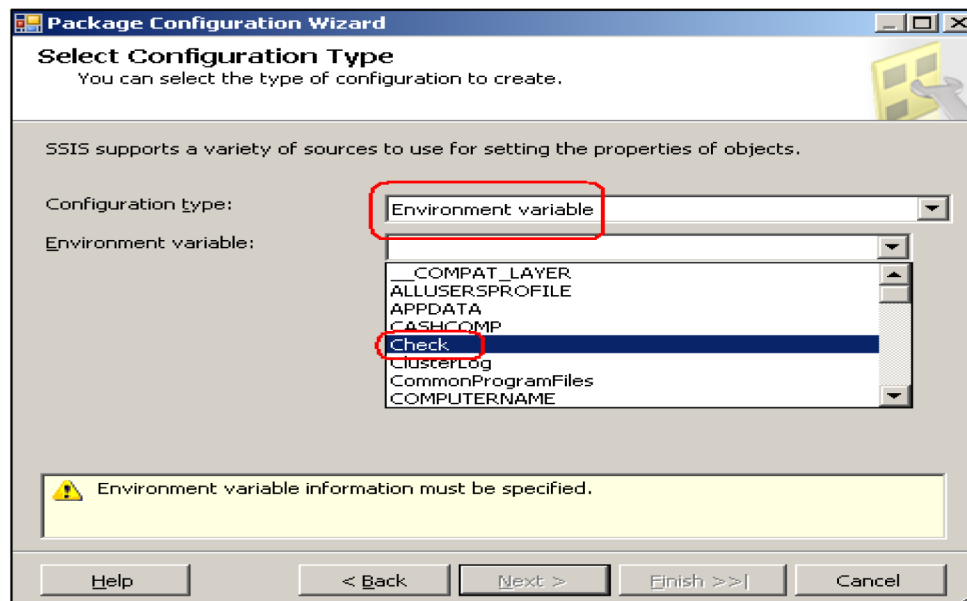


Figure 12: Selecting Environment variable configuration type

2. Select the **Environment variable** from the drop-down option (**Check** is the environment variable name as shown in fig. 12.).
3. Click **Next** and select the target property for this variable as shown in fig. 13.

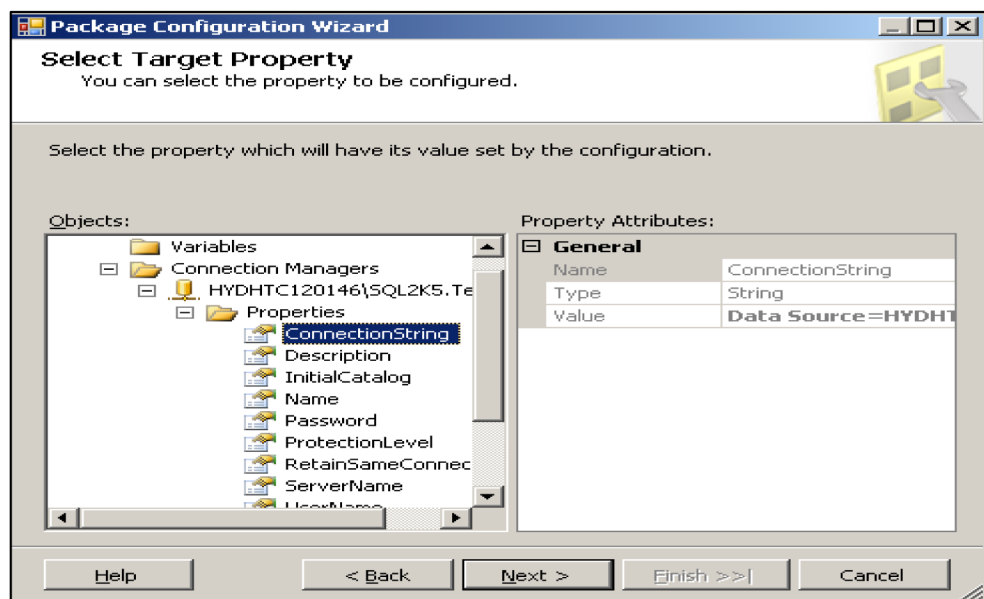


Figure 13: Selecting the configurable property

Now, the package will read the selected property from this variable. The environment variable needs to be created beforehand (creating environment variable is discussed later) as there is no [Create new environment variable](#) option available in this wizard.

If the database connection manager's connection string is selected as [Target property](#), then the value of environment variable needs to be changed if there is some change in the database connection string.

Steps to create environment variable:

1. Navigate to My Computer - Properties - Advanced tab.
2. Go to [Environmental variables](#) and create the variable (System variable). Ensure that same variable is available on the target server where package is to be deployed.
3. Value of the variable on the target server may or may not be same as the value on development machine.

Note: Sometimes the newly created environmental variables will not be seen inside BIDS. In order to see these new variables in the drop down box, BIDS needs to be restarted.

In this configuration type, each environment variable can store only one value. Hence If more than one property is to be made as configurable there should be one environment variable for each of the configurable property. There is no upper limit on the number of environment variables that can be associated with a package, but from maintenance perspective this type of configuration can be used when the configurable properties are not more than two. This mode of package configuration is not recommended to store user credentials as in case of database connection strings for SQL authentication because of security reasons.

Note: The upper limit of two is to avoid creating many environment variables. Also, if the number of configurable properties is more than two then as many package configuration needs to be added to the package which requires following the package configuration wizard screen repeatedly and is a tedious task.

3.3 SQL Server

This option is used when the configuration variables are to be stored in a SQL Server table. In order to use SQL Server configuration follow these steps:

1. Select configuration type as [SQL Server](#) and select the [Connection Manager](#) of your choice as shown in fig. 14. If the table is not available to store these values, the wizard allows creating a new table.

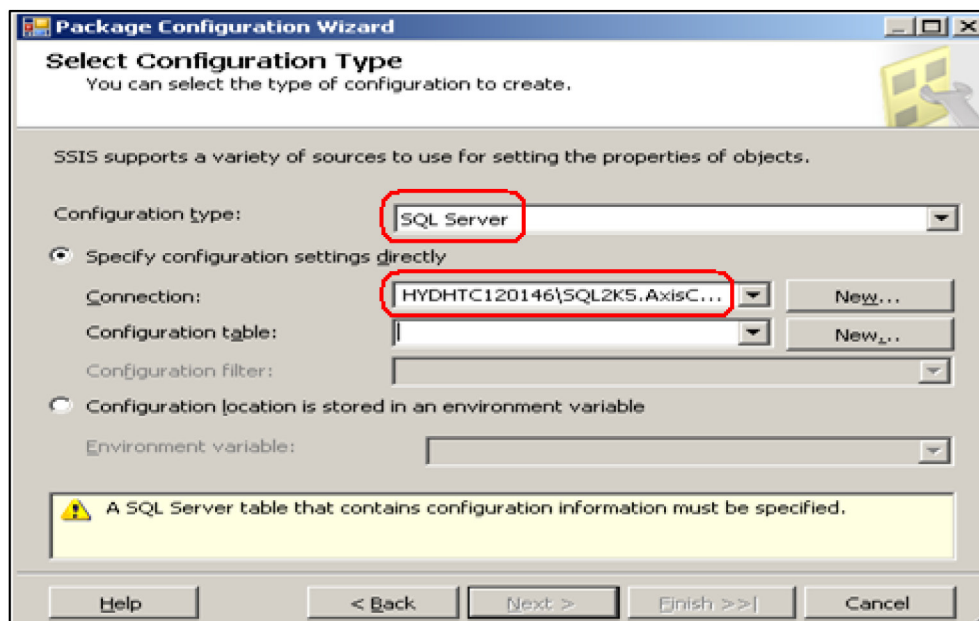


Figure 14: Selecting SQL Server configuration type

2. Click [New](#) to create the configuration table. Wizard will generate script for the configuration table as shown in fig. 15.

```
CREATE TABLE [dbo].[SSIS Configurations]
(
    ConfigurationFilter NVARCHAR(255) NOT NULL,
    ConfiguredValue NVARCHAR(255) NULL,
    PackagePath NVARCHAR(255) NOT NULL,
    ConfiguredValueType NVARCHAR(20) NOT NULL
)
```

Figure 15: Create table script

3. Click [OK](#) to create the table mentioned in the above script or copy the script and execute it inside SQL Server Management Studio (SSMS) upon editing (changing the table name, adding an identity column as Primary Key etc.).
4. Click [Next](#) to select the properties that should be made as configurable.

The number of rows created in the SSIS Configuration table will be equal to the number of configurable properties. For all the properties selected as configurable, the wizard will populate the table with appropriate column values. All the properties that have been selected as configurable are part of the configuration filter which is either selected from drop-down option or any newly created configuration filter.

The various columns of the configuration table can be understood by taking a sample configuration as shown in Table 2.

Package Path	Configuration Filter	Configured Value	Configured Value Type
ABC	Sample Filter	X	String
ABD	Sample Filter	Y	String
ABC	New Filter	Z	String

Table 2: Sample SQL server configuration table

SQL Server configuration enables designers to include multiple configuration paths for a single package. For example, while creating a package adding provision to include configurations that map to different environments like Development/Testing/UAT/Production. At any given point in time only one configuration can be active.

Following content explains the columns of the Table 2 in detail with their purposes:

- **Package Path**

Each property that is mentioned as configurable will have a path inside the package. Upon selecting [Connection Manager](#) DBCONN as configurable property, the package path value for that particular connection manager will be: \Package.Connections[DBCONN].Properties [ConnectionString]. Similarly each of the configurable properties has a package path which gets stored in the configuration table. In the sample table (ref. Table 2.), the package path is stored under the column name [Package Path](#).

• Configuration Filter

A configuration filter is a name which is given for each distinct set of configurations. Package path and configuration filter can be treated as Primary key for the configuration table. Hence there can be more than one value for a package path (configurable property) inside the configuration table but with a different configuration filter. One configuration filter is allowed per package configuration which means that a package can use only those values which belong to one configuration filter. For example, a [Configuration table](#) has 2 filters, Sample Filter and New Filter (as shown in the Table 2). For a package path ABC, only one filter can be used. If [Sample Filter](#) is used then the value assigned to the package path will be X, while using [New Filter](#) will assign Z to the same package path. Package path could be a connection string or a package variable. If the configuration table is empty then a filter cannot be selected using drop- down option and it has to be created manually as shown in fig. 16.

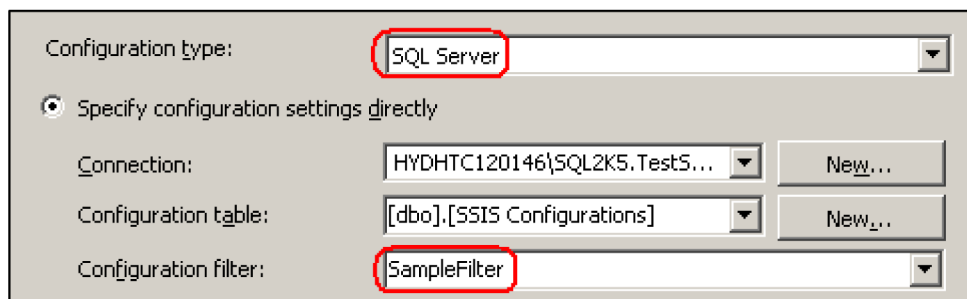


Figure 16: Creating the configuration filter

• Configured Value

The value in this column will be used by the SSIS package at runtime for the corresponding property. Update the configured value for a package path in the configuration table to bring in any change inside the connection manager by issuing an UPDATE statement against configuration table.

• Configured Value Type

It stores data type information of the configurable properties inside the package.

A package can be deployed using deployment manifest file (check the references section for deployment manifest file). Once the package using SQL Server configuration is deployed on a target server, same configuration table should be available on the target server.

- In case of SQL Server 2008, [Generate Script](#) option can be used to script out schema and data of the configuration table.
- Data script cannot be generated in SQL Server 2005. Data can be moved to other table by writing t-sql select query or by using bcp or extracting the data in a text file and load the configuration table on target machine using this text file.

In case of SQL Server configuration, the configuration table can be secured from unauthorized access by giving proper read and write permissions to a user. It is also easy to handle and maintain the table by issuing delete or update statements against it. To move the table from one machine to other the [Create script](#) has to be executed on target machine. Moving configuration table data from development machine to target machine depends upon the version of SQL Server being used as discussed above.

3.4 Parent Package Variable

If a package calls another package using execute package task, then calling package is referred as **Parent package** and called package is referred as **child package**. This option is used when parent package variable is to be passed to a child package.

For example, create two packages as mentioned below:

- **Package 1**

- Create an SSIS package and name it as **Parent**.
- Create a string variable in that package and name it as **Parent** with default value as SSIS
- Add an Execute package task inside the Parent package
- Configure this to use **Child** Package

- **Package 2**

- Create another package and name it as **Child**.
- Create a string variable **Child** with default value as “Child”.
- Add a script task inside the **Child** package
- Write a script to show the value of child package variable “Child”.

Execute Parent package which in turn will execute **Child** package and find the result as (child package variable value will be displayed) shown in fig. 17.

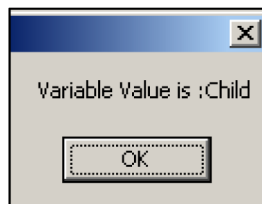


Figure 17: Value of the child package variable

Following steps enable passing a parent package variable to child package:

1. Add a configuration to the child package
2. Select its configuration type as **Parent Package Variable** as shown in fig. 18.
3. Give the name of the parent package variable as **Parent** as shown in fig. 18.

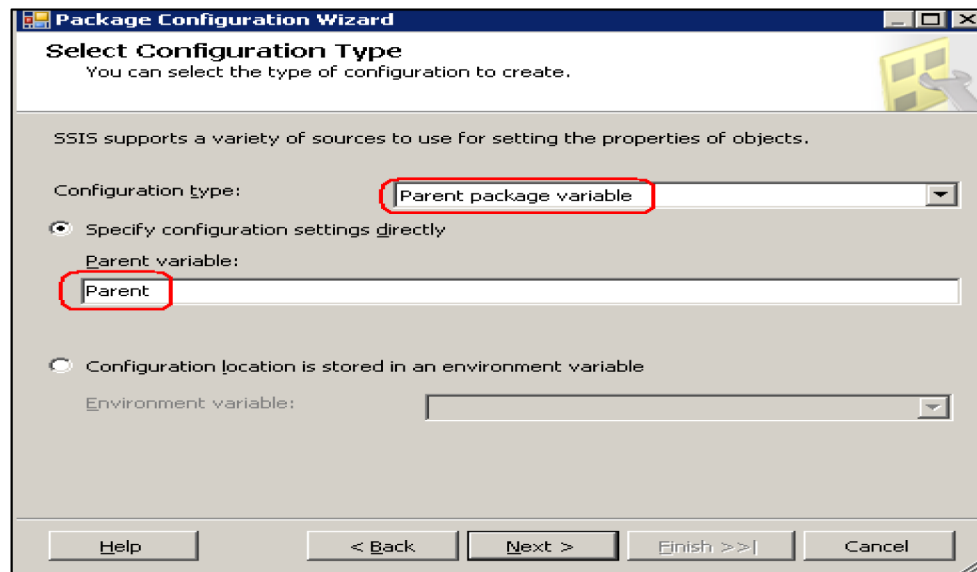


Figure 18: Selecting parent package variable configuration type

4. Click **Next** and select the **Child** package variable as shown in fig. 19 (by doing this, the Child package variable's value gets overwritten with the value of parent package variable).

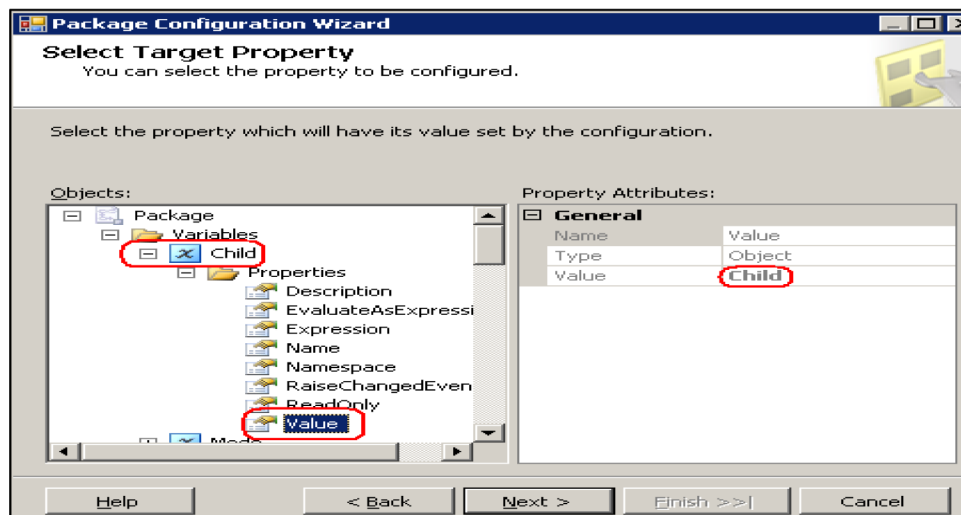


Figure 19: Select the configurable property

- Click [Next](#) and complete the wizard by clicking [Finish](#) button.

Now execute the Parent package and find the result as shown in fig. 20 (this time parent package variable value will be displayed).



Figure 20: Value of parent package variable

If more than one variable's value is to be passed from parent package to child package, then it is mandatory to have one [Parent package variable](#) configuration in the child package for each of the parent package variable. This configuration mode is recommended to use only when there is a requirement of passing a variables value from one package to other.

Note: Parent package variable is basically passing a parent package variable's value to a child package variable and it has nothing to do with the environment change. Parent package variables value can be made as configurable by using any of the package configuration modes.

3.5 Registry Entry

Windows registry is another mode to store package configuration values. The package configuration value can be stored either in an existing registry key or in a new registry key created. It can further be changed to different value of the key if required at a later point of time. Using the same package as example in Parent Package configuration, consider the variable Child. Use a registry entry to change the value of variable Child.

Steps to create [Registry key](#):

1. Create a new key in HKEY_CURRENT_USER section of registry. Go to registry editor and create a new key as shown in fig. 21.

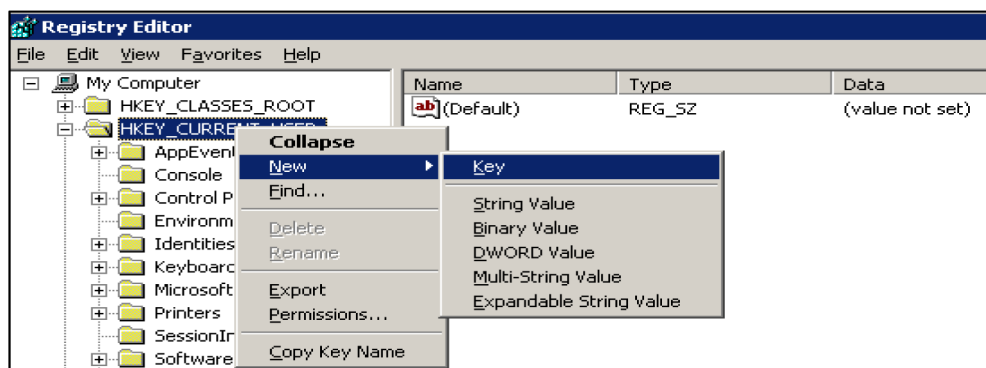


Figure 21: Creating a new key

2. Give the key name as [SSIS](#) and right click [SSIS](#) and select new string value as shown in fig. 22.

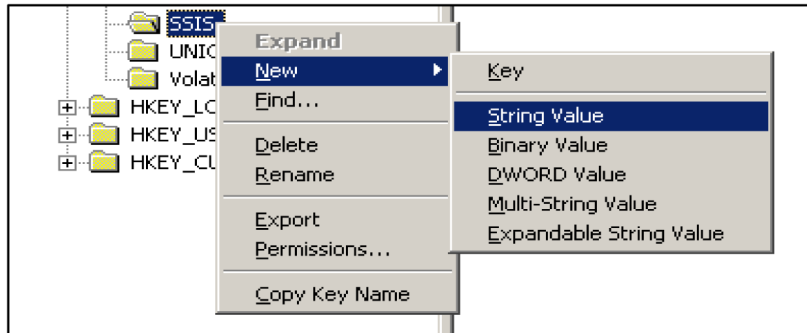


Figure 22: Creating a string value

3. Select name as [Value](#) (only this is a valid option). Double click [Value](#) and give string [Registry](#) as shown in fig. 23.

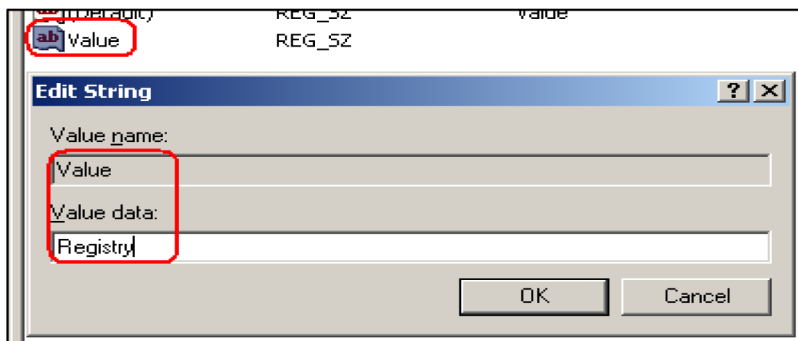


Figure 23: Assigning the value

Above three steps completes the process of creating a new key to store the value of a package configurable property. In order to use [Registry entry](#) configuration follow these steps:

1. Select [Registry entry](#) as [Configuration type](#) and enter the name of registry as shown in fig. 24.

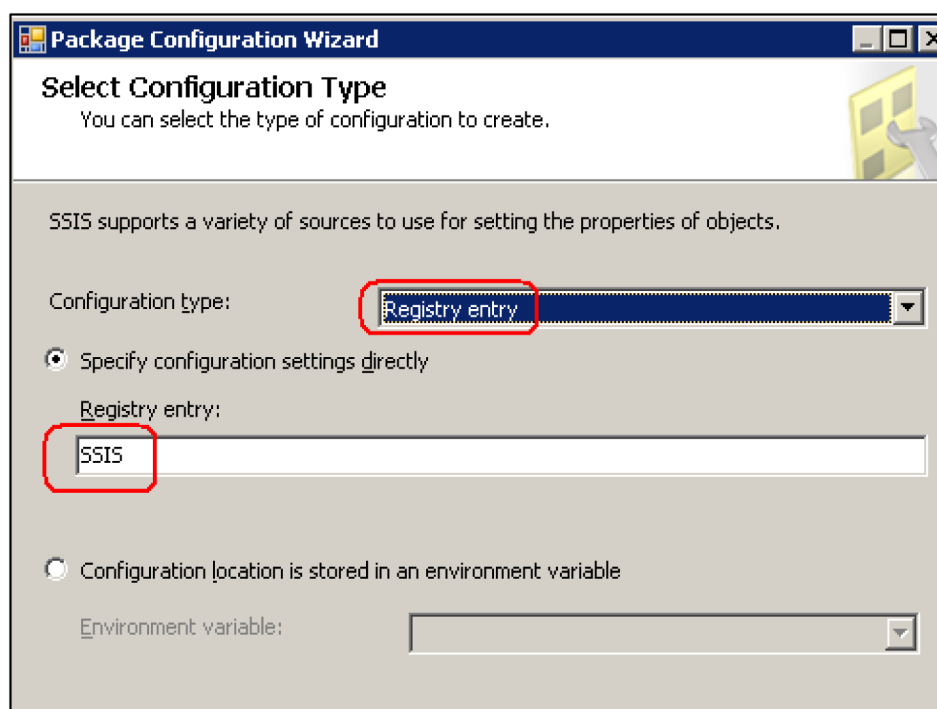


Figure 24: Selecting registry entry configuration type

2. Click [Next](#) and select the value property of variable [Child](#) and finish the wizard.

On executing the package, the value stored in the SSIS key will be displayed. This configuration mode is very rarely used. It should not be used to store database or user credentials if security is a concern. As far as maintenance is concerned, the same registry entry should exist on target machine in order to use this mode of package configuration.

4. Recommendations

Configuration Mode	Areas of concern		
	Security	Maintainability	Portability
XML	✗	✗	✓
SQL Server	✓	✓	✗
Environment variable	✗	✗	N/A
Registry entry	✗	✗	N/A

Table 3: Recommendations on various package configuration modes

Note: Portability is only relevant to XML configuration or SQL Server configuration in the above table. Parent package variable configuration is not included because it is used to pass variable's value from one package to other and is independent from changes in environment.

Configuration Mode	Configurable properties count	
	<=2	>2
XML	✓	✓
SQL Server	✓	✓
Environment variable	✓	✗
Registry entry	✓	✗

Table 4: Considering package configuration mode based upon number of configurable properties

Note: XML and SQL Server configuration can be used for any number of configurable properties.

5. Conclusion

This paper detailed on how to enable package configuration inside an SSIS package and how important it is whenever there is a change in the environment. Xml configuration and SQL Server configuration are two most widely used configuration types but as shown in Table 3 and Table 4, SQL Server configuration is best amongst all the available package configuration modes. It is recommended to use sql server configuration mode and try to avoid registry entry and environment variable configuration mode.

6. References

- Indirect Configurations
<http://blogs.conchango.com/jamiethomson/archive/2005/11/02/2342.aspx>
- Different Types of Package Configurations
<http://msdn.microsoft.com/en-us/library/ms141682.aspx>
- Setting up package configurations
<http://www.sqlis.com/post/Easy-Package-Configuration.aspx>
- Deploying a SSIS packages using deployment manifest file
<http://www.databasejournal.com/features/mssql/article.php/3600201/SQL-Server-2005-Integration-Services---Packages-Deployment---Part-23.htm>
- Package configuration using registry entry
http://www.sql-server-performance.com/articles/dba/package_configuration_2005_p2.aspx

About the Author

Nitesh Rai

He is a Technical Analyst with 4+ years of experience in SQL Server. He works for Microsoft Technology Center and his major responsibilities includes exploring new technologies and preparing artifacts, perusing external and internal forums and works on client projects.

Acknowledgement: The author would like to acknowledge contributions of

- Naveen Kumar (Principal Architect, MTC),
- Atul Gupta (Principal Architect, MTC),
- Virendra Wadekar (Senior Technical Architect, MTC) and
- Phaneendra Subnivis (Technical Architect, MTC)

for their timely support, guidance and providing critical and valuable inputs to achieve the current structure.



For more information, contact askus@infosys.com

About Infosys

Many of the world's most successful organizations rely on Infosys to deliver measurable business value. Infosys provides business consulting, technology, engineering and outsourcing services to help clients in over 30 countries build tomorrow's enterprise.

For more information about Infosys (NASDAQ:INFY), visit www.infosys.com.