MSSQLTips.com
brought to you by Edgewood Solutions

Your daily source for SQL Server Tips

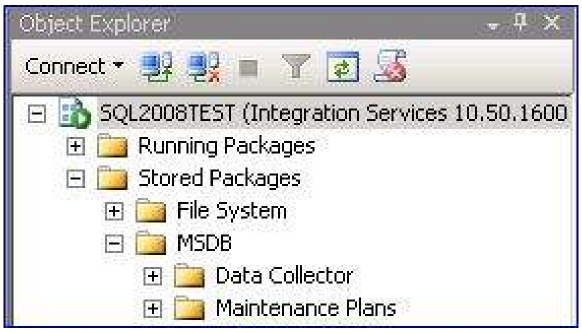
How to Manage SSIS Packages Stored in Multiple Database Instances

Written By: Ray Barley -- 7/13/2010

Problem
We have deployed SSIS packages to the default instance of SQL Server. We recently installed a named instance of SQL Server on the same server and would like to manage the SSIS packages stored in both the default and named instances of SQL Server from SQL Server Management Studio. However, after installing the named instance of SQL Server we no longer see the SSIS packages stored in the MSDB database of the default instance of SQL Server. How can we get this working correctly?

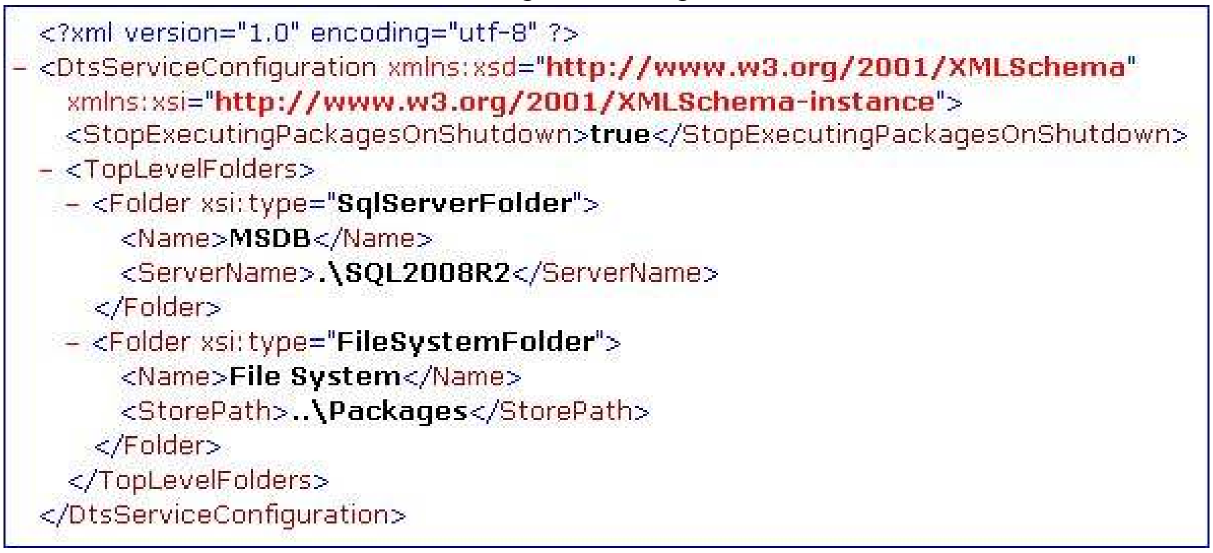
Solution
SQL Server Integration Services is not instance-aware; i.e. you can only install a single instance of the Integration Services service on your server. If you try to install Integration Services and it is already installed, you will get a warning message that continuing with the install will upgrade Integration Services.

When you connect to Integration Services using SQL Server Management Studio (SSMS) and expand the Stored Packages node, you will see the following:



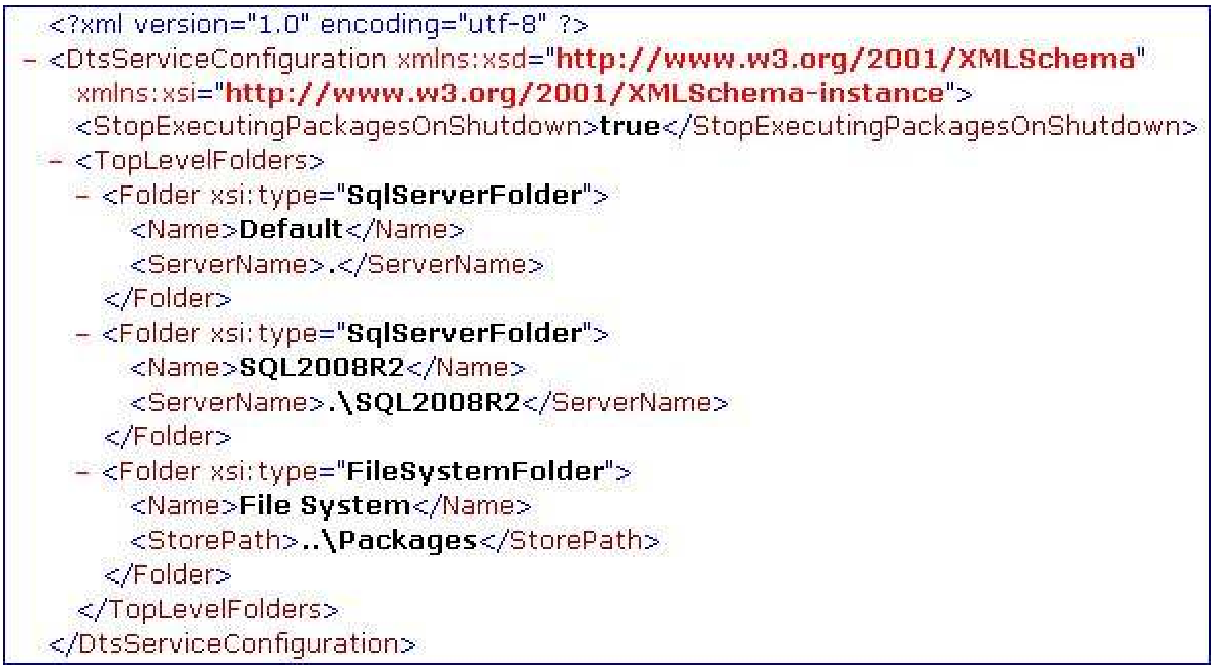
On my test system I have a default instance of SQL Server 2008 installed, then I added a named instance of SQL Server 2008 R2. The Data Collector and Maintenance Plans folders are created by the SSIS installation; none of the SSIS packages I deployed prior to installing SQL Server 2008 R2 are shown. What we would like to see is two MSDB nodes - one for SSIS packages stored in the MSDB database of the default SQL Server instance and a second one for SSIS packages stored in the MSDB database of the named SQL Server instance.

The answer to our problem lies in the MsDtsSrvr.ini.xml file which is stored in the folder %Program Files%\Microsoft SQL Server\100\DTS\Binn for SQL Server 2008. The contents on the file are shown below:



```
  <?xml version="1.0" encoding="utf-8" ?>
- <DtsServiceConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <StopExecutingPackagesOnShutdown>true</StopExecutingPackagesOnShutdown>
  - <TopLevelFolders>
    - <Folder xsi:type="SqlServerFolder">
        <Name>MSDB</Name>
        <ServerName>.\SQL2008R2</ServerName>
      </Folder>
    - <Folder xsi:type="FileSystemFolder">
        <Name>File System</Name>
        <StorePath>..\Packages</StorePath>
      </Folder>
    </TopLevelFolders>
  </DtsServiceConfiguration>
```

Note the folder with the type SqlServerFolder. The ServerName node value is .\SQL2008R2. This is my SQL Server 2008 R2 named instance that I installed. There is no reference to my SQL Server default instance. To fix this you simply add another folder with a type of SqlServerFolder and specify the ServerName of the default instance of SQL Server 2008 (just a period for the default instance). The revised MsDtsSrvr.ini.xml file is shown below:



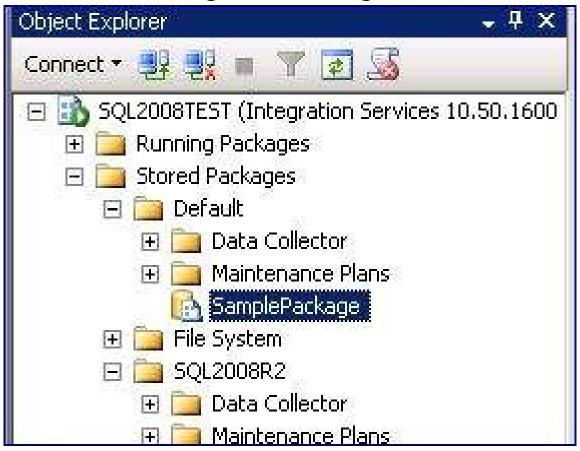
```
  <?xml version="1.0" encoding="utf-8" ?>
- <DtsServiceConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <StopExecutingPackagesOnShutdown>true</StopExecutingPackagesOnShutdown>
  - <TopLevelFolders>
    - <Folder xsi:type="SqlServerFolder">
        <Name>Default</Name>
        <ServerName>.</ServerName>
      </Folder>
    - <Folder xsi:type="SqlServerFolder">
        <Name>SQL2008R2</Name>
        <ServerName>.\SQL2008R2</ServerName>
      </Folder>
    - <Folder xsi:type="FileSystemFolder">
        <Name>File System</Name>
        <StorePath>..\Packages</StorePath>
      </Folder>
    </TopLevelFolders>
  </DtsServiceConfiguration>
```

I also changed the Name node from MSDB to Default and SQL2008R2 so I can easily identify the database instance of the MSDB database. Without changing the Name node both would be MSDB.

One important note - after you change the MsDtsSrvr.ini.xml file, you have to restart the Integration Services service and also refresh the SSMS Object Explorer in order to see your changes. You can restart the Integration Services service by using SQL Server Configuration Manager or the following PowerShell command:

```
restart-service -name MsDtsServer100
```

After restarting the Integration Services service and refreshing the SSMS Object Explorer, you can now see both MSDB instances (the SamplePackage deployed to the MSDB database in the default SQL Server instance is now shown):

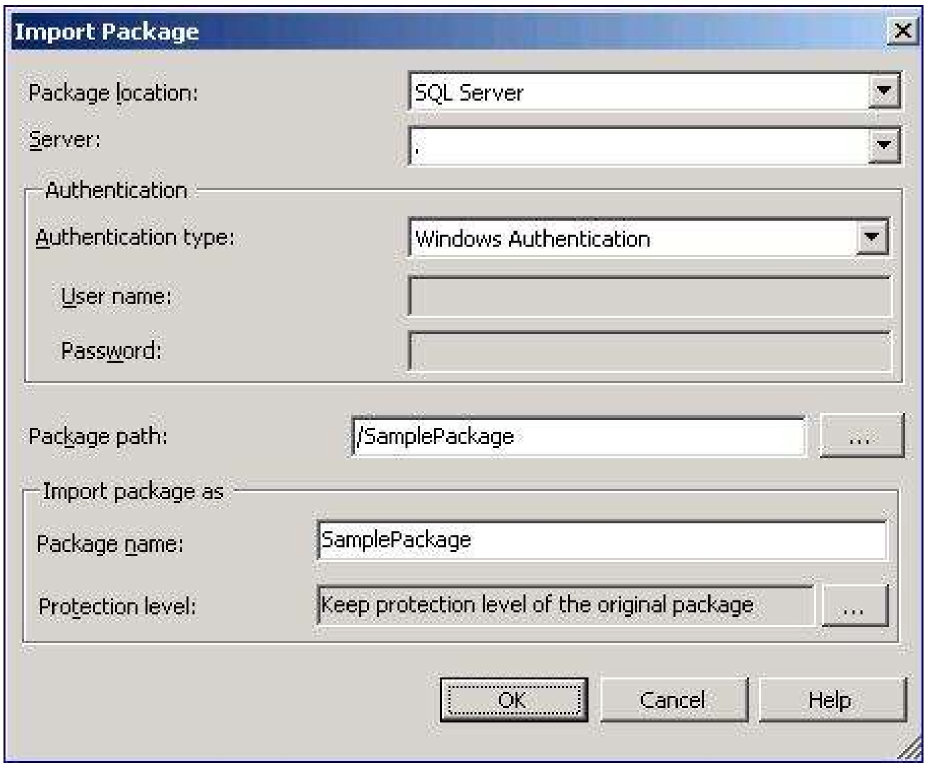

Object Explorer

Connect

- SQL2008TEST (Integration Services 10.50.1600
  - Running Packages
  - Stored Packages
    - Default
      - Data Collector
      - Maintenance Plans
      - SamplePackage
    - File System
    - SQL2008R2
      - Data Collector
      - Maintenance Plans

Copying or Moving Packages

You can use SSMS to import an SSIS package from SQL Server, the file system or the package store. To import the SamplePackage from the MSDB database in the default instance of SQL Server to the SQL2008R2 instance, right click on SQL2008R2 in the Object Explorer and select Import Package; fill in the dialog as shown below:


Import Package

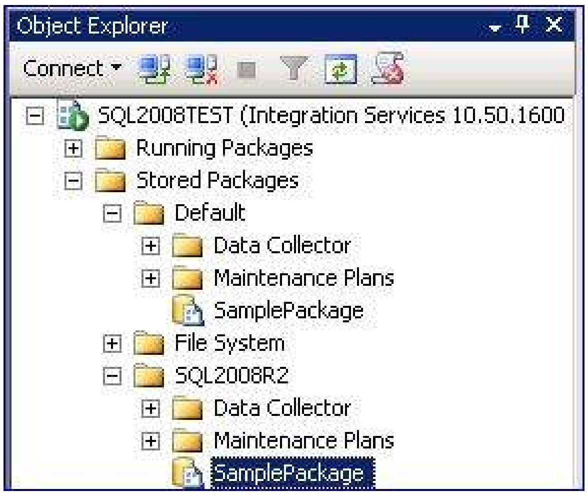
Package location: SQL Server
Server: .

Authentication
Authentication type: Windows Authentication
User name:
Password:

Package path: /SamplePackage

Import package as
Package name: SamplePackage
Protection level: Keep protection level of the original package

You will now see the SamplePackage deployed to the MSDB database of the SQL2008R2 instance of SQL Server as shown below:


Object Explorer

Connect

- SQL2008TEST (Integration Services 10.50.1600
  - Running Packages
  - Stored Packages
    - Default
      - Data Collector
      - Maintenance Plans
      - SamplePackage
    - File System
    - SQL2008R2
      - Data Collector
      - Maintenance Plans
      - SamplePackage

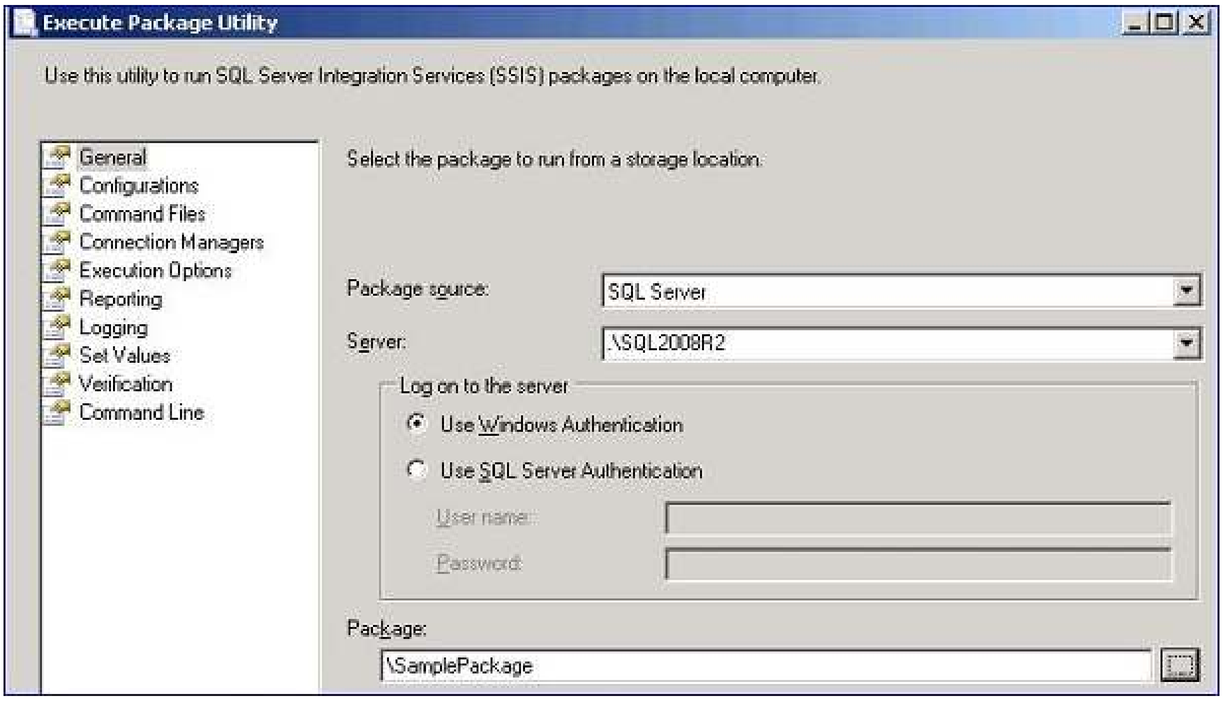
You may find that the command line utility DTUTIL.EXE is a more efficient way of copying or moving your SSIS packages. Take a look at our earlier tip Import, Export, Copy and Delete SQL Server Integration Services Packages SSIS for some examples of using DTUTIL.EXE.

Executing SSIS Packages

There are several ways to execute an SSIS package:

- SQL Server Business Intelligence Development Studio (BIDS)
- DTEXEC.EXE command line utility
- DTEXECUI.EXE graphical user interface
- SQL Server Agent

Keep in mind that when you are executing packages deployed to SQL Server, you are going to have to specify the server where the package is deployed. For example if you are using DTEXECUI.EXE you specify the Server then you can click on the button next to the package name to select the package to run from the list of deployed packages on that server as shown below:



Use this utility to run SQL Server Integration Services (SSIS) packages on the local computer.

Select the package to run from a storage location.

Package source: SQL Server

Server: .\SQL2008R2

Log on to the server

(●) Use Windows Authentication

( ) Use SQL Server Authentication

User name:

Password:

Package: \SamplePackage

Take a look at our earlier tip Different ways to execute a SQL Server SSIS package for the details on the various ways to execute an SSIS package.

Next Steps

- Keep in mind that when you use DTEXEC to execute SSIS packages deployed to a named instance of SQL Server, you need to specify the /Server option with the server name and instance name. Without the /Server option DTEXEC will try to load the SSIS package from the default instance on the local server.
- Use DTEXECUI to get the DTEXEC command line to run your SSIS packages. After running your package successfully, you can click on Command Line to get the parameters that you need to append to DTEXEC.
- The above tip used SQL Server 2008 and SQL Server 2008 R2 as examples. The same applies to SQL Server 2005 except the MsDtsSrvr.ini.xml file is in the folder %Program Files%\Microsoft SQL Server\90\DTS\Binn and the name of the Integration Services service is MsDtsServer.
- Take a look at Managing Integration Services in Books on Line for additional details on the Integration Services service.