Managing and Configuring SSIS Packages

SQL Server Technical Article

Writers: Brian Knight, Devin Knight, Mike Davis

Technical Reviewer: Dustin Ryan

Published: 6/1/2009

Updated: 6/1/2009

Summary: This white paper covers several interesting and unique methods for managing, auditing and configuring SSIS Packages.

Copyright

Table of Contents

Table of Contents

Introduction

Administering SSIS can be a challenging task for a DBA who's not familiar with the SSIS environment and some of the new hurdles it presents. In this whitepaper, we discuss how to configure your packages so you have zero-touch deployments. We dive deep into a notification and logging framework to alert you when a problem occurs or when you run out of capacity. We also discuss how to deploy and schedule your SSIS packages.

SSIS Configuration Options

One of the main challenges you'll have as an SSIS developer is configuration of the packages. You will want to externalize the key elements in the package like connections and variables so you can seamlessly migrate the packages from development to production without having to open up each individual package. This is where SSIS package configurations help. Package configurations enable a developer or an administrator to have one key area where specified settings are stored. The administrator can then change the connection information one time and any referring packages will change as well.

Before starting a discussion on configurations, it's important to know the execution order of a package. When a package starts, it goes through many phases:

1. Package starts
2. Package is validated
3. Input variables and configurations are accepted from dtexec.exe from the /SET command
4. Package configurations are applied
5. Expressions are applied

You have two primary options for where to store your SSIS configurations: configuration files or tables (there are other more lesser-used options). The advantage of using a configuration file is the files are very portable from environment to environment but the disadvantage is that there's not a way to centrally manage them. For configuration tables, the entries are stored in SQL Server, making them easy to report against centrally but they are not very portable.

You can access package configuration through the Control Flow tab in BIDS by right-clicking in the design pane and selecting Package Configurations. This opens the Package Configurations Organizer dialog box. You can then add multiple configurations that will overlay each other top to bottom in this screen by clicking the Add button. You can also temporarily disable configurations, which would revert the package back to its hard-coded values.

Configuration Files

Configuration files are one of the easiest configuration devices in SSIS since they require no SQL Server backend and are very portable. They resemble web.config files in the ASP.net environment. Once you're at the Package Configurations Organizer dialog box, click Add to add a new configuration. This opens the Package Configuration Wizard, where you can add many types of configurations. Some of those types of configurations are XML files, registry entries, SQL Server tables, environment variables or parent package variables.
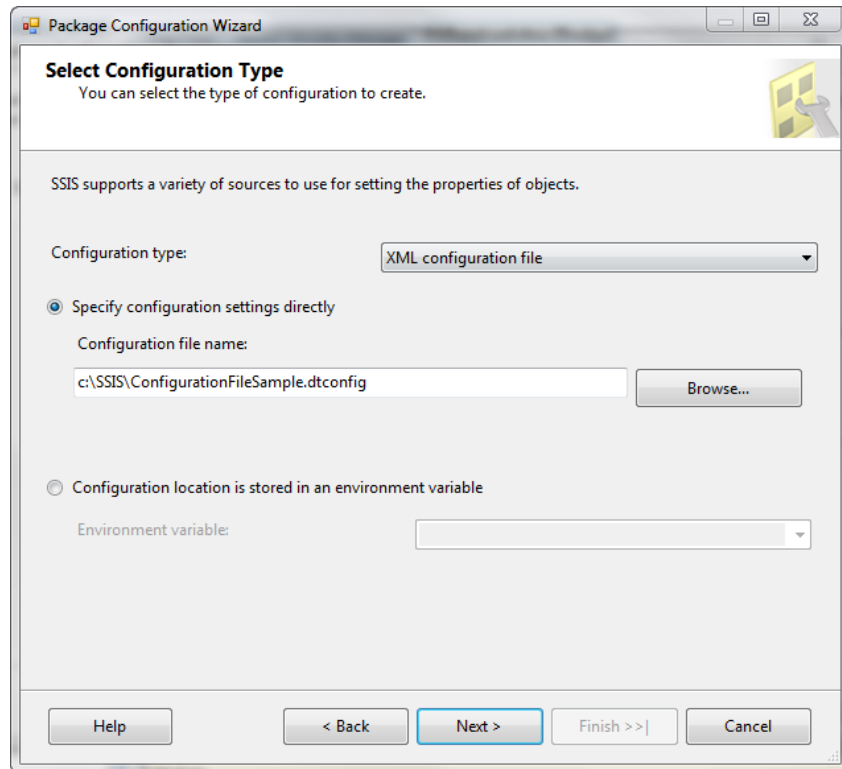
If you choose "XML configuration file" , type the path of the file that you wish to be created or the location of an existing configuration file then click Next. If the file already exists, you'll be asked if you wish to reuse the same configuration file over again. This would enable you to have a single configuration file that contains the connection information for a hundred packages.
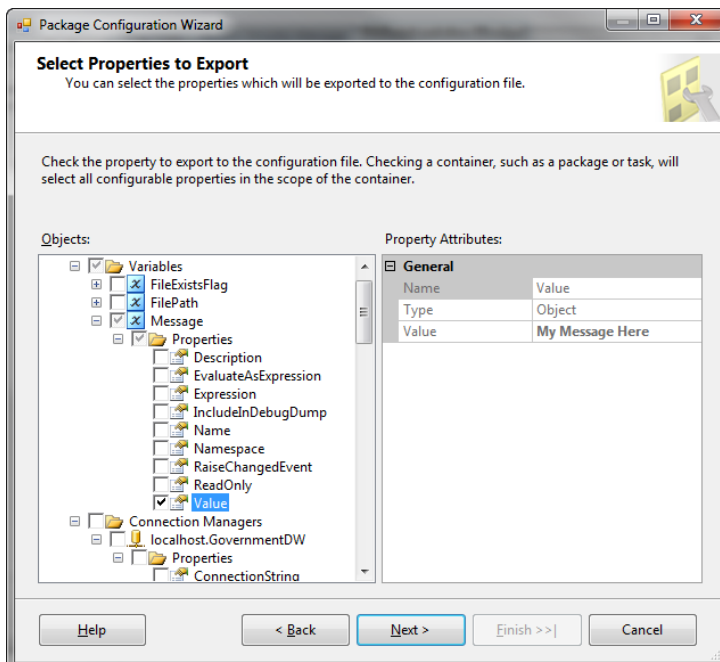
If the file does not exist, you'll be taken to the next screen where you can check the list of properties that you want to export to the file the "Select Properties to Export" screen. You simply must drill into each property of a package, task, connection or variable and check the individual items you wish to have in the file. If you want to reuse the file for multiple packages, the object's name must exactly match in each of the packages. For example, if you want to

externalize the ServerName property in the configuration file, you need to make sure the connection manager is named the same name across all the packages.

Whether you choose to use configuration files or tables, the entry is going to be in clear text and if security is a concern, you must protect your file or table. By combining a Script Task with the file or table, you could encrypt the properties but this is a customized solution.

SQL Server Configuration Tables

Configuration tables operate much like configuration files but rely on SQL Server to store you settings. You create a configuration table in the same wizard that controls other configurations. In that wizard, you can create a table in any data source that you wish that has a connection manager. Once it's created, place a filter on it that describes what type of entries these represent like "Staging Connection Info". When the package begins, it will select against that table and apply a where condition on the select statement, retrieving all properties that have that filter on associated to them.

Other Configuration Recommendations

If you have a configuration file with ten connections in it and your package only needs a single connection, there is a chance of a warning or error. We recommend that you create a configuration file or filter for each data source. This will enable you to pick the configuration file that contains your connection and leverage it over and over again without fear of a warning due to missing connections.

SSIS Logging Standards

SSIS logging helps you keep track of the packages running on your servers. The logging options in SSIS allow you to record at run time package information such as execution time and errors. You can log to a number of key areas like SQL Server tables, the Windows Event Log, text files or XML files to name just a few.

Native Logging

To use the built-in logging options of SSIS, right click in the design pane of control flow of a package and select logging. The drop-down menu of logging options contains:

- **Windows Event Log**-The Windows Event Log option logs the data to the Windows Event Log. This is perfect if you want to integrate with 3rd party monitoring solutions like Tivoli or MOM.
- **Text File**-The Text File option saves the information into a plain text file with the data comma separated.
- **XML File**-The XML File option saves the data in an XML File parsed into XML nodes.

- **SQL Server**-The SQL Server option logs the data to a table in your SQL Server. The table logged to on the server is named SYSssisLog." If it does not exist, this table is created automatically by SSIS when the package runs.
- **SQL Server Profiler**-The SQL Server Profiler option logs the data to a file as SQL that can be captured in SQL Server Profiler.

Once you select the provider type you will need to place a check next to the package name in the left pane. Then you can then select the details of what information you would like to log. Under the Details tab you select the events you want to cause logging. If the event occurs during the package run time it will log the data to the selected provider. No matter which provider you select you will need to select a connection to use in logging. The drop down menu show compatible providers. If you do not see a provider then you can click <New Connection…> to create the proper connection.
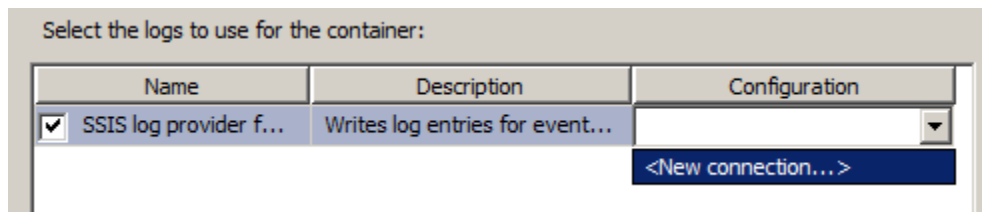
In the details tab you can select the details of what you would like to log. This can be customized for each event.
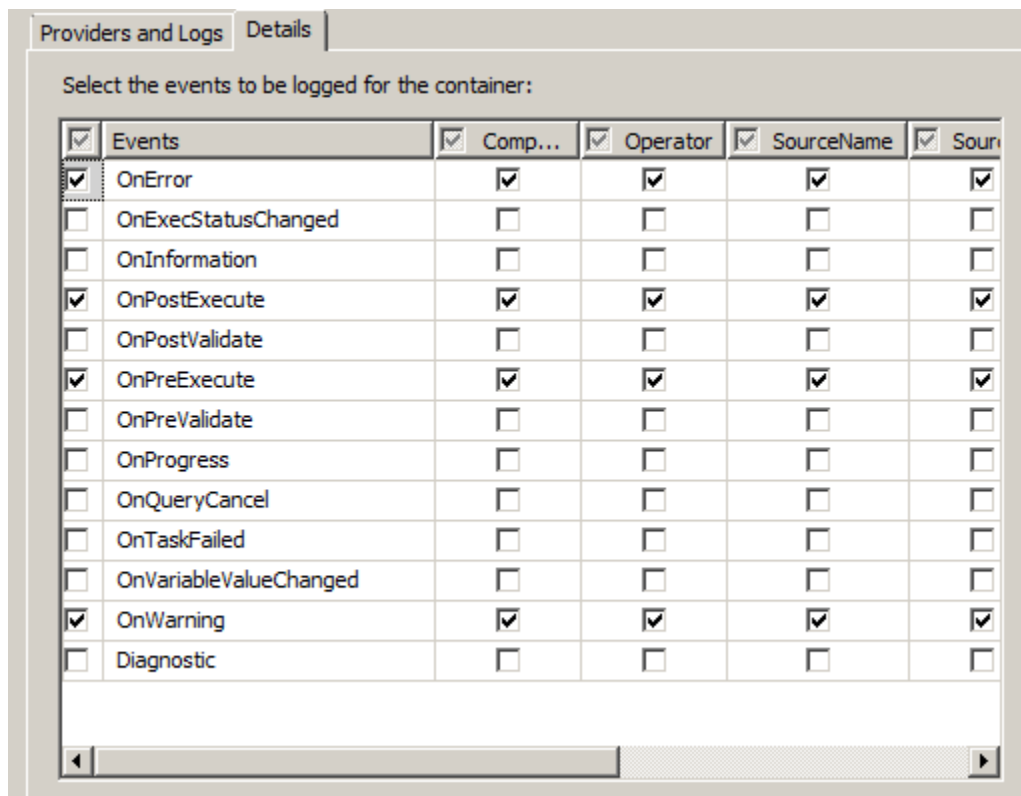
Under the Details tab is where you select the event to log. If the event occurs during the package runtime it logs the data to the selected provider. OnError, onWarning, onPreExecute, and onPostExecute are some commonly used events.
OnError fires if an error occurs during the execution of the package. OnWarning fires if a warning occurs during the package execution. The onPreExecute event logs all the data before the package begins execution. The onPostExecute event option logs all of the data after the package has completed execution.

The data that is saved by the SSIS logging options is shown in the advanced window, which you bring up in the Details tab by clicking the advanced button at the bottom of the screen. In the advanced screen you select data to log and on what event. You can select or unselect each item on each event. You'll see some key columns that would be nice to have are not there like the package's name that generated the event. Instead, this is under the SourceName column sometimes and a self-join with the ExecutionID can derive the package's name.

To create a log provider, select a provider type from the drop-down menu on the Providers and Logs tab. Then click the Add button. The provider will appear in the "Select the logs to use for the container" pane below the menu. Put a check in the Name column to engage the log provider. To disable a log but not remove it, only remove the check in the Name column next to that undesired log provider.

After the log provider is created and the name column is checked, click on the Configuration drop-down menu. If a proper connection type exists on the package it will show in this window. If not, create it by clicking the < New connection … > option.

In the Details tab choose the events you need to log. Place a check next to the events you need to log. Once you have selected the events click the advanced button and uncheck any data you do not want to log. The current configuration can be saved to an XML file with the Save button at the bottom of the Details tab. This XML file can be used in another package to select the same check boxes in this logging configuration screen automatically by clicking the Load button that also appears at the bottom of the Details tab.
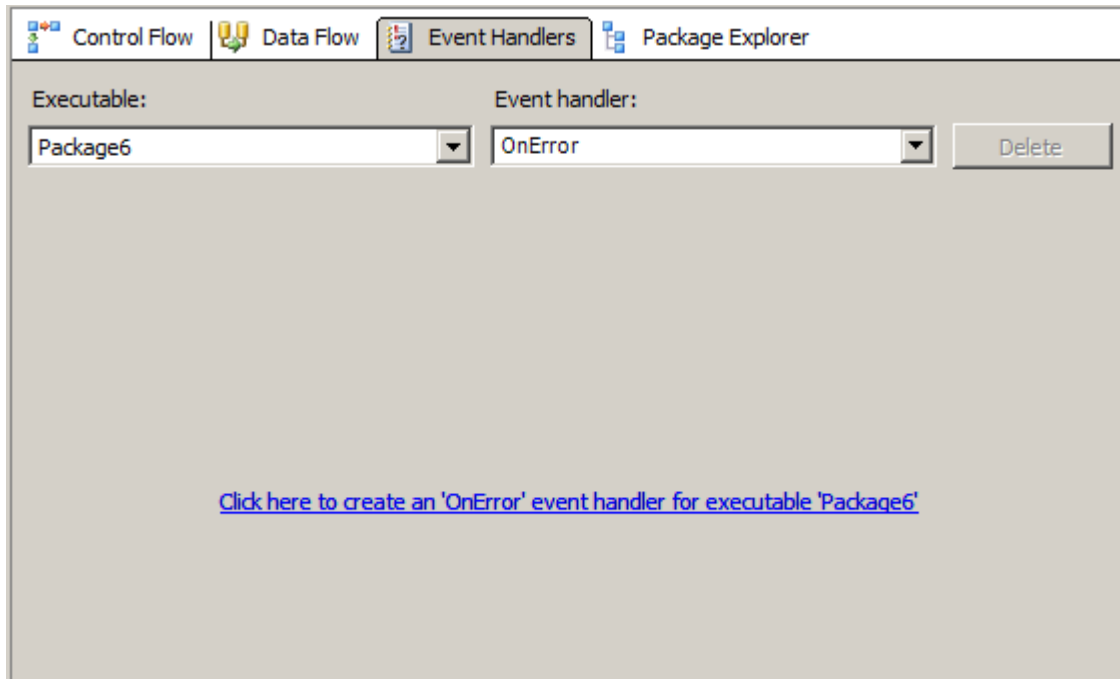
Event Handlers

Event handlers call tasks when events like errors or warnings occur on a package during execution, this can be used in logging errors or sending notifications to users. The event handler's tasks only run if that event is called. Here are the events that can occur in a package to cause an event to fire and call an event handler:
- OnError
- OnExecStatusChanged
- OnInformation
- OnPostExecute
- OnPostValidate
- OnPreExecute
- OnPreValidate
- OnProgress
- OnQueryCancel
- OnTaskFailed

- OnVariableValueChanged
- OnWarning

Some of the most frequently used events are OnError, OnWarning, OnPreExecute, and OnPostExecute. The OnError event fires when an error occurs on the package. The OnWarning event fires when a warning occurs on a package. The OnPreExecute event fires just before the executable or the package start. The OnPostExecute event fires just after the selected executable or the package finishes executing.

In the Event Handlers there is a blue link in the middle used to create an Event Handler. When clicked, this link creates an event handler on the package. Two drop-down menus exist at the top of the Event Handlers Tab. The drop-down menu on the left contains a list of all executables in the package. This list contains all the tasks in the package. If no tasks exist in the package, the only executable will be the package.

When creating an event handler, it is important to select the executable from the drop-down menu to ensure the tasks in the event handler execute when needed. The drop-down menu on the top right, contains a list of all the events that can be chosen for the selected executable.

Clicking the blue link in the middle of the tab creates an event handler for the package. This causes the tasks in the event handler to execute if the event occurs during any tasks in the package. If you want the tasks in the event handlers to fire only for a certain task in the package, select the task in the left drop-down menu and then click the blue link in the event handler. This creates an event handler for the explicit task and executes only when the selected event occurs.

Some of the common uses for the event handlers are notification and logging. When you need to be notified with an email about a specific event that occurred in a package, the event handlers are the proper place to execute this task.

To be notified via email that a package has completed, drag in and configure a Send Mail Task in the OnPostExecute Event Handler for the package. Send Mail Tasks can contain information about the package. You can include any system variables in the message to tell you what occurred in the package and when including the Error Description.

You can create a custom logging framework if you are not content with the native SSIS logging. Execute SQL Tasks in the event handlers can write information to a database. These Execute SQL Tasks execute only when the event occurs. This logs errors when they occur with the OnError Event Handler and warnings with the OnWarning Event Handler.

It can be hard to keep track of multiple event handlers on many different executables in a package. The Executable drop-down menu in the event handler tab shows all of the event handlers on the package and each executable. There is an Event Handler folder under the package and each executable in the package. The plus next to this folder opens the folder and shows the event handlers in them. In the event handlers is an Executables folder showing each task in the event handler. The same tasks can be seen in the Package Explorer tab.

Logging in the Data Flow

In some Data Flows of a package there is a need to log information. One of the common items to log is a row count. To log row counts in a Data Flow you will need to create a Row Count transform in the Data Flow. The Row Count will need to be in the flow with the rest of the tasks in the Data Flow. A package variable will need to be created to hold the numeric information from the Row Count transform. An example of a good variable naming convention would be intRowCount. If you have multiple row counts then multiple variables will be needed. If you want to log the extracted rows and the loaded rows then you will need a Row Count transform immediately after the source and one more immediately before the destination. Each will need to write to a different variable.

After the Row Count transforms and variables have been created you can create an Execute SQL task in the OnPostExecute Event handler of the package with an Insert statement to log the information to a SQL table. The row count variables would be passed in as parameters to the SQL command.

Logging in BI xPress

BI xPress from Pragmatic Works (http://www.pragmaticworks.com) is an enterprise ready tool that will automatically add all needed row counts to each data flow and event handlers in a package. It will also apply this same framework across all packages in your environment. This can save hours or days of work and gives you a consistent and robust Auditing Framework for all of your packages that will help any company pass strict regulations like Sarbanes-Oxley (SOX).
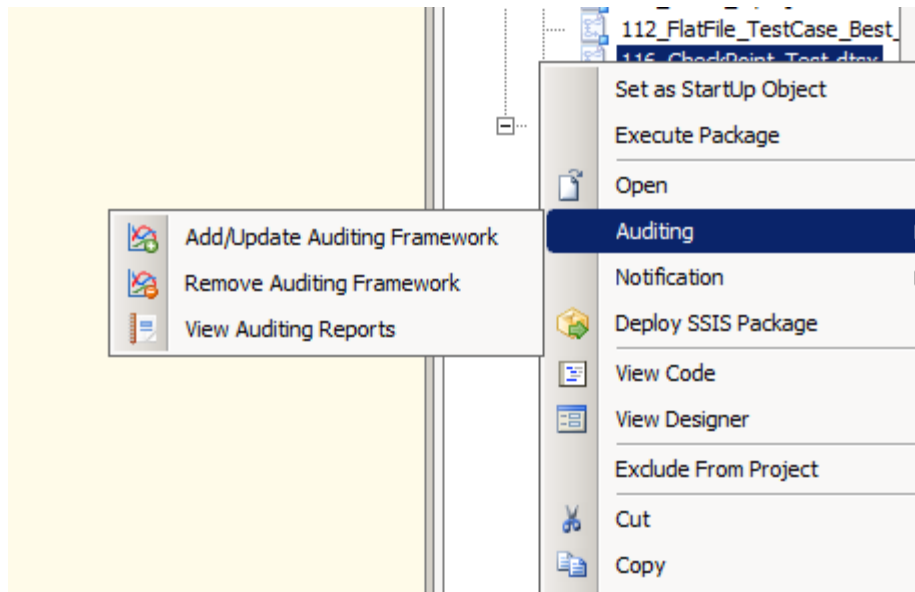
After BI xPress is installed it can be run from outside of Visual Studio or as a BIDS add-in. When in Visual Studio, you can right-click on a package and select Add/Update Auditing Framework. The framework can be applied to a single package or to multiple packages. On the Connection Information tab you select the server and database where you would like to save the auditing information. The default database name is SSISops. Clicking the New option in the database will give you the create screen. This will create the tables and stored procedures needed to run the event handlers.
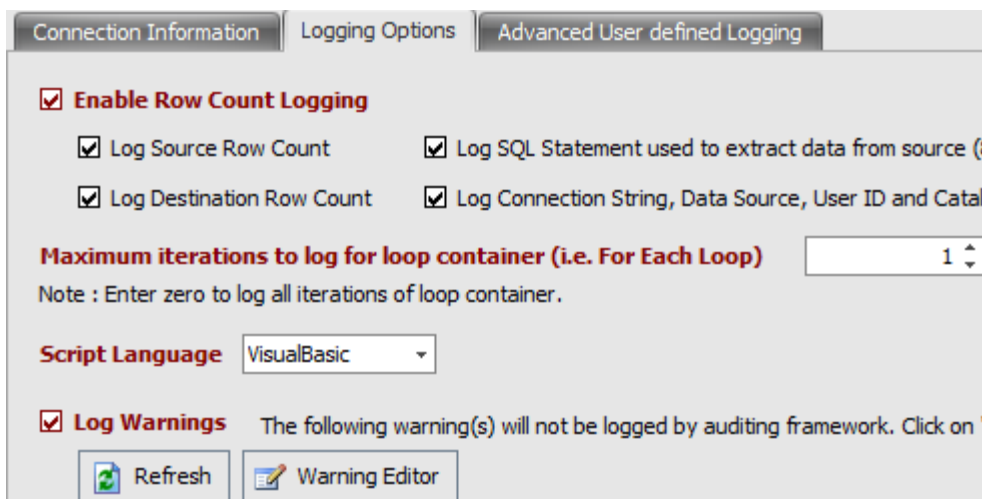

On the Logging Options tab you will select the options of what you would like to log and how many times in a loop to log. You can select your script language and use the warning editor to remove warnings you don't want to log. The Advanced tab allows you to enter custom variable names that you need to log. This is so you can incorporate

your own custom auditing components into the framework. Once all of the options are set click start to apply the auditing framework.

BI xPress will create at least two row counts for each Data Flow in your package. One will log the extracted rows one will log the loaded rows. BI xPress also creates all the needed event handlers to log all the data.

Once the package has run you will need to view the data. BI xPress includes custom reports to view the data in grid and graph forms. The reports contain multiple parameters to allow you to view just the data you need to see.
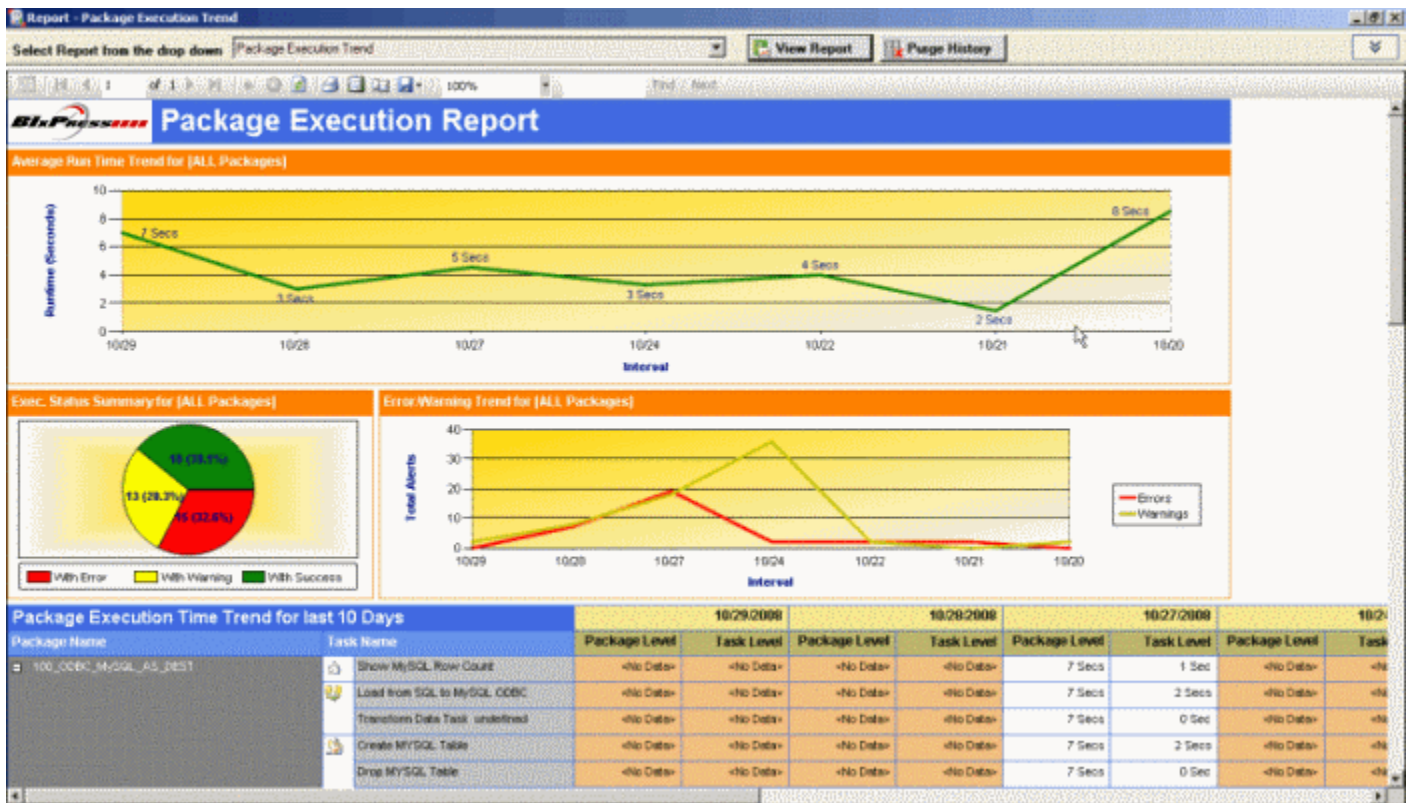

Notification of Problems

If an error happens at 2:00 in the morning, how are you going to be alerted of the problem so you make your SLA. Most businesses need to have packages notify employees about package executions. Whether the package was successful or had an error. Sometime just the fact that the package has started is needed to be sent to notify the proper individuals or groups. Notification via email is the most common way to perform this in SSIS.

Email Notification

The Send Mail Task is SSIS will send can email via SMTP to multiple email addresses. The proper place to create these Send Mail Tasks for notification is in the Event

Handlers of a package. The Event Handlers are called when specific events fire during the run time of a package. For example: The OnPreExecute fires before the package begins execution. This allows you to receive an Email showing the package has begun execution.
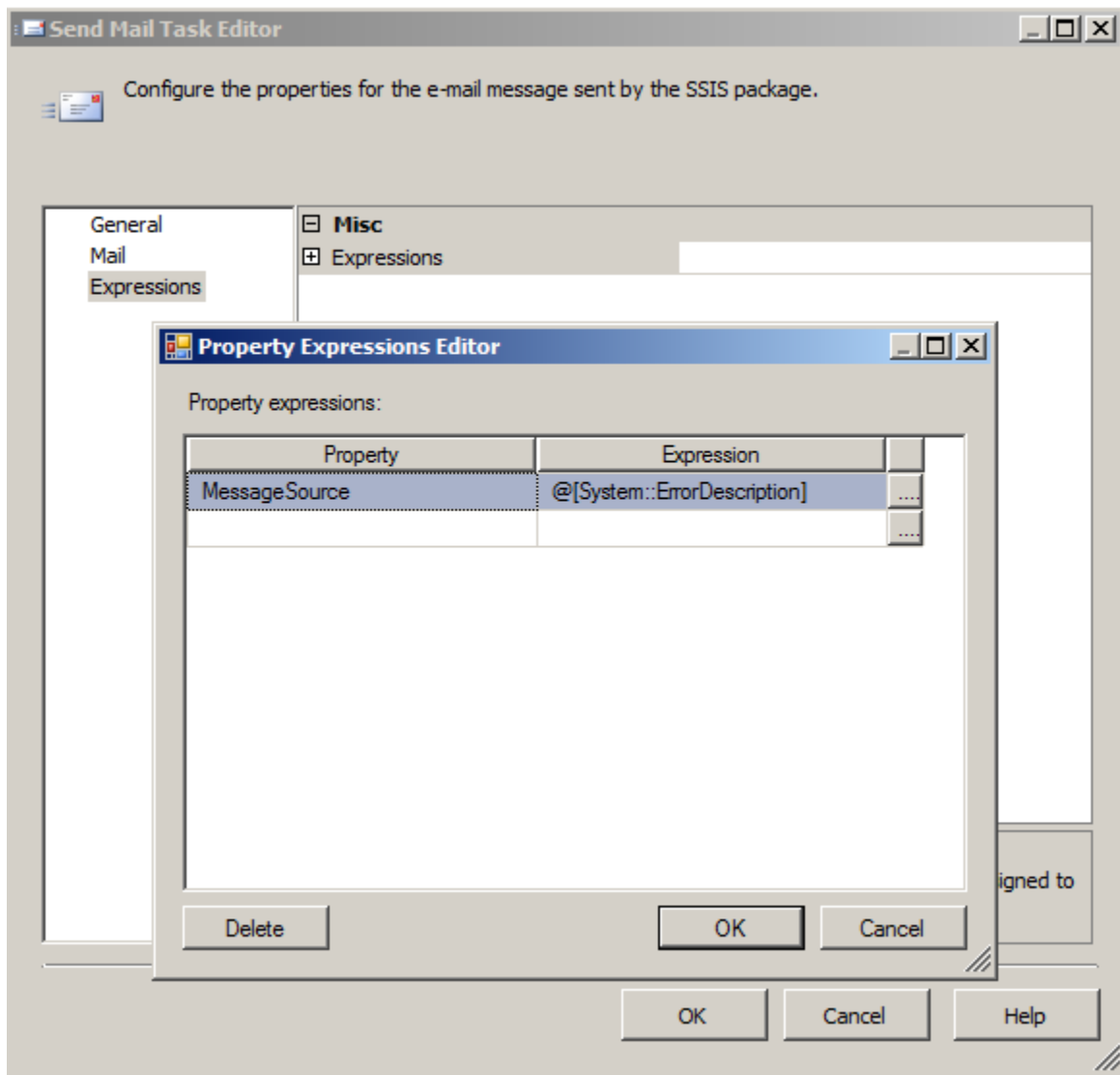
You can also pass in variables from the package to show important information about the package. To be notified of an Error, place a Send Mail Task in the OnError Event Handler and in the message body place the System::ErrorDescription variable. This can be done in the expressions of the Send Mail Task. Any other variables in the package can be used also. If the row counts in a Data Flow have been saved to a User defined Variable using the Row Count Transform, then those variables can be sent to users via email. Again this can be done in the expression node of the Send Mail Tasks in the Event Handler.

Here is an example of an expression to place in the message source of the Send Mail Task in the OnError Event Handler to show all of the important information when an error occurs on a package:

"Package Name: " + @[System::PackageName] + "\n" +
"Start Time: " + (DT_WSTR, 30) @[System::StartTime] + "\n" +
"Error: " + @[System::ErrorDescription] + "\n" +
"Source: " + @[System::SourceName] + "\n" +
"Computer: " + @[System::MachineName] + "\n" +
"User: " + @[System::UserName]

Notice the DT_WSTR command in front of the start time, this is needed to cast the start time as a string instead of a date time. At the end of each line is a new line feed. This is an escape command. This is just one example of how to use the Send Mail Tasks to be notified when a package fails.
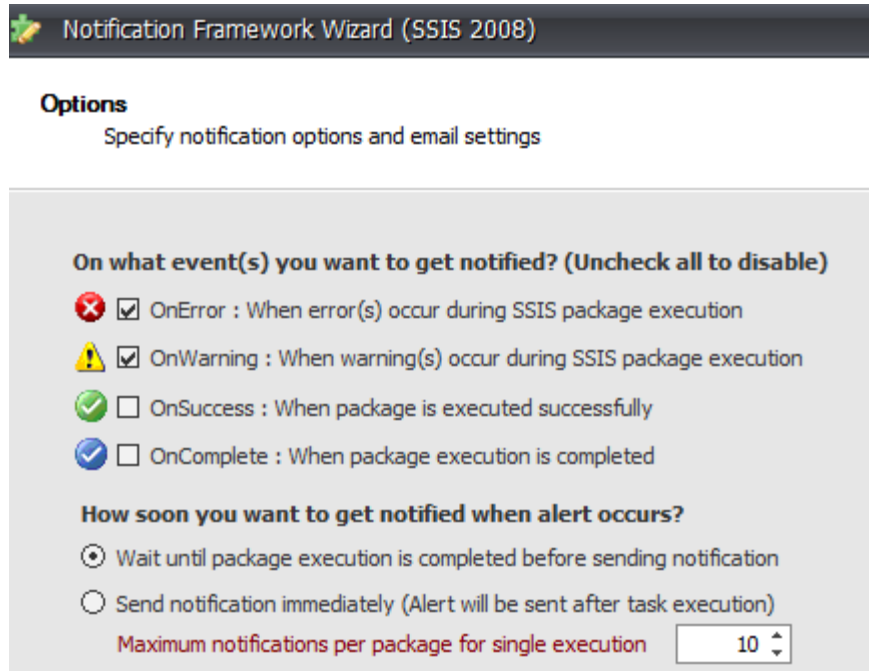
Notification in BI xPress

BI xPress builds the notification into a package automatically. You can customize when and how you would like to be notified. To add notification to a package with BI xPress right click on the package in the package explorer and select Add/Update Notification under the notification menu.
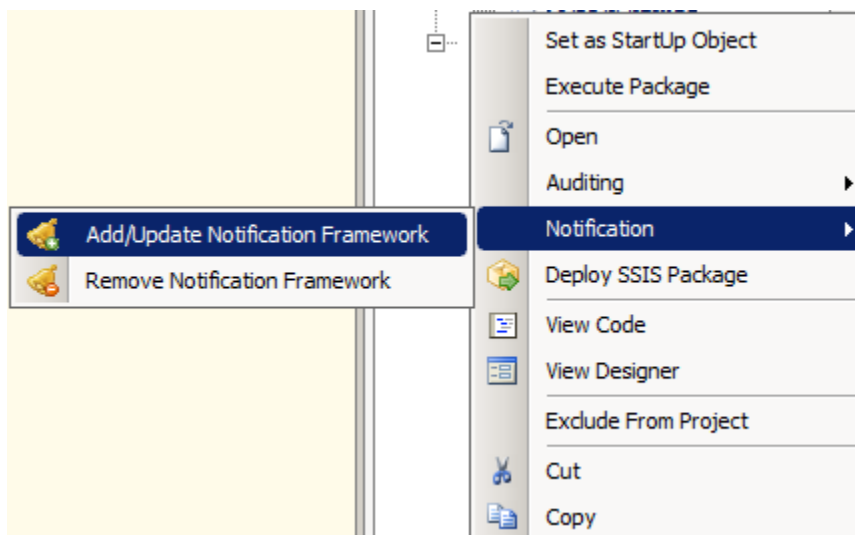
This will start the notification wizard which will allow you to select the option you prefer in notification. BI xPress can apply the notification to one single package or to multiple packages simultaneously. Place a check next to each package that needs notification. BI xPress can set up notification via email and text message over SMTP. You can be notified when a package has a warning, an error, completes, or is successful. This can be customized to notify you when the error occurs or wait until the package completes.

There is even a warning editor to remove warnings you do not want to be notified about.

After setting the options and clicking start on the last screen to add the notification to your packages, you will see a script tasks in the Event Handlers of your packages. These script tasks will perform the notification and send the emails and text messages via SMTP.

SSIS Deployment

So you have developed a set of packages and you're satisfied with how they run in Business Intelligence Development Studio (BIDS), now it's time to deploy your packages so you can schedule them to run over night. There are several methods that you could use to deploy packages. There are also different locations that you can deploy to, which are discussed in the next section titled Storage Options.

One method used for deploying a set of packages is by creating a Deployment Manifest. Once Deployment Manifest is created you will notice it is simply a wizard that deploys all the packages from the project to one location. This is a little limited because often you may only need to deploy one package at a time. To create a Deployment Manifest right-click on the project in the Solution Explorer and select the Deployment Utility
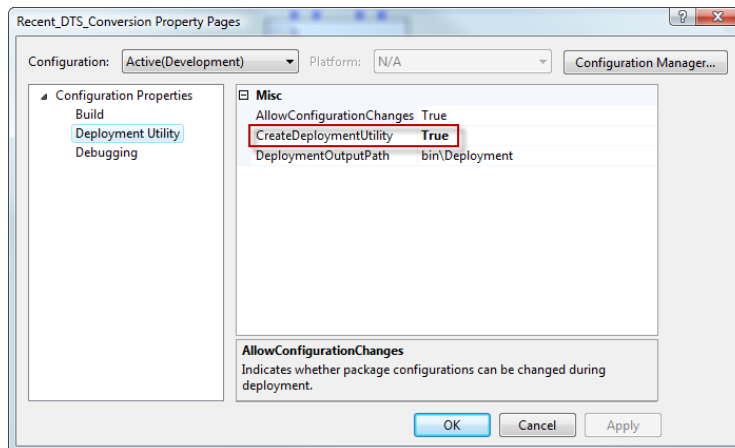
property. Here you will change CreateDeploymentUtility to True. Now the next time there is a build on this project you will find the Deployment Manifest in the bin\Deployment folder inside the same location as the project. This method uses a wizard to deploy all your packages to a location, but what if you only want to deploy one package at a time? To only deploy a single package make sure the package is open and select File -> Save Copy packagename.dtsx As.

Storage Options

The packages are ready to deploy but now you have to decide where to deploy them to. There are many factors to consider when deciding where to deploy your packages. The real deciding factors generally come down to security or ease of deployment. Are you more concerned with the security of your packages, or are you more concerned with the ease of deployment during a development process that could require you to deploy packages several times? So depending on the employer the security of the packages could be the priority and outweigh any other factors, which provides the best reasoning for choosing MSDB for the deployment target.

	MSDB	File System
Disaster Recovery		X
Ease of Deployment		X
Security	X	
Debug		X
Performance	X	X

Deployment Through BI xPress

Wouldn't it be nice to have a simplified process for deploying packages? BI xPress has removed much of the headache that package deployment created in the past. With Business Intelligence Development Studio (BIDS) integration the deployment utility not only allows you to deploy individual or multiple packages at a time but it also will deploy package configuration files. This is key because often you will store configuration files in on location on development

and some where completely different in production. BI xPress also allows for Bi-Directional Transfer, meaning you could move packages back and forth between source and target locations.
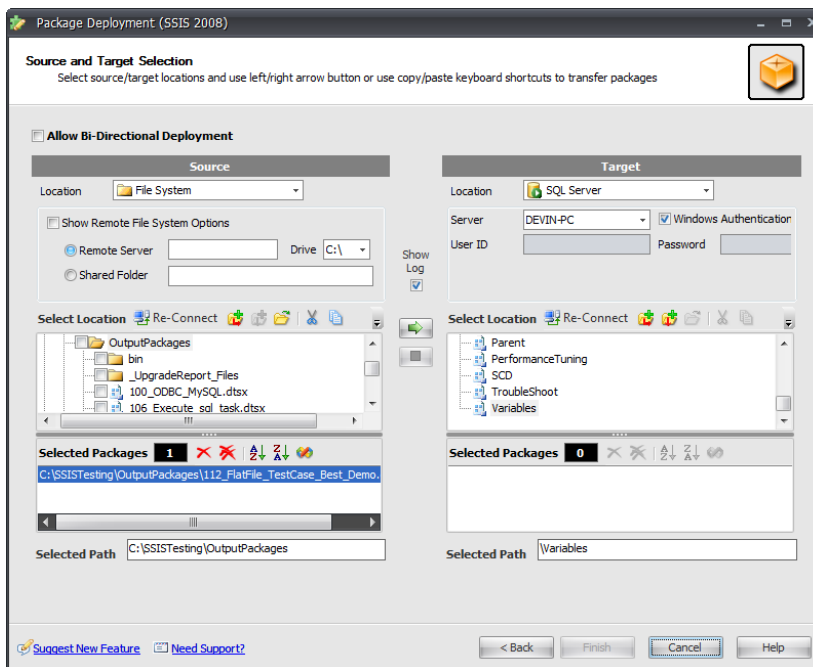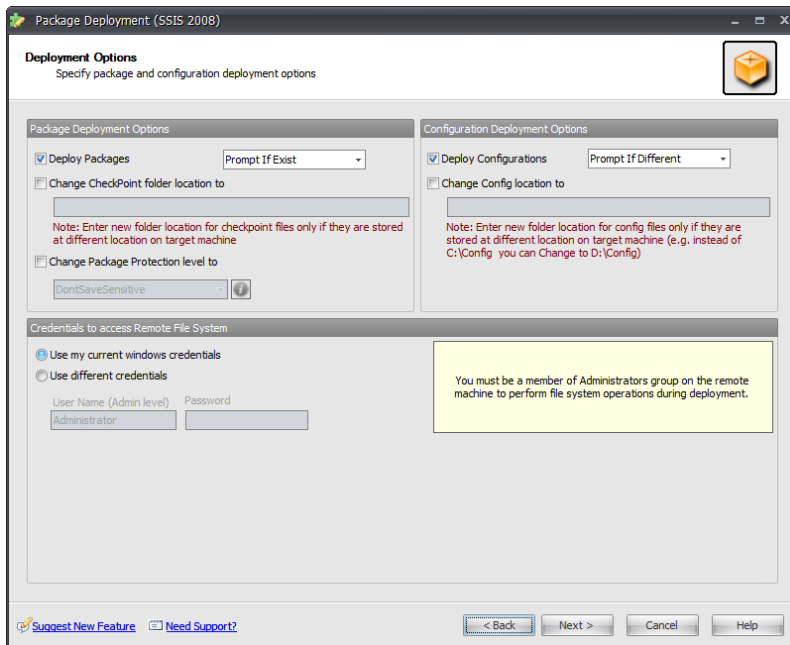
To start the deployment process using BI xPress select the Deployment icon inside of BIDS or inside the standalone BI xPress tool. The first screen that will appear is the deployment options. Here you can specify the desired folder location for checkpoints and configuration files in the environment that the packages are being deployed to. You also have the ability to change the package protection level if you decide this is necessary in the target location.

Now that you have the deployment options set click Next to select the source and target locations for deployment. Here you will tell the tool where the packages are originating from and where you wish to deploy them to. After providing the tool with these two locations simply select the package(s) that you would like to deploy and click the green arrow in the middle to initiate the package deployment.

Scheduling Packages

Using the scheduling features in SQL Server Agent provides the benefit of automating the execution of your SSIS packages on a SQL job. This becomes very useful on servers that are highly active during regular business hours because you can schedule your packages overnight when the server is least active.

To schedule a package you first create a SQL Server Agent job that runs the package then assign a scheduled time for the package to run. You will create the job by connecting to the database engine in management studio on the server you want the package to run. Right-click on the Jobs folder located under SQL Server Agent and select New Job to open the New Job dialog box. Here you will give the job a name and assign a list of step that this job will execute. For example, it is common to have one step that runs all the SSIS packages to update a Data Warehouse then upon completion a second step that process a cube. After selecting the steps page you will create a new step by clicking New, which will bring up the New Job Step dialog box.
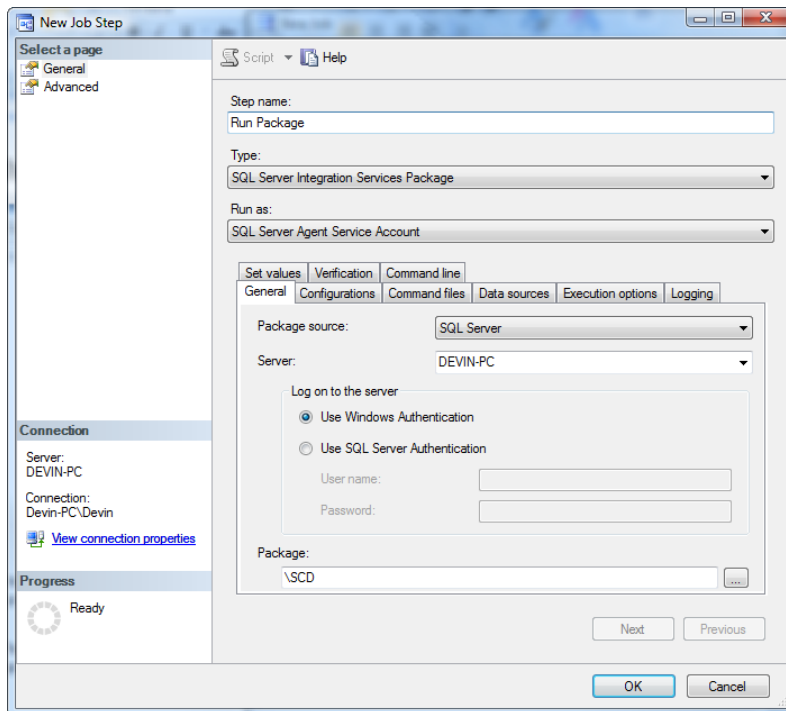
Assuming that the step you want to add is a SSIS package change the type to SQL Server Integration Services Package. Next select the package source, which could be from either a SQL Server or a File System, and choose the package to run. Before clicking OK to complete this step, be sure to give it a name in the Step name property.

The next step would be to schedule this job so this package can run unattended and during non peak server time. Select the Schedule page and select New to enable this feature. This will bring up the New Job Schedule dialog box where you can made decisions on the frequency and duration of this job. Once you have made these decisions give the schedule an appropriate name and click OK to save it.

Instead of creating a new schedule each time you run a job you can also pick from a list of schedules by selecting Pick on the schedule page. Be careful though not to run too many jobs at the same time or you could run into a problem that defeats one of the purposes of scheduling jobs. With too many jobs running at the same time your server could take a pounding much the same that it does during peak hours of usage. For this reason be sure to carefully space out the times that your jobs run.

The data behind the scenes for jobs and schedules is stored in the msdb database and can be queried and reported on just like any other database. The sysjobhistory is one of several system tables in the msdb database that can be used to query and report on job executions.

Common Problems

One of the most frequent problems associated with jobs is with the permissions given to the SQL Server Agent Service Account which is used to run the steps in a job within the SQL Server Agent. If you have created a package that may require permissions beyond what this service account is granted you will probably want to create a proxy account to run the selected job step.

Implementing a proxy account is a three part process. The first step is to create Credentials.   A credential is a record that holds authentication information. Under the Security folder in Management Studio you will find the Credentials. Right-click and select new credentials to open the New Credentials dialog box.
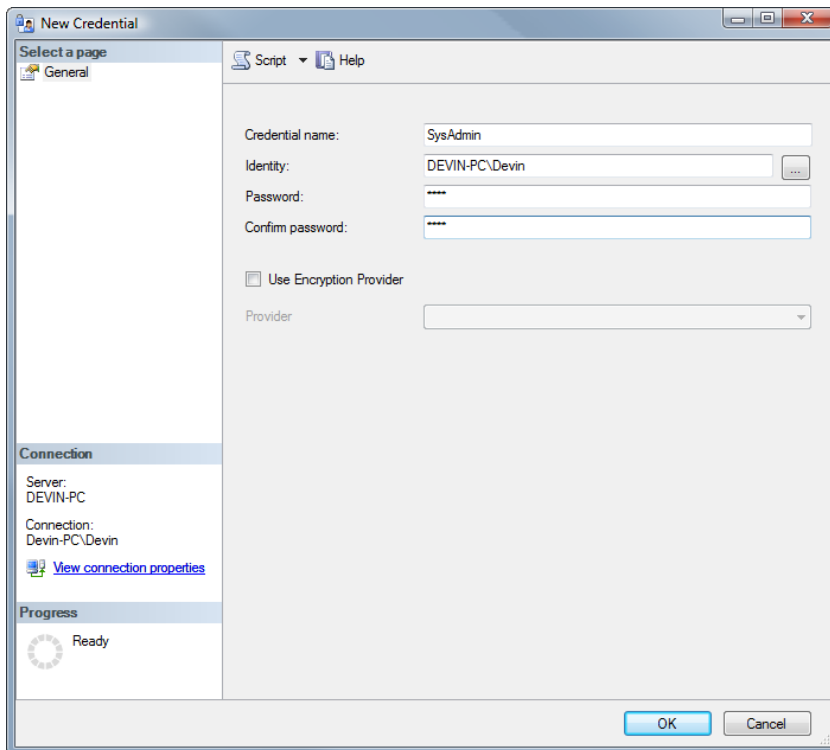
In this window you will provide an Identity, which is the name of the account that has needed rights. Often this will be, but is not limited to, a Windows user account. Next, you will specify the password used for this account. Then finish creating the credential record by giving it a name.

The second step is to create a proxy account that uses the credentials you just created in the last step. In the Object Explorer you will right-click on Proxies

under the SQL Server Agent folder and click New Proxy to create a new proxy account. This will open the New Proxy Account dialog box where you will first input the credentials you just created in the last step. Click the ellipsis next to Credential name to open the Select Credential dialog box. Here you will click Browse to select from the credentials that have already been created. Once you have selected the appropriate credentials click OK twice to return to the New Proxy Account editor. Then select which Subsystems the account will use in the Active to the following subsystems box. In this case we are only interested in using this account to run SSIS packages so place a check next to SQL Server Integration Services Package, shown in the screenshot below. Click OK to complete the proxy account setup.

The last step in this process is to apply the proxy account to the job that requires it. To do this open the job properties then edit the step you will be adding the account to. Here you will change the Run as option to the proxy account previously created.

Conclusion

In this whitepaper, you learned how to use package configurations to give you a zero touch deployment. You also learned how to deploy and schedule your packages using a series of various techniques, including a 3rd party product called BI xPress, which makes the deployment much simpler. Lastly, you learned how to create your own auditing framework for your packages or how to create a framework for notification and auditing in just a few clicks with BI xPress.

For more information:
http://www.pragmaticworks.com
Brian, Devin and Mike's new book:
http://www.amazon.com/gp/product/images/0470496924/sr=1-14/qid=1244072790/ref=dp_image_0?ie=UTF8&n=283155&s=books&qid=1244072790&sr=1-14