# Troubleshooting SSIS Failures and Performance (using BI xPress)

Written By: Mike Davis
MDavis@PragmaticWorks.com
@MikeDavisSQL
Bidn.com/Blogs/MikeDavis

Pragmatic Works

# Contents

## Introduction

SSIS is a fantastic Microsoft tool for performing Extract Transform and Loading (ETL) procedures. A challenge in SSIS though is identifying a problem once you deploy your packages to production.
In this white paper you will learn how to set up auditing and logging on SSIS packages to trap for problems and performance issues. You will learn how to do this with the native tasks built into Integration Services and how to audit SSIS using BI Xpress from Pragmatic Works.

The package used in this white paper can be downloaded at PragmaticWorks.com. All examples, code, and figures will be based on this package. Below is an image of the package with a brief description.

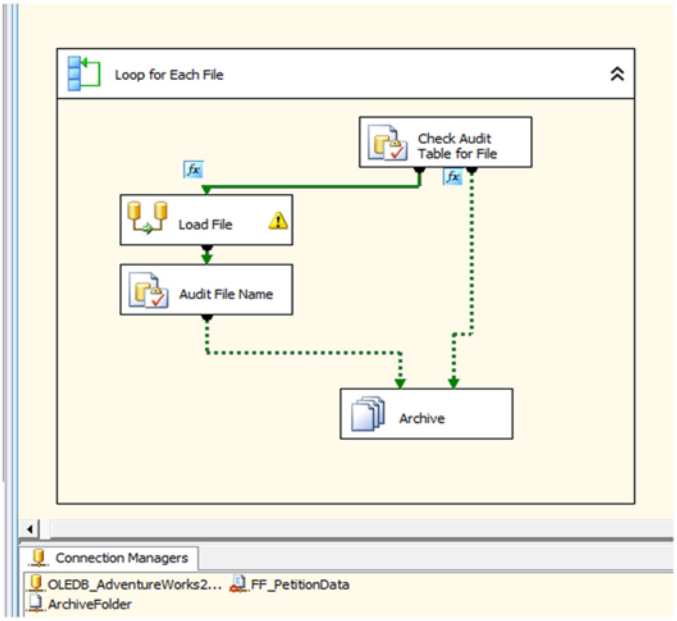


Figure-1

This package will loop through a set of file. Each file name will be compared to a table to determine if the file has already been loaded. If the file is not found on the table it will be loaded into a table. Then the file name is added to the audit table. Finally the file is moved to an archive folder. If the file is already on the audit table it will be archived only.

The example package has three variables:
1. intFileCount = Hold the number of files
2. intRowCount = Hold the number of rows in a file
3. strPetitionFiles = Holds the file name currently in the For Each Loop

## 1. Setting up SSIS Auditing

SQL Server Integration Services (SSIS) is a powerful tool used to move data. Once you have created several SSIS packages and scheduled them to run in your production system you inevitable have failures at some time. Having a robust auditing framework will make troubleshooting and performance tracking your packages much easier.

Imagine you have dozens of SSIS packages that run in SQL Agent jobs throughout the night in your production system. You arrive at work the next morning you find data is missing for reports that are critical to the business. It is your job to find out why this data is missing. First you need to know the package that is loading this data. Next you need to find the error (if there is one) or any other

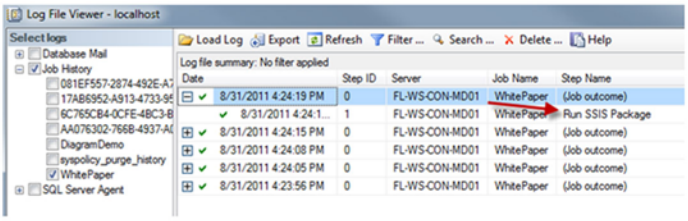


Figure-2

The first stop for most SSIS troubleshooters is the SQL Agent job log. In this example all of the packages ran with no error last night. You can see this job history in Figure 2. What now?

If you have logging or a form of detailed auditing on your packages then you would be able to track down the issue. In this example the issue was the package loaded no rows due to a where clause in a Data Flow Source so there was no error to find. This problem would be obvious if you have auditing on the row counts of your Data Flows.

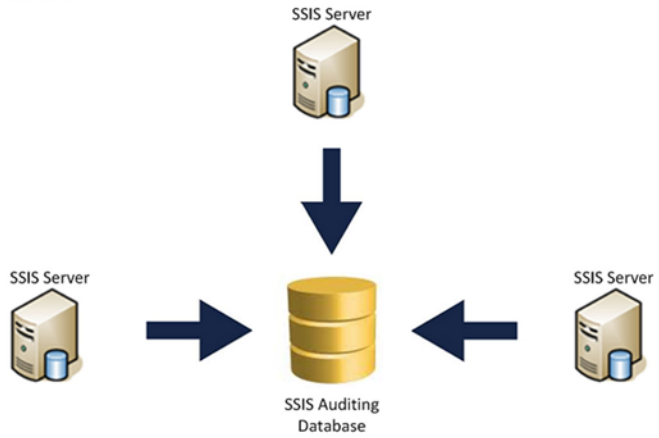


Figure-3

With packages spread across multiple servers, having all of the auditing data in one location makes tracking SSIS package issues and performance easier. A form of centralized auditing would log the run time information of packages from each of your servers. In Figure 3 you can see an example of this type of setup. There is a central database that holds all of the auditing data. Several methods exist for monitoring SSIS,native logging, and custom logging frameworks.

## 1.1 Logging

The native logging feature in SSIS can write information about the package run into several locations. To open the logging screen right click in the control flow of an SSIS package and select logging. The logging menu can be seen in Figure 4.

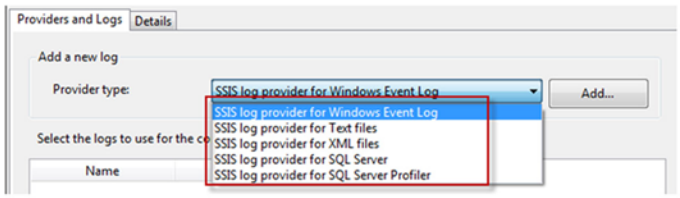


Figure-4

### 1.1.1 Logging to Files

Once in the Logging window you will need to place a check next to the package name in the top left. Select SSIS log Provider for Text Files and click add. Then place a check next to the newly added log provider. Under Configuration select a file name where the logging information will be saved. See figure 5 for an example of this set up.

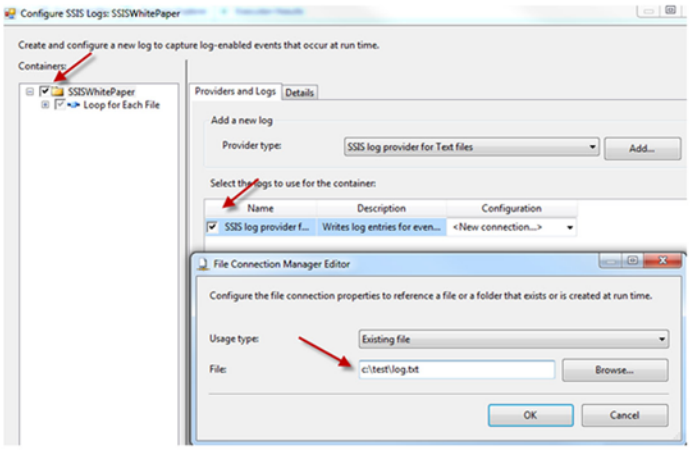


Figure-5

Under the Details tab you can select which options you would like to save and on which events. Select the following event handlers, OnError, OnWarning, OnPostExecute, and OnPreExecute. These Events call any tasks you have added to the corresponding Event Handler window. For Example, when a task is run in the Control Flow the onPreExcute task is called, if you have placed an Execute SQL Task in the onPreExecute event handler window, the Excute SQL Task would execute in the Event Handler. If the Control Flow task causes an error, the

onError event will be called and execute any task under the onError. These are the most common event handlers logged for packages. They tell you about errors, warning, start times, and stop times.

The advanced button at the bottom of the details screen allows you to select the information collected at each event. Leave the advanced settings at default for this example.

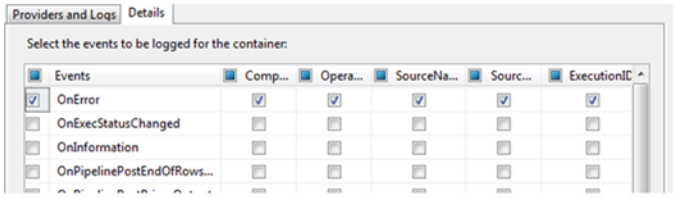


Figure-6

Once you have configured the text file logging for the package run the package one time by right clicking on the package name in the solution explorer and click execute package, and then open the log file.

Note: There is a Reset Package in the solution you can run to truncate the tables and move the file back.

In figure 7 you can see a small extract from the log file. This file is approximately 550 lines of text. This is a lot of information about a small package. A lot of the information is repeated also. Some of this repeating is due to the how the event handlers are fired in SSIS. The SSIS packages fire some events multiple times, once for the package and once for the tasks. This makes the logging cumbersome to read and hard to find the important information you are looking for in a logging solution.

Once you have configured the text file logging for the package run the package one time by right clicking on the package name in the solution explorer and click execute package, and then open the log file.

*Note: There is a Reset Package in the solution you can run to truncate the tables and move the file back.*

In figure 7 you can see a small extract from the log file. This file is approximately 550 lines of text. This is a lot of information about a small package. A lot of the information is repeated also. Some of this repeating is due to the how the event handlers are fired in SSIS. The SSIS packages fire some events multiple times, once for the package and once for the tasks. This makes the logging cumbersome to read and hard to find the important information you are looking for in a logging solution.

Figure-7

Since this log file is a CSV you can open it in excel. A major problem occurs in trying to read through the error messages in Excel. Some SSIS errors contain commas in there description. This breaks the error up into separate columns. Trying to load this file into a table or some other data driven tool would be problematic at best. If you want to log the information to a table, you should select the SQL Server provider in the logging options in the first place.

### 1.1.2 Logging to SQL Server

Logging to a SQL Server table gives you querying power of the database engine. To set this up, go back to the logging menu by right clicking in the control flow of the package and select logging. Delete the text logging if it still exists. Once in the Logging window you will need to place a check next to the package name in the top left. Select SSIS log Provider for SQL Server and click add. Then place a check next to the newly added log provider. Under Configuration select a Database where the logging information will be saved and run the package. This example will log to a database named testing.

The same logging details and options exist as in the text file logging example above. Select the following event handlers, OnError, OnWarning, OnPostExecute, and OnPreExecute. Now run the package. After the package has completed, open SQL Server Management Studio (SSMS) and run the following query in in the Testing database selected in the logging options.

*Select * From SysSSISLog*

This will return the logging data. This table is found in the system table folder of the database. Now that the data is in a table you have a lot more control of how to display the data. The issue still exist where the messages (which is the information you need) is duplicated. It is shown once for the package and once for the tasks that sent the message. To control how often the data is written to the table you will need to build a custom solution in the event handler of the package.

### 1.2 Creating a Custom Auditing Framework

Creating an auditing solution is time consuming but once built gives you the detailed information you want in your table and allows you to filter unnecessary data. You will create this custom auditing by adding tasks to the Event Handlers of the package.

#### 1.2.1 Event Handlers

There are several event handlers listed under the event handler tab. Click on the event handlers tab at the top of the example package and you will see two drop down menus at the top. On the left there is a drop down with the tasks in the package and the package itself as seen in figure 8. You can create specific auditing for each task if desired. In this example you will create auditing for the entire package, so ensure the package name is selected.

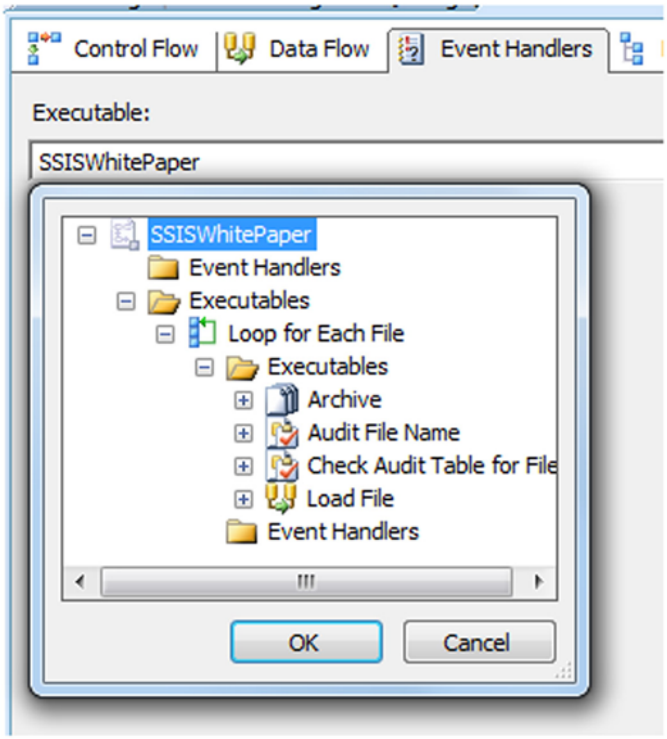


Figure-8

The right drop down menu contains the events available for the package. Select the onError event handler and click the blue link in the center to create the onError event handler. Before you can start auditing you will need to create a table to write the data too. For this example you will be auditing the package name, task name, error description, and the date. Open SSMS and run the following SQL in your auditing database to create the auditing table.

```
CREATE TABLE [SSISErrorLog](
  [RunTime] [datetime] NULL,
  [Package] [varchar](500) NULL,
  [Task] [varchar](500) NULL,
  [Error] [varchar](500) NULL
) ON [PRIMARY]
```

### 1.2.2 Execute SQL Task

Now you are ready to insert data into this table. Before we insert data we need to create one more variable. There is a problem with the date format in SSIS. The DateTime format in SQL is different than the System variables in SSIS. The format in SQL is 1900-01-01 00:00:00.000, and the format in SSIS is 1/1/1900 12:00:00 AM. So you will need to convert the SSIS date to the SQL format. To do this, create a variable on the package named strStartTimeand set the type to string. Set the variable to evaluate as an expression in the properties of the variable. Click on the expression ellipsis and enter the following code:

(DT_WSTR, 10)(DT_DBDATE)@[System::ContainerStartTime] + " " + (DT_WSTR, 8)(DT_DBTIME)@[System::ContainerStartTime]

This is the SSIS Script language. It will convert the start time of the current container to a format SQL will recognize.

Go back to the package onError Event Handler. Drag in an Execute SQL Task. Open this task and set the connection to the testing database. Enter the following SQL into the Execute SQL Task. Notice the convert function used to convert the string value of the date to a datetime for SQL.

```
INSERT INTO SSISErrorLog
    ([RunTime]
    ,[Package]
    ,[Task]
    ,[Error])
VALUES(CONVERT(datetime,(?)),?,?,?)
```

Click on the Parameters tab of the Execute SQL Task and enter the parameters as shown in figure 9 below. Notice the first parameter is the variable you create previously, the rest are system variables. Click ok to close the task and return to the control flow of the package.

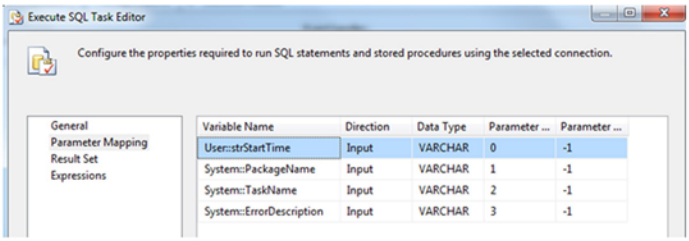


Figure-9

You will need to cause an error in the package to have the Event Handler fire. Open the first Execute SQL Task in the For Each Loop and put the letter 'X' in front of the SQL command. This will cause a syntax error. Run the package. The package should fail. Open SSMS and query the SSISErrorLog table and you should see the data from the package run as seen in figure 10 below.

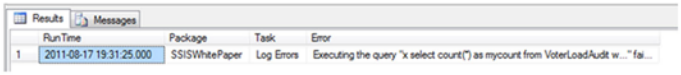


Figure-10

If you donot see any data, return to the package and look under the Progress/Execution Results tab and find the error on the event handler. It should tell you why the insert statement failed.

### 1.2.3 Expressions

This was a simple example of writing data to a table to audit a package. You can use more variables and expressions to make the package more customized. For example you can create some of the variables below and use the corresponding expressions. These variables would be the parameters in the Execute SQL Task instead of the system variables. Of course you would need to alter your table to write these

| VariableName | Variable Type | Expression |
|---|---|---|
| strUser | String | @[System::MachineName] + "\\" + @[System::UserName] |
| strDate | String | (DT_WSTR, 10) (DT_DBDATE) @[System::ContainerStartTime] |
| strPackageLoc | String | @[System::MachineName]+ "\\"+@[System::PackageName] |
| strExecBy | String | @[System::InteractiveMode] == true ? @[System::UserName] : @[System::MachineName] |

You can see by this small example that creating and maintaining a robust auditing solution will take quite a bit of time. This type of solution would need to be added to every package in your environment that you need to audit. You can use a package as a template and make any adjustments to the auditing as needed during package development.

To avoid this time consuming work, you can use a tool by Pragmatic Works that can do this work for you. That tool is BI xPress.

### 1.3 BI xPress

BI xPress by Pragmatic Works can be downloaded at Pragmatic Works.com and comes in a free Community Edition or higher scale edition, which has a free trial.BI xPress has hundreds of features but in a nutshell it will help you turn on notification and auditing easily by injecting code into your package automatically. It can also speed up development by helping with code reusability.

Once you have BI xPress installed you will see a new toolbar appear at the top of BIDS. You will also get some new right click menus in the Solution Explorer as seen in the figure 11.

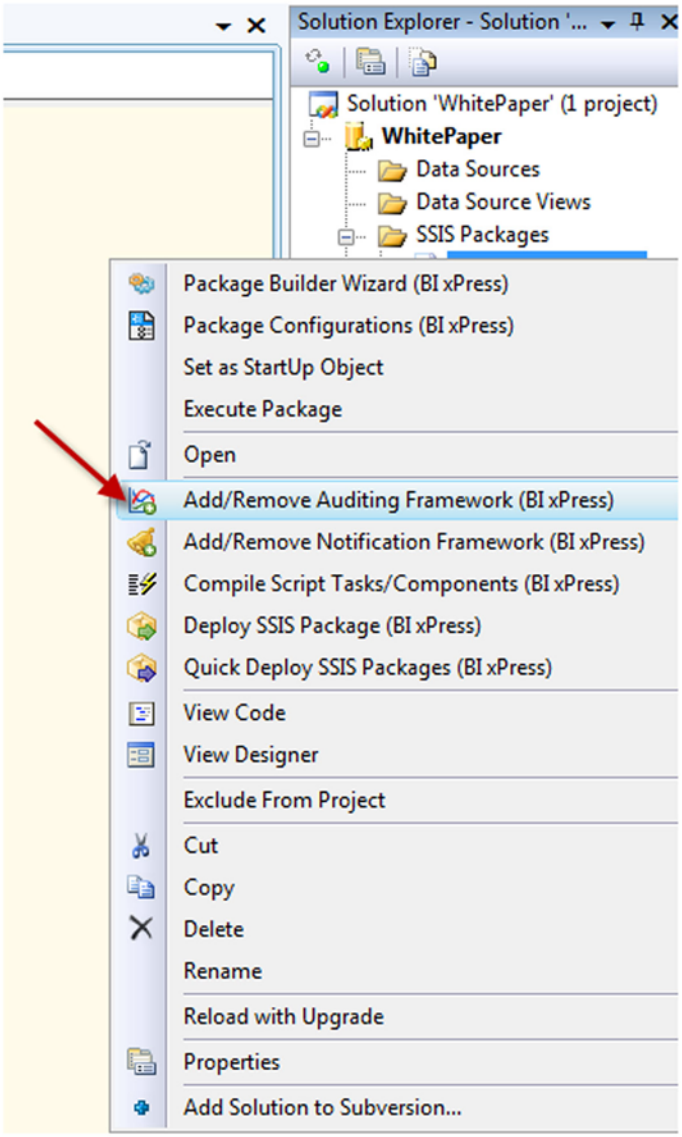


Figure-11

You will notice there is an option to Add/Remove Auditing Framework. If this is the first time you have run BI xPress you will need to create the BI xPress auditing database. BI xPress will create this for you.1.3.1 Automatic Creation of the Audit Database

To create the BI xPress auditing database right click on the package in the solution explorer and select Add/Remove Auditing Framework. Click next on the first window. Select the radio button next to Add/Re-Apply Auditing Framework and click next. Ensure there is a check next to the package in the next window and click next again. Click the Create Database button and select the desired database name and settings as seen in figure 12. Once you have the desired settings in place click Create Database. You have now created a complete auditing database with the needed tables and stored procedure to audit your SSIS packages.

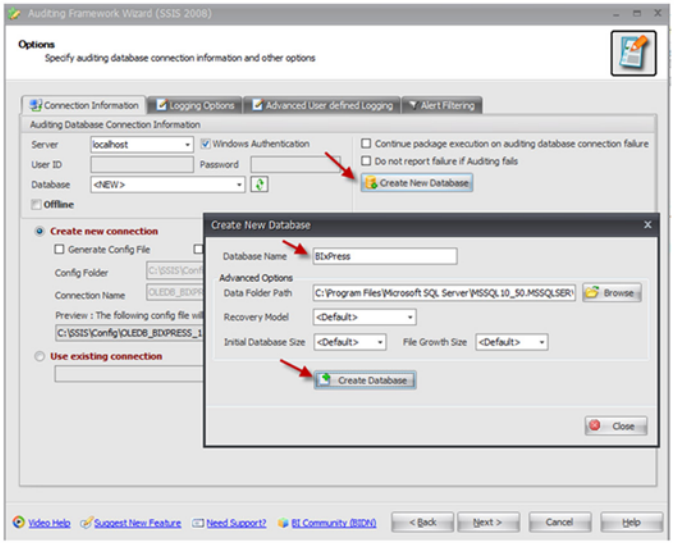


Figure-12

## 1.3.2 Applying the Auditing Framework

Now that you have created the auditing database you are ready to apply the auditing framework to the package. In BIDS right click on the package in the solution explorer and select Add/Remove Auditing Framework. Click next on the first window. Select the radio button next to Add/Re-Apply Auditing Framework and click next. Ensure there is a check next to the package in the next window and click next again.

In the Server field, enter the name of the auditing server. This is where you created the auditing database in the previous section. Then select the auditing database in the database drop down menu. In this example the auditing database is named BIxPress as seen in figure 13.
Uncheck the Generate Config file. It is a best practice to use configuration file or configuration tables on all packages. To simplify this example you are skipping this step.

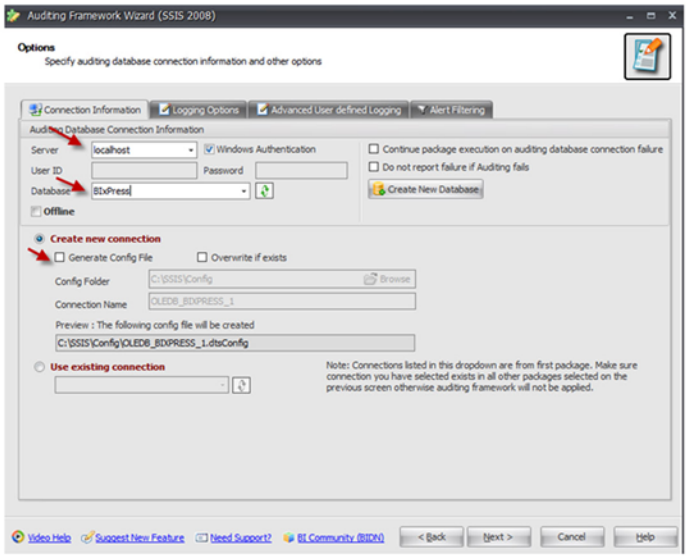


Figure-13

Now click on each of the tabs across the top of the window to see all of the available options for auditing. You will leave all of these options at the default values for this example. But take notice of the ability to customize your auditing at a very low granular level. You are even able to filter out alerts in the last tab on the right. Now click the next button and click the start button on the next

screen. This will create the auditing tasks in the Event Handlers of the package.

Once BI xPress has finished applying the auditing framework it will close the window automatically. Return to BIDS and you will receive a popup asking you to reload the package as seen in figure 14. Click the Yes to All button.

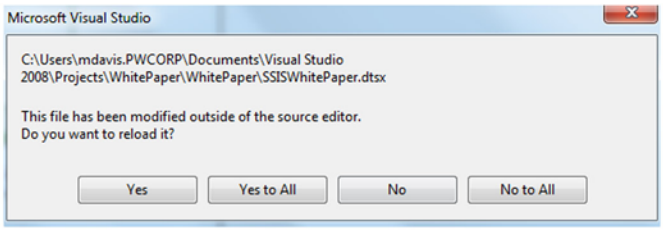


Figure-14

Click on the Event handlers tab to see the new task added by BI xPress seen in figure 15.

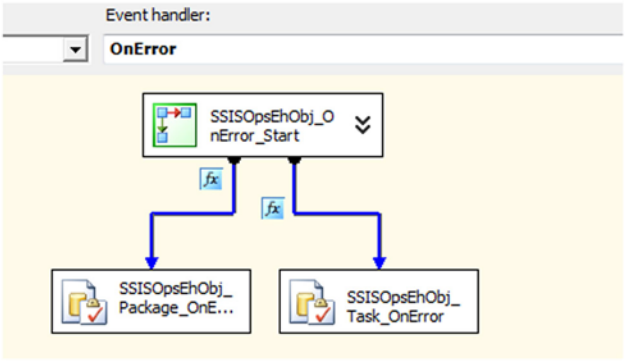


Figure-15

Open SSMS and you will see the tables a stored procedures in the auditing database you created, seen in figure 16. If you run the package with the auditing framework it will write to these tables.

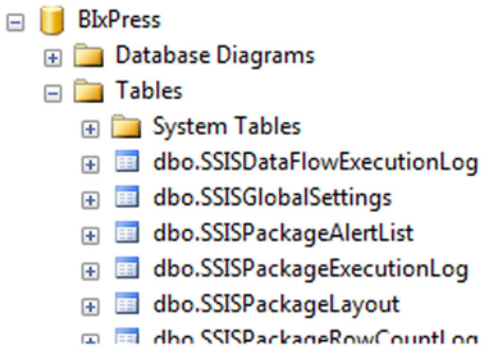


Figure-13

## 2. Troubleshooting SSIS Failures

Inevitably your SSIS packages will have a failure at some point. In most environments SQL Agent jobs will be schedule to run during the night. The next day you will realize the packages have failed. The quality of the auditing on your SSIS packages will determine how easy it is for you to find the issues.

### 2.1 Viewing Auditing Data

The logging, covered in the first section of this paper, contains the information about auditing the package executions. Viewing these should show you the information needed to troubleshoot the packages. Although the native logging captures so much data it can be difficult to find the desired information. This section covers viewing this data in with different methods.

#### 2.1.1 Viewing Log Files

Viewing the log files from an SSIS package is fairly simple. If you same the file as a text file, then double clicking on the file, it should open it with notepad or your default text viewer. Viewing the data is easy but finding the important messages can be troublesome. You can search the text file for key words like “Error”.

Type CTRL+F on your keyboard and type error in the find box. Figure 17 below shows an example of this technique.

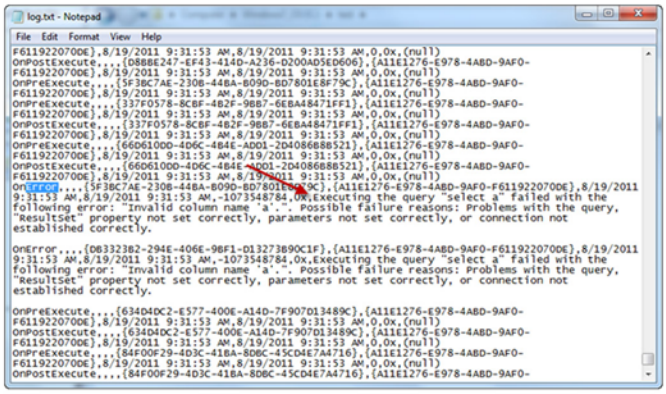


Figure-17

Due to the obvious limitation of this method, text file logging is not the most popular method and it is virtually impossible to monitor the performance trends of packages with these log files.

### 2.1.2 Viewing Log Tables

The logging option of SQL Server in SSIS makes it easier to view the errors and other messages from the packages. The same is true if you set up your own custom auditing tables as shown previously. It is easy to search through the messages because you have the power of the query engine at your disposal.

In the figure 18 below you can see the data on the log table. With a little work on a query you can find the error messages from the SSIS package

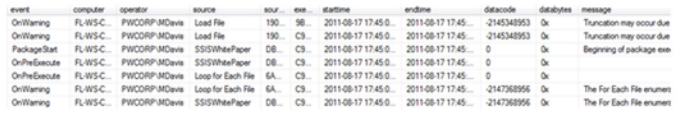

Figure-18

Run the following query on the package and you will see the errors on the log table.

```
SELECT [event]
      ,[computer]
      ,[operator]
      ,[source]
      ,[starttime]
      ,[endtime]
      ,[message]
  FROM [Testing].[dbo].[sysssislog]
  Where event = 'OnError'
```

Figure 19 shows the results of the query

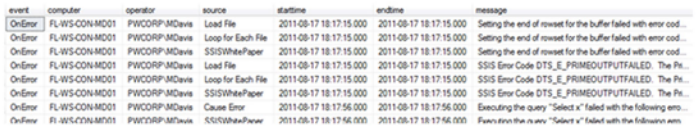

Figure-19

Now you see the errors from the packages. With a little more work on the query you can eliminate the unwanted rows and retrieve the rows you desire. This query would need to be altered every time you want to view a different package or different event.

### 2.1.3 Viewing Manual Auditing

The same issues apply here that applied in the previous section. Viewing the data on a custom table may be easier due to you having more control, but the data is not in a report format and you would still need to develop some type of reports, maybe in Reporting Services, to show this data to end users.

### 2.1.4 Custom Reports

Whether you want to create reports from the native logging tables or your own custom logging tables, Reporting Services (SSRS), is the tool for

report created using the native logging tables from SSIS. This report has parameters on it that allow for the selection of events, start time and end time. It shows the details of the SSIS package execution and the message text below. This 2008 r2 report can be downloaded from PragmaticWorks.com

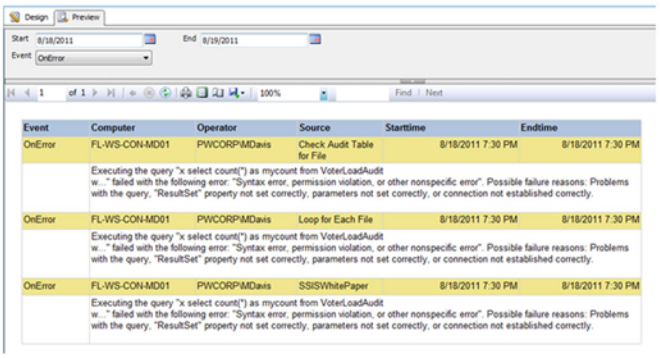


Figure-18

This report uses a very simple query with parameters.

```
SELECT      event, computer, operator, source,
starttime, endtime, message
FROM        sysssislog
WHERE       (event = @Event) AND (starttime
BETWEEN @Start AND @End)
```

The event parameter has a drop down menu with a list of all of the available events in the SSIS log table. The query for the events uses a distinct function to show the events once.

```
SELECT DISTINCT event
FROM        sysssislog
Order by Event
```

If you have created a custom log table then you will need to adjust these queries and probably write new ones to pull the data in the format you created. Reports like this show you information on the package runs, but they do not allow you to easily compare run times and row counts in the packages. You will need to create more complex trending reports to be able to analyze and compare executions.

## 2.2 BI xPress Monitoring Console

Once you apply the auditing framework to a package with BI xPress, you will be able to start analyzing the data in those tables immediately with the monitoring console that is included with BI xPress. There are several built in reports. You can even do live monitoring of packages while they are running on the server. The reports even have images of the packages and looks as if you are watching the package run in BIDS live. You can open the monitoring console in two ways, In BIDS, or in BI xPress stand alone. To open in BIDS, click on the BI xPress drop down menu from the top and select Monitoring Console seen in figure 21.

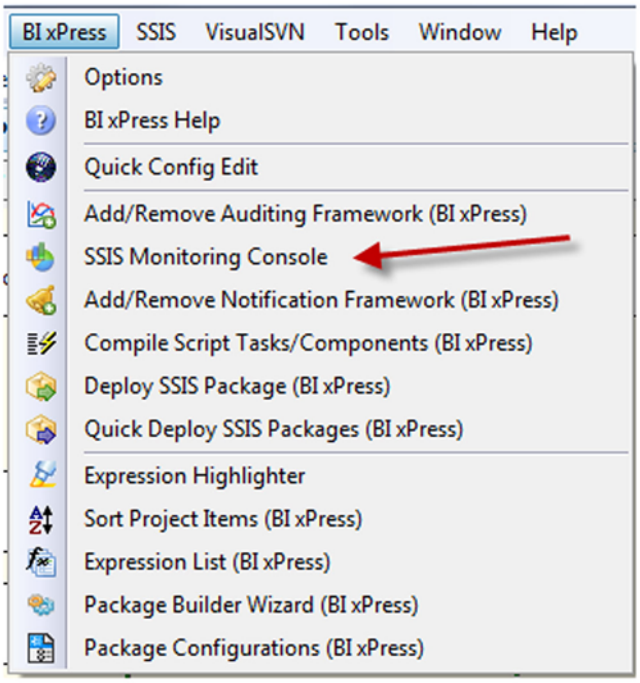


Figure-19

To open BI xPress, click on the windows start button, go to all programs, and open the BI xPress folder and launch BI xPress. Then click on the Monitoring button seen in figure 22.

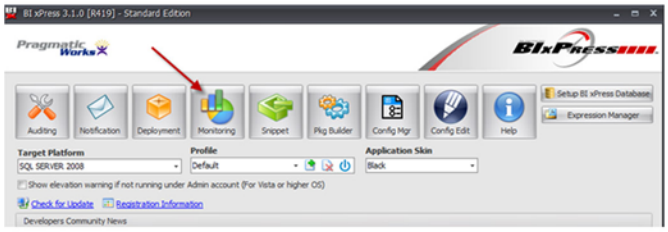


Figure-22

## 2.2.1 Static Reports

On the right hand side of the first BI xPress monitoring console you will see a list of the static reports. These are the reports that show data of the package runs in the past. There are reports that show performance trends and row count trends also.

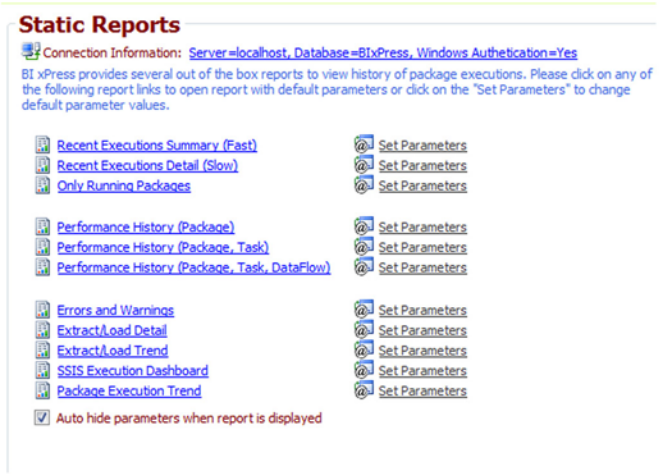


Figure-23

The Recent Execution Summary reports show the details of the report runs. In figure 24 below you can see some of the details of the package runs provided by the report. You can view the details in each of theitems by clicking the plus next to the items.

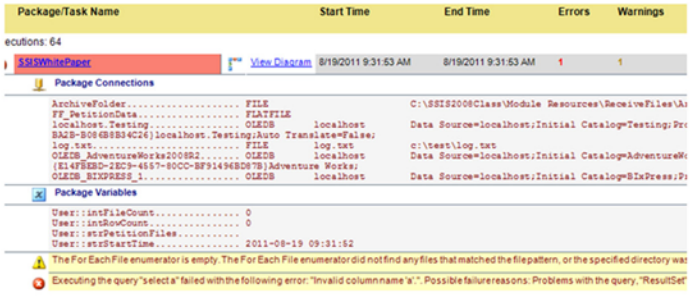


Figure-24

Each report will show you a visual of what the package run would look like in the BIDS environment. This can be seen under the control flow tab and the data flow tab as seen in figures 25 and 26 below.

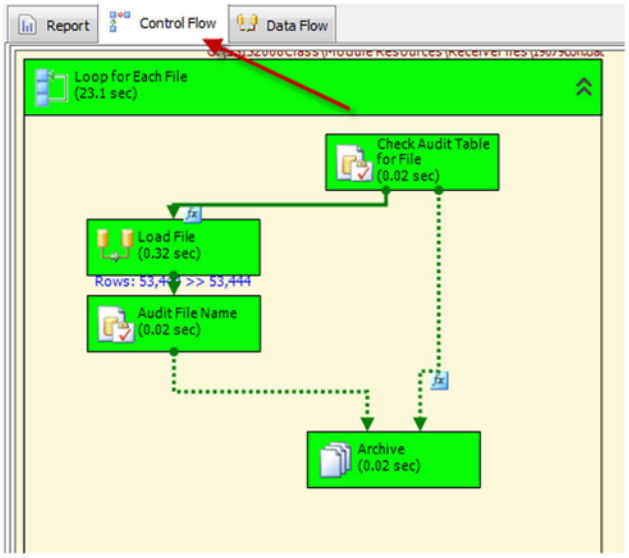


Figure-25

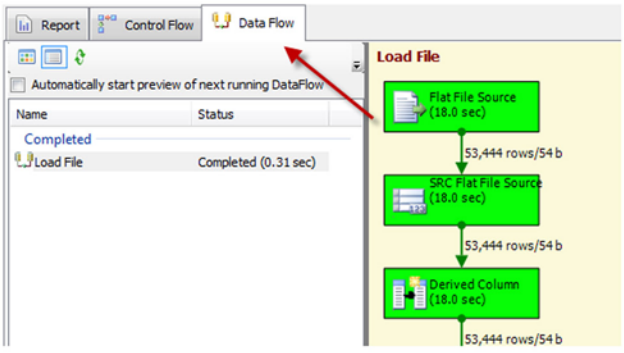


Figure-26

To open BI xPress, click on the windows start button, go to all programs, and open the BI xPress folder and launch BI xPress. Then click on the Monitoring button seen in figure 22.

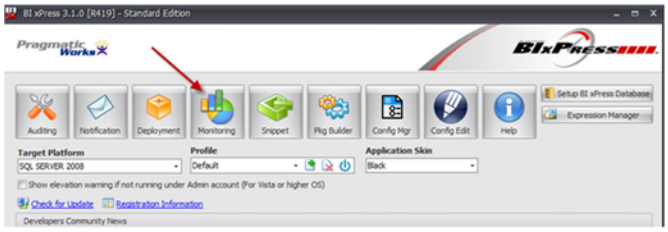


Figure-22

## 2.2.1 Static Reports

On the right hand side of the first BI xPress monitoring console you will see a list of the static reports. These are the reports that show data of the package runs in the past. There are reports that show performance trends and row count trends also.

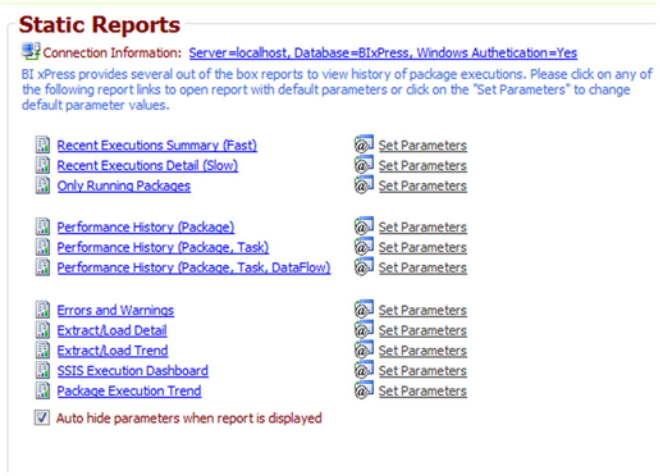


Figure-23

The Recent Execution Summary reports show the details of the report runs. In figure 24 below you can see some of the details of the package runs provided by the report. You can view the details in each of theitems by clicking the plus next to the items.

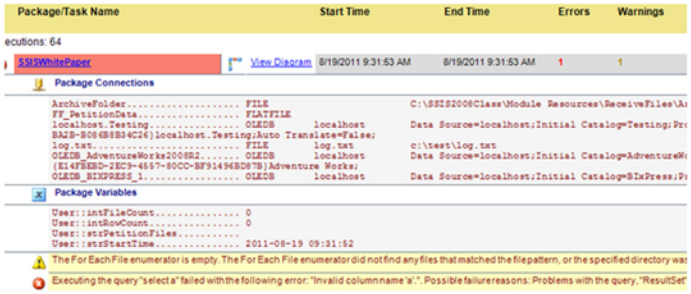


Figure-24

Each report will show you a visual of what the package run would look like in the BIDS environment. This can be seen under the control flow tab and the data flow tab as seen in figures 25 and 26 below.

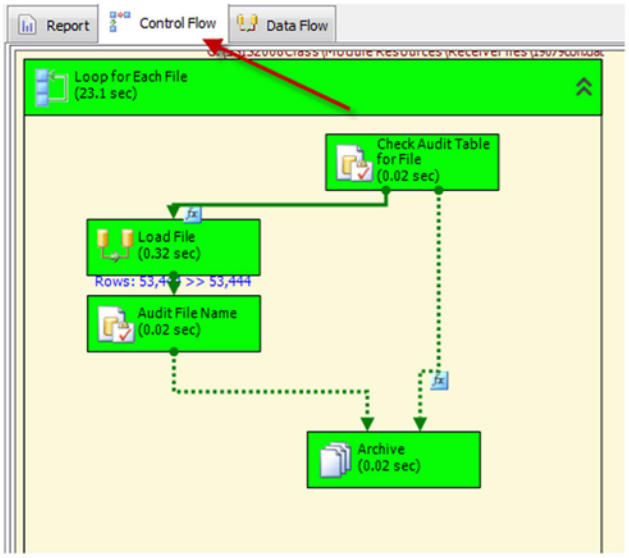


Figure-25

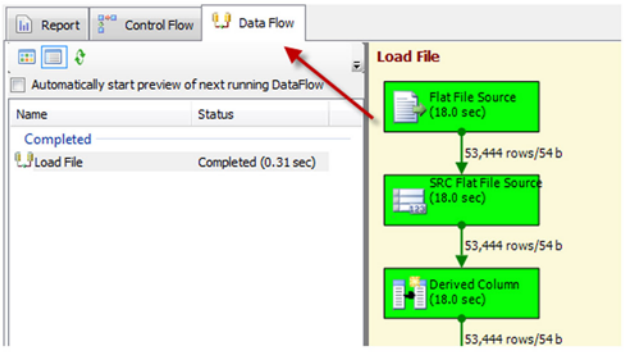


Figure-26

### 2.2.3 Live Monitoring

One of the best features of the BI xPress Monitoring Console is the ability to watch your packages run on any server as if they were running in BIDS. You may have dozens of servers and hundreds of packages spread out across those servers. The BI xPress Monitoring Console give you the power to see the packages run in real time on any of those servers.

On the left side of the home screen in the monitoring console, there are two options for live viewing, one for single packages, and the other for multiple packages. If you have a master package calling several child packages, you can watch them all one screen.

In figure31 below you can see a master package in the bottom right and the three child packages running too. You can see the times for each task. If there is a failure in any of the packages it will show on the screen in real time.

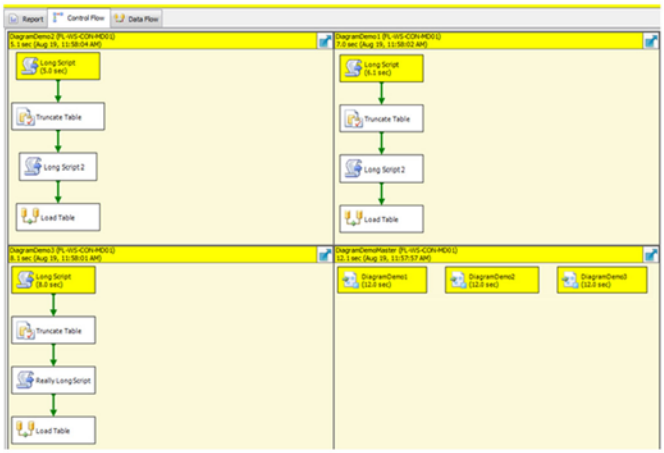

Figure-31

## Summary

This white paper has covered the auditing and monitoring options available in Integration Services. Whether you use the native tasks, build your own custom solution, or use BI xPress, it is critical to any business to have auditing on your packages and a way to show that data in a practical manner. The native logging gives lots of data but parsing the data can be cumbersome. A custom solution gives you complete control, but is time consuming to develop. BI xPress gives you a complete auditing and monitoring solution and is