Michellea David

sqlmichelleadavid@gmail.com

Centralize SSIS

Store all SSIS projects on one SQL Server

Contents

[Centralize SSIS 1](#_Toc3475745)

[Advantages 1](#_Toc3475746)

[Disadvantages 1](#_Toc3475747)

[Where to store packages? 1](#_Toc3475748)

[Where does the SSIS take resources (Memory and CPU) – It Depends 2](#_Toc3475749)

[Where will the job execution information stored? 3](#_Toc3475750)

[DEMO: How to run the same SSIS code in several environments. 4](#_Toc3475751)

[DEMO: How to create a log table 17](#_Toc3475752)

[Appendix A 20](#_Toc3475753)

[OnError 20](#_Toc3475754)

[OnInformation 21](#_Toc3475755)

[OnPreExec 22](#_Toc3475756)

[end 22](#_Toc3475757)

[OnPostExec 23](#_Toc3475758)

# Centralize SSIS

## Advantages

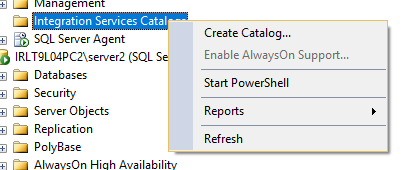
* + 1. Central management location
    2. Easily scale packages by adding more memory to the central server
    3. Keep the ETL workload away from other databases
    4. Standardize and control multiple SQL instances using the same SSIS package

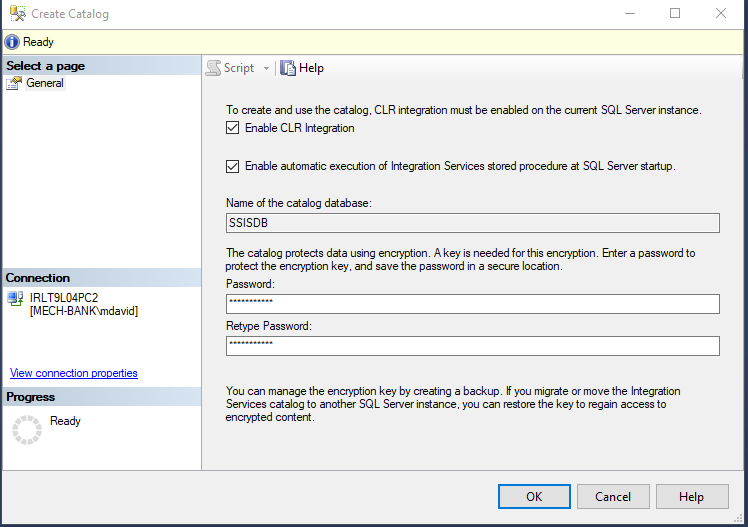
## Disadvantages

* + 1. Must license the SQL Server
    2. Remote data must be copied over the network before entering the Data flow buffer.

# Where to store packages?

* + In database – Integration Services Catalogs
  + To create the catalog right click on Integration Services Catalog/Create Catalog



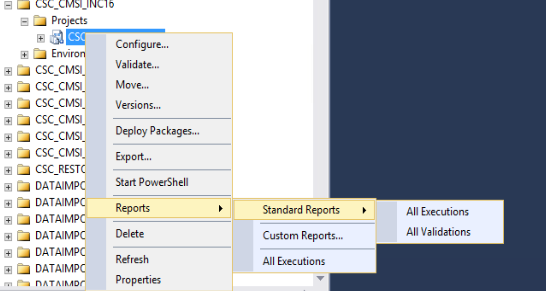


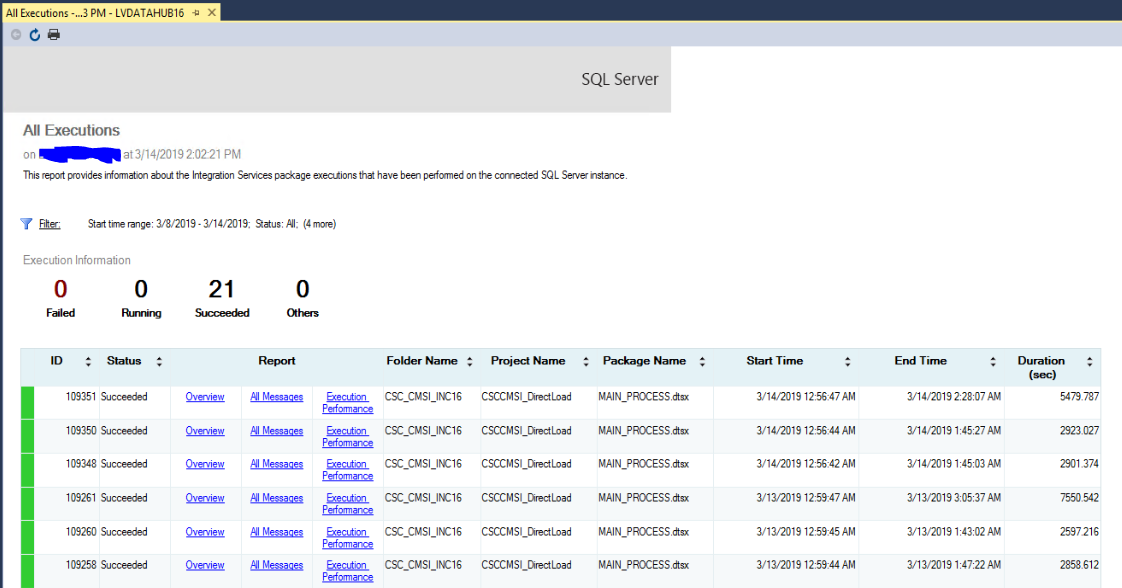
# Where does the SSIS take resources (Memory and CPU) – It Depends

* + Job A created on Server A but points to Server B to pull data
    1. The job is kicked off on Server A using Server A resources
    2. The select statement will run on Server B and the data will pull across the network to Server A for further analysis or transformation
  + Job A created on Server A but points to Server B to run backup
    1. Job A will kick off on Server A using Server A resources
    2. However the backup will occur on Server B using Server B resources

# Where will the job execution information stored?

* + On the centralized server.
  + In order to review the execution, right click on the project, select reports/Standard Reports/All Executions

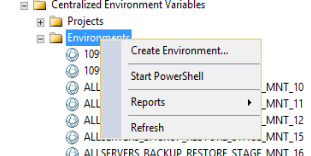




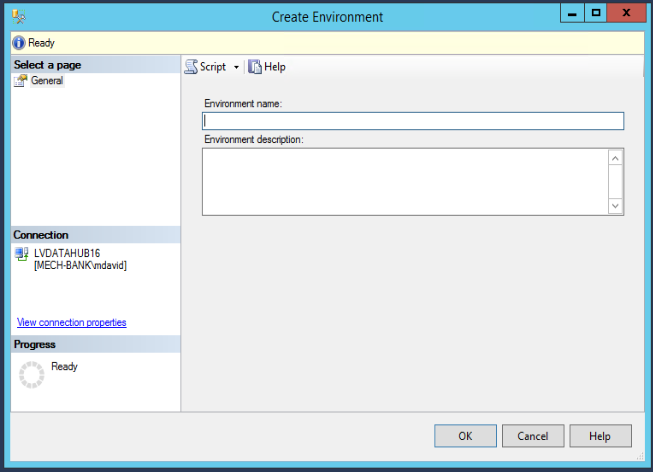
# DEMO: How to run the same SSIS code in several environments.

* + In this demonstration, we will run the same SSIS in two separate environments, without changing the SSIS code.
  + Permissions: The Service Account running SQL Agent must have permissions on the target server to execute in the server or database what the SSIS code is executing.
  + Permissions tip: I run my master Server/SQL Agent and all target Servers/SQL Agents under the same Service Account.

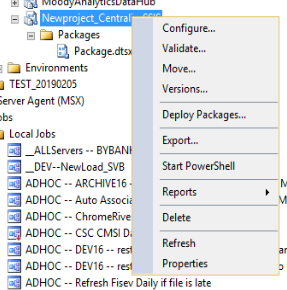
1. Create your SSIS package and deploy to the centralized server
2. Create a folder for the Environment Variables:
   1. Under “Integration Services Catalogs”, create a folder for your environment variables; or under the package, create an environment variable.
3. Add an Environment variable by right clicking on “Environments”



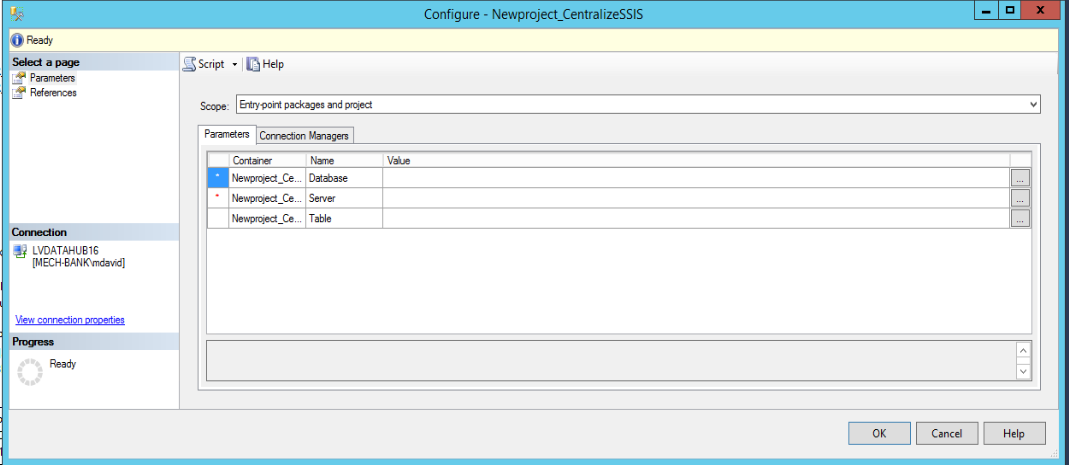
1. Select Create Environment…



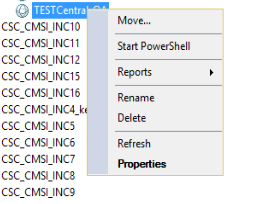
1. Type an Environment Name and then click the OK button.
2. Go to the project and then right click. Select “Configure…”



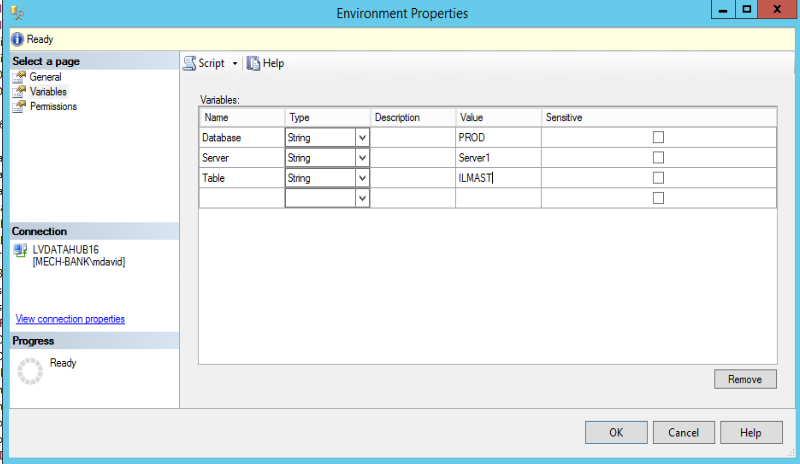
1. Make a screen shot of the parameters needed.



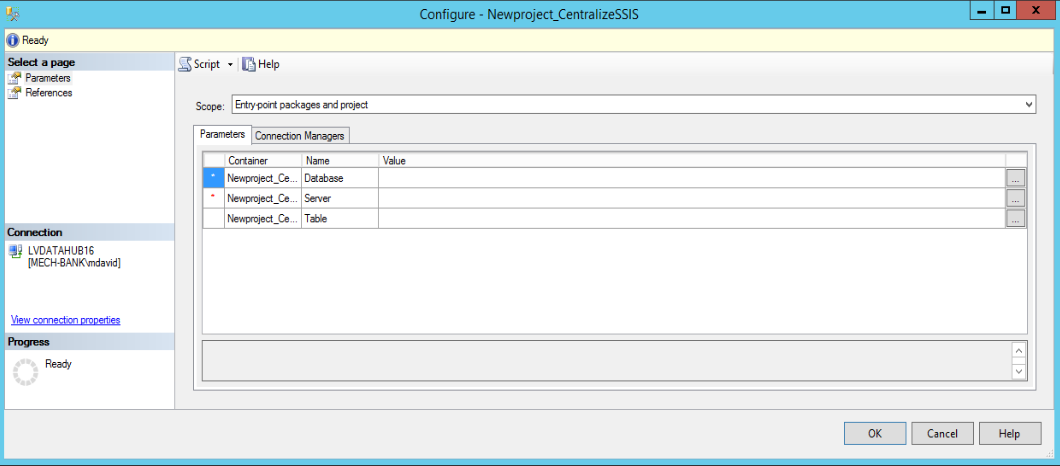
1. Find your new Environment variable and right click, select properties.



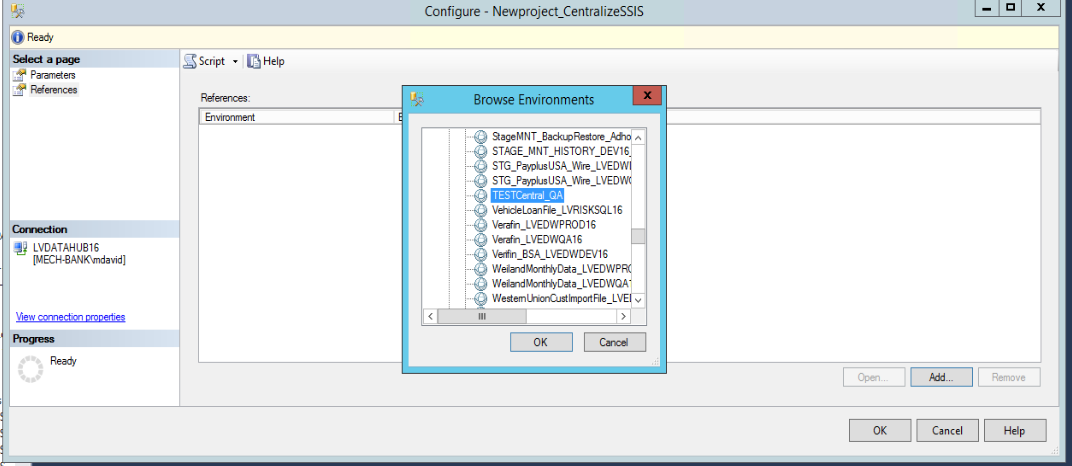
1. Type in the parameter names from the screen shot into the variable names on the environment variable.



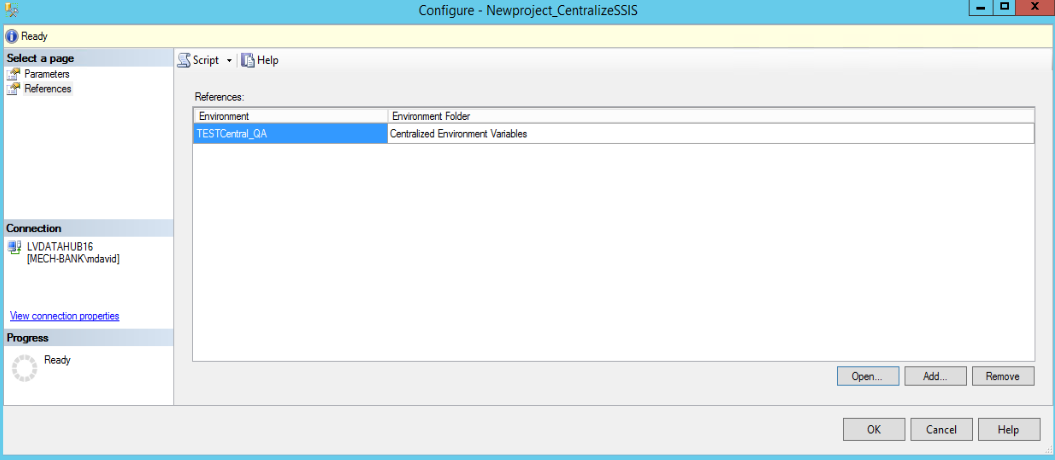
1. Type in the values you want for this Environment; click OK
2. Right Click on the project again



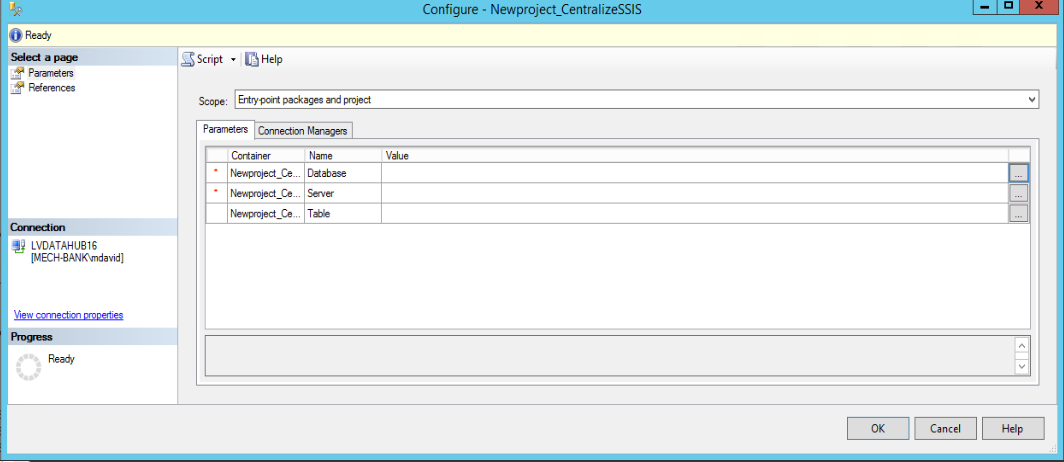
1. Now we are going to assign the parameters to the environment variables.
2. Select References
3. Click Add…



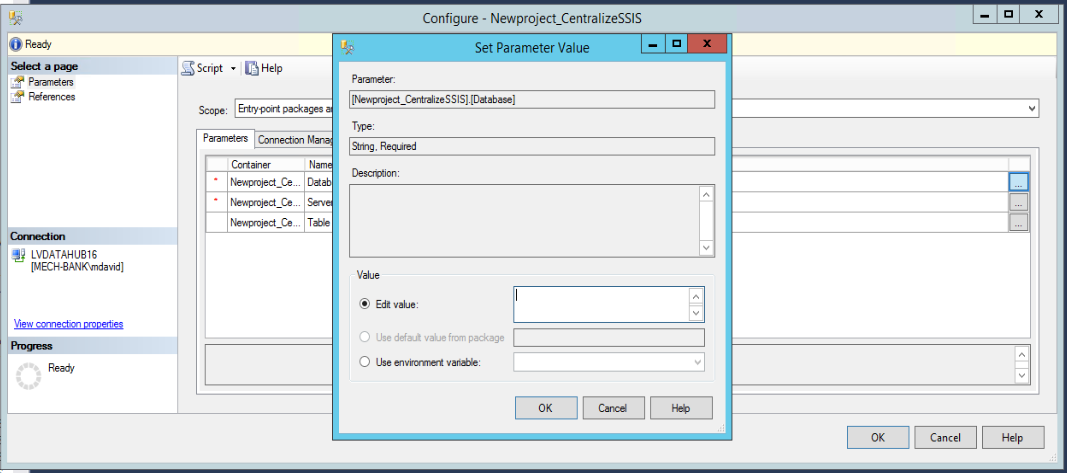
1. Select the Environment variable you created and select OK



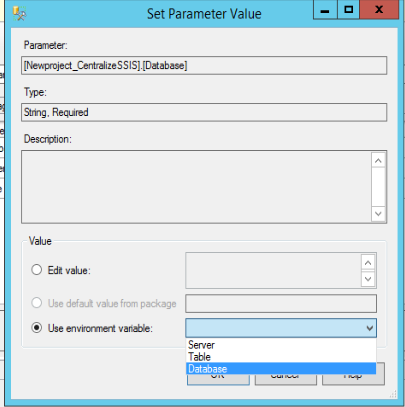
1. Select Parameters from the left menu



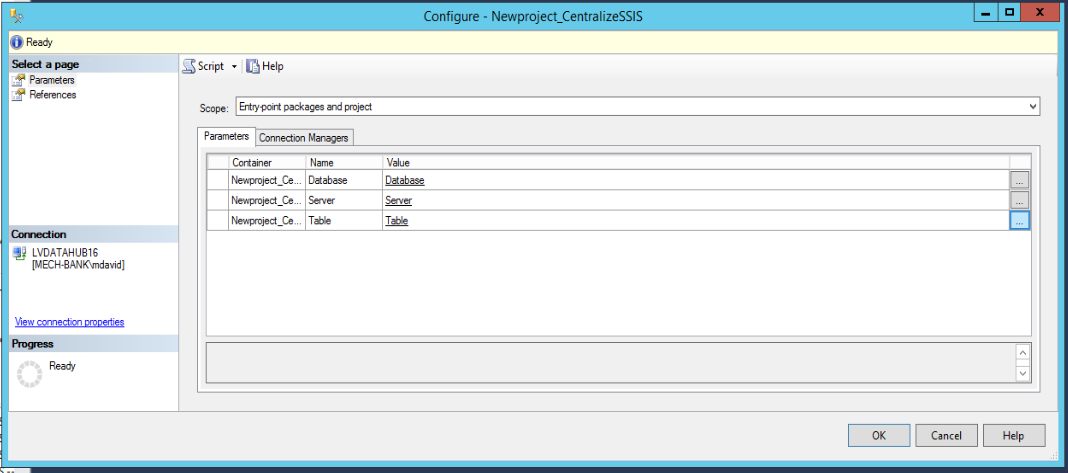
1. Click on the ellipses next to the first parameter name



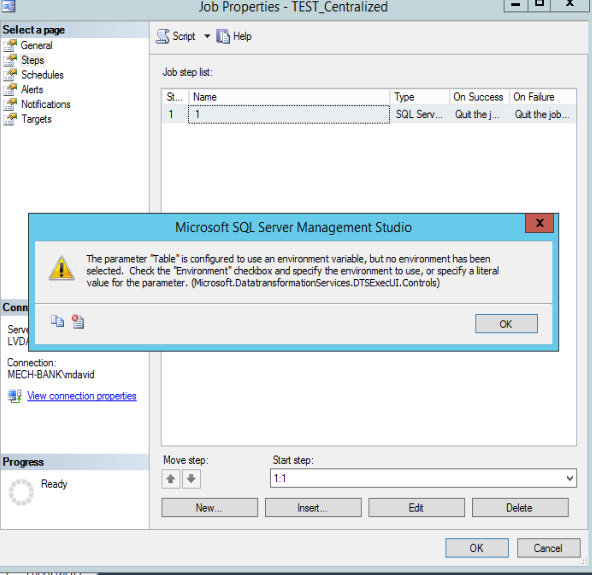
1. Select “Use Environment Variable:” and select the variable that ties to this parameter.



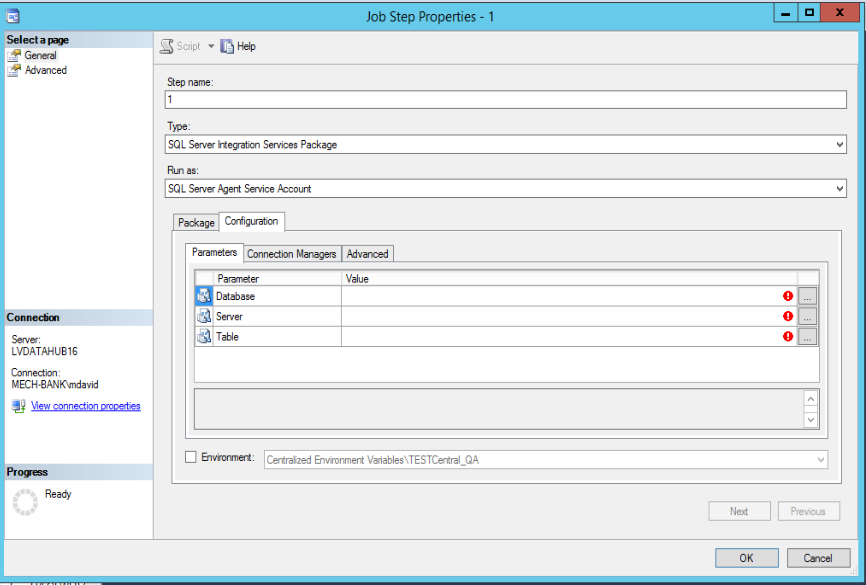
1. Click OK and notice the value is underlined. The Underline means the value is now linked to the environment variable.
2. Click the ellipses for each name and tie it to a variable. When finished assigning variables, click OK.



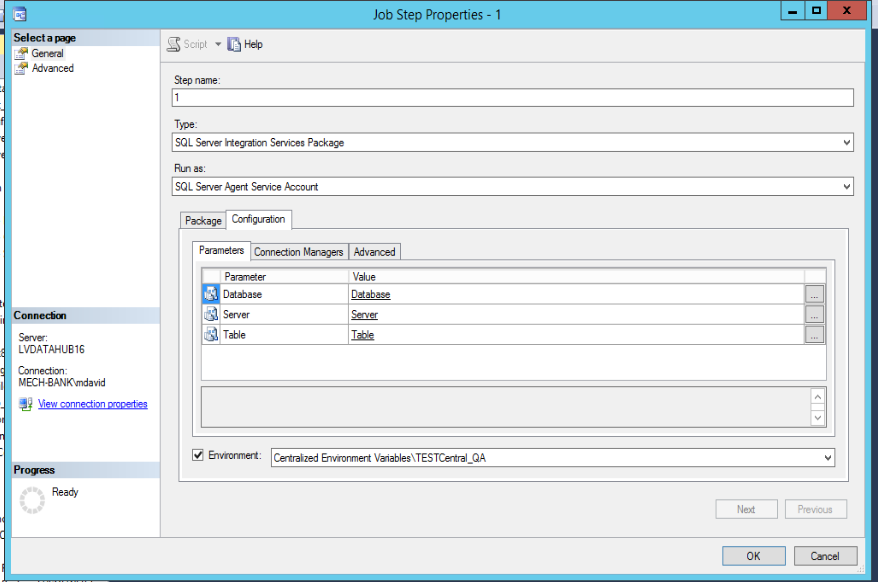
1. Go back to the job you created. Right Click, select properties, and then select steps and edit the SSIS step we created.



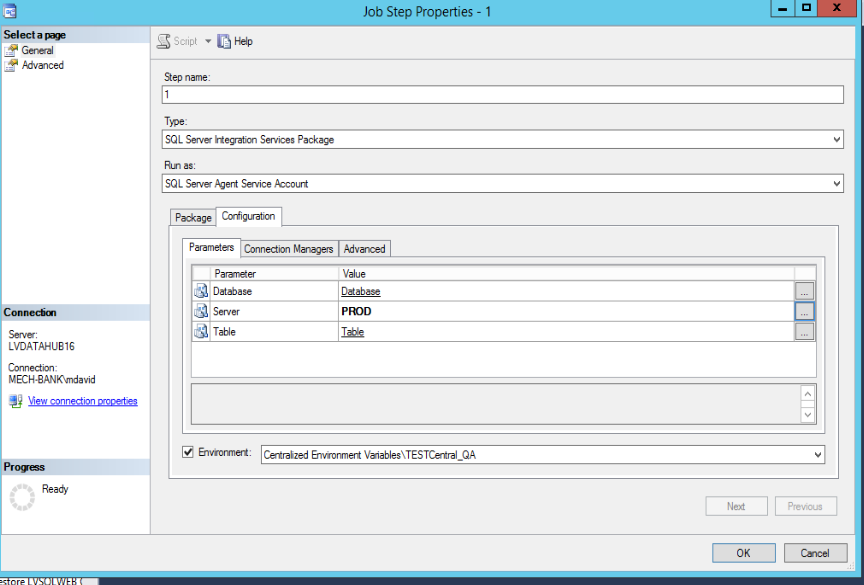
1. You will receive a warning, click OK.
2. The Job step properties box will open. Select the Configuration tab.



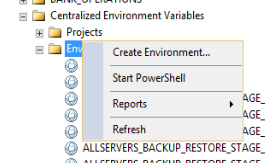
1. Check the Environment box and select the centralized environment variable you want to use.



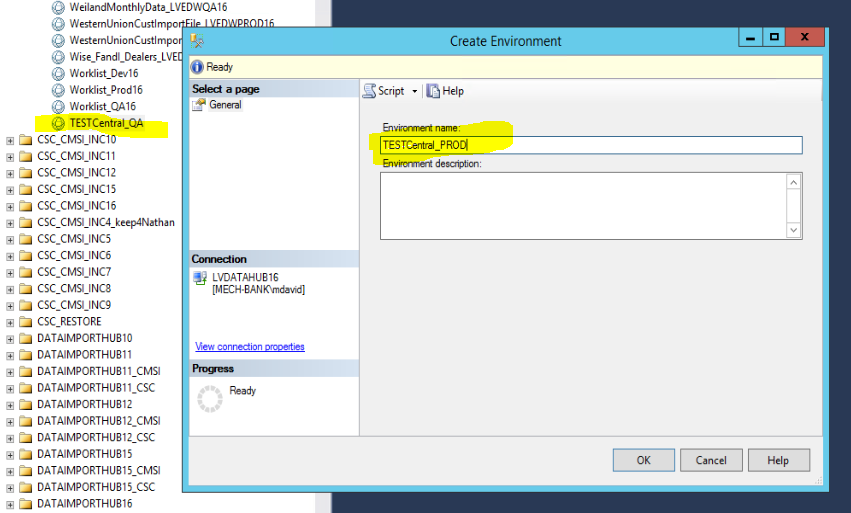
1. Notice the values from the environment are shown and they are underlined.
2. NOTE: When someone changes a variable in the job, instead of the environment variable, the underline will change to a value without an underline. The only way to switch back to an environment variable is to recreate the job.



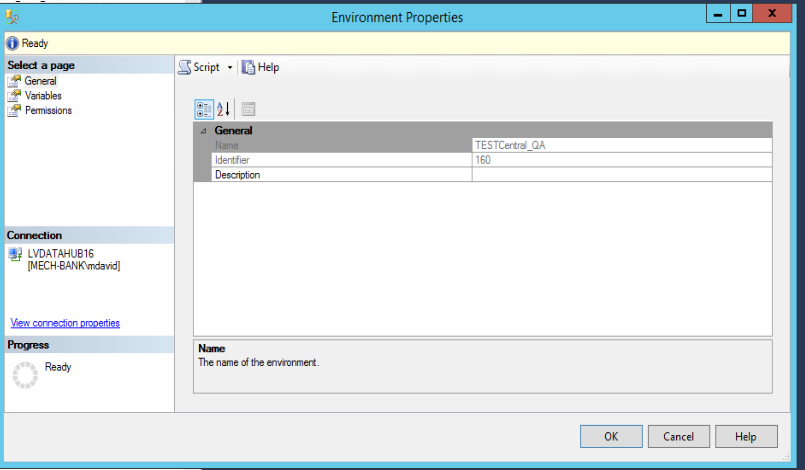
1. Do Not change the variables in the job, instead change the environment variables or create a new environment variable to use.
2. The first environment variable we used was for QA, now I want to move into Prod. Just create a new environment variable and reference in the package; and then select in the job. Let’s do this.
3. Create a New Environment Variable. Go to the directory where you stored the QA environment variable. Right click and create a new variable.

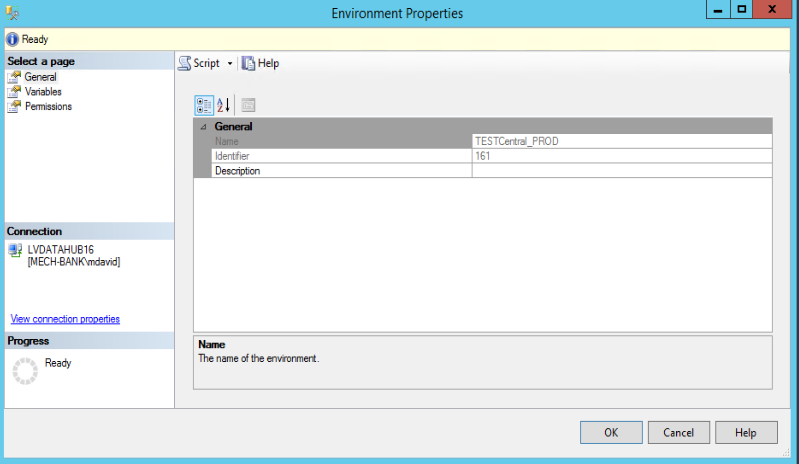


1. I keep the same naming scheme for each environment variable that I am linking to my TestCentral package. Keeping the same name at the front, helps me find the variable when I need to link it in the job.



1. Click OK to create the environment variable.
2. We can copy the variables from the QA variable to the PROD variable. This is helpful when there are too many variables to type. Be careful to copy QA to PROD and not PROD to QA, because PROD is blank and if copied to QA then QA will become blank.





1. Below is the code to copy QA into PROD

INSERT INTO [SSISDB].[internal].[environment\_variables]

([environment\_id]

,[name]

,[description]

,[type]

,[sensitive]

,[value]

,[sensitive\_value]

,[base\_data\_type])

SELECT 161 as environment\_id --New Environment ID

,[name]

,[description]

,[type]

,[sensitive]

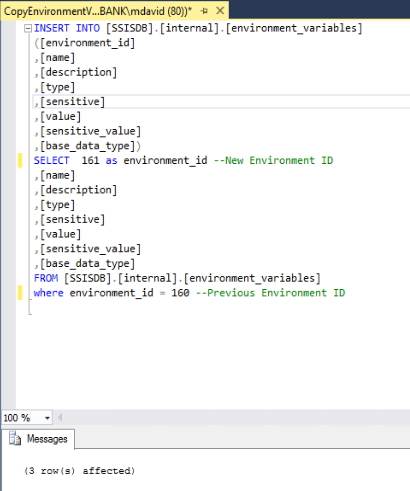
,[value]

,[sensitive\_value]

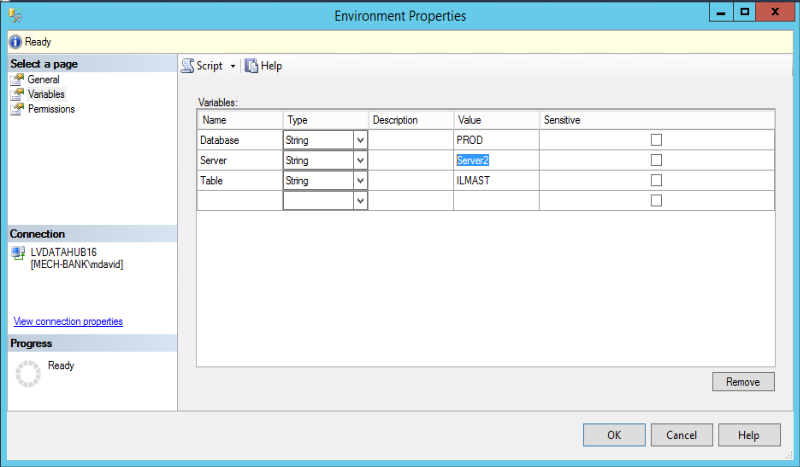
,[base\_data\_type]

FROM [SSISDB].[internal].[environment\_variables]

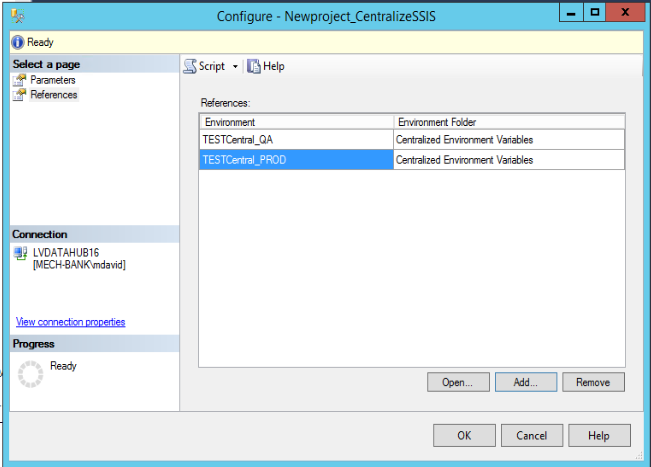
where environment\_id = 160 --Previous Environment ID



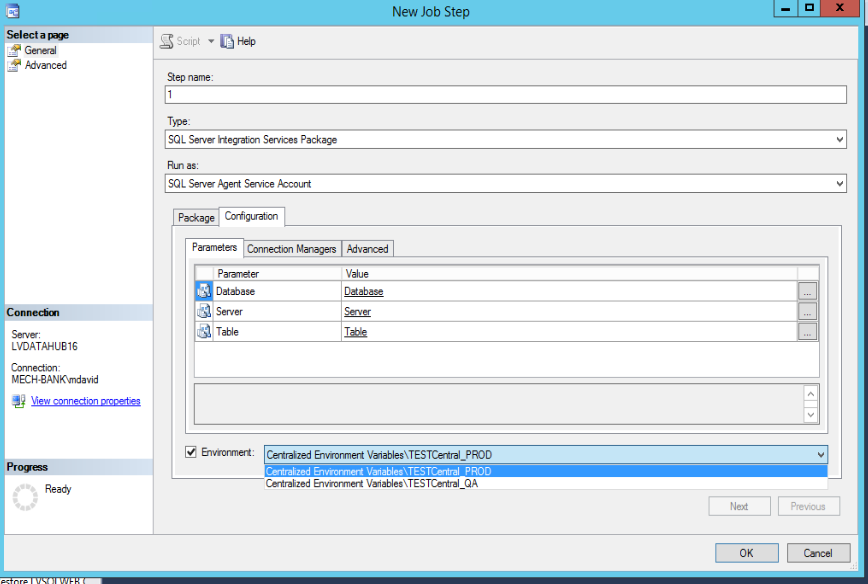
1. Verify and change the values for the new environment variables.



1. Link the new PROD Environment variable to the Package; and then click OK.



1. Either change the job step and select the new PROD environment variable or create a new job to point to PROD. I am creating a new job for Production.



# DEMO: How to create a log table

* 1. Create a database to store the centralized logs, procedures, functions, etc…
  2. Decide what you want to log.
  3. Create a table to store all the variables you would like to log.

CREATE TABLE [dbo].[ProcessStatusLog](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Notification\_Code] [nvarchar](100) NULL,

[Type] [char](10) NOT NULL,

[ServerName] [nvarchar](100) NOT NULL,

[DBName] [nvarchar](100) NOT NULL,

[JobPackageName] [nvarchar](100) NOT NULL,

[ExecutionPath] [nvarchar](500) NOT NULL,

[TaskName] [nvarchar](100) NOT NULL,

[EventName] [nvarchar](100) NOT NULL,

[ErrorMessage] [nvarchar](max) NULL,

[Note] [nvarchar](max) NULL,

[EntryTimeStamp] [datetime] NOT NULL,

CONSTRAINT [PK\_ProcessStatusLog] PRIMARY KEY CLUSTERED

(

[ID] ASC

) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

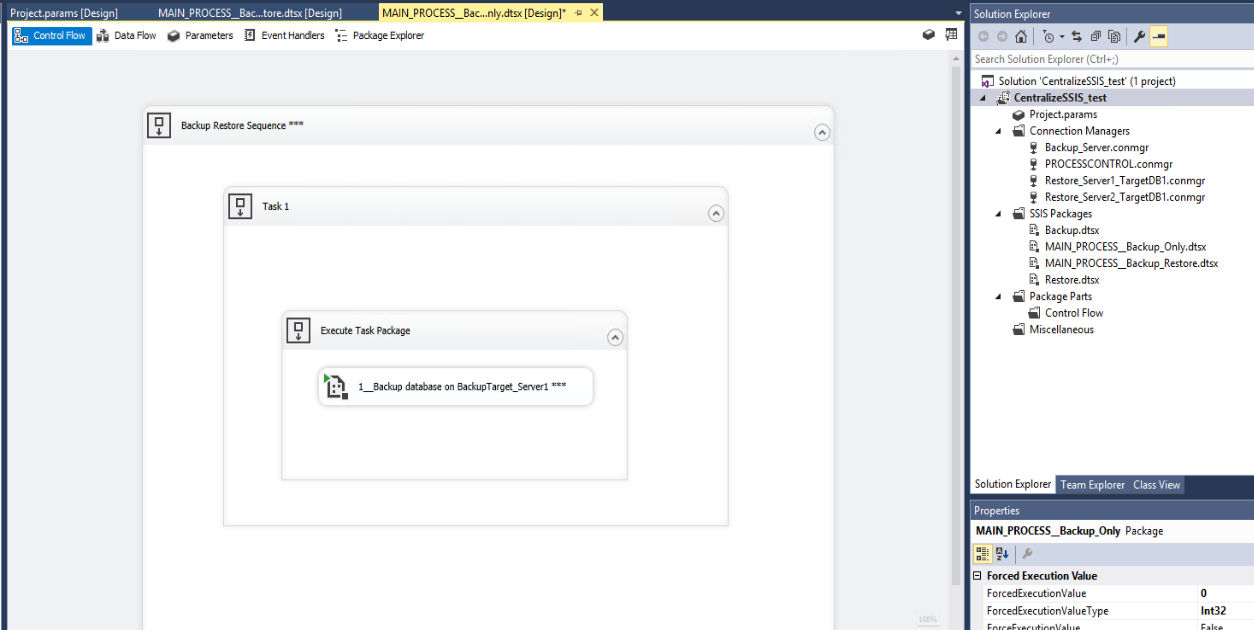
ALTER TABLE [dbo].[ProcessStatusLog] ADD DEFAULT (getdate()) FOR [EntryTimeStamp]

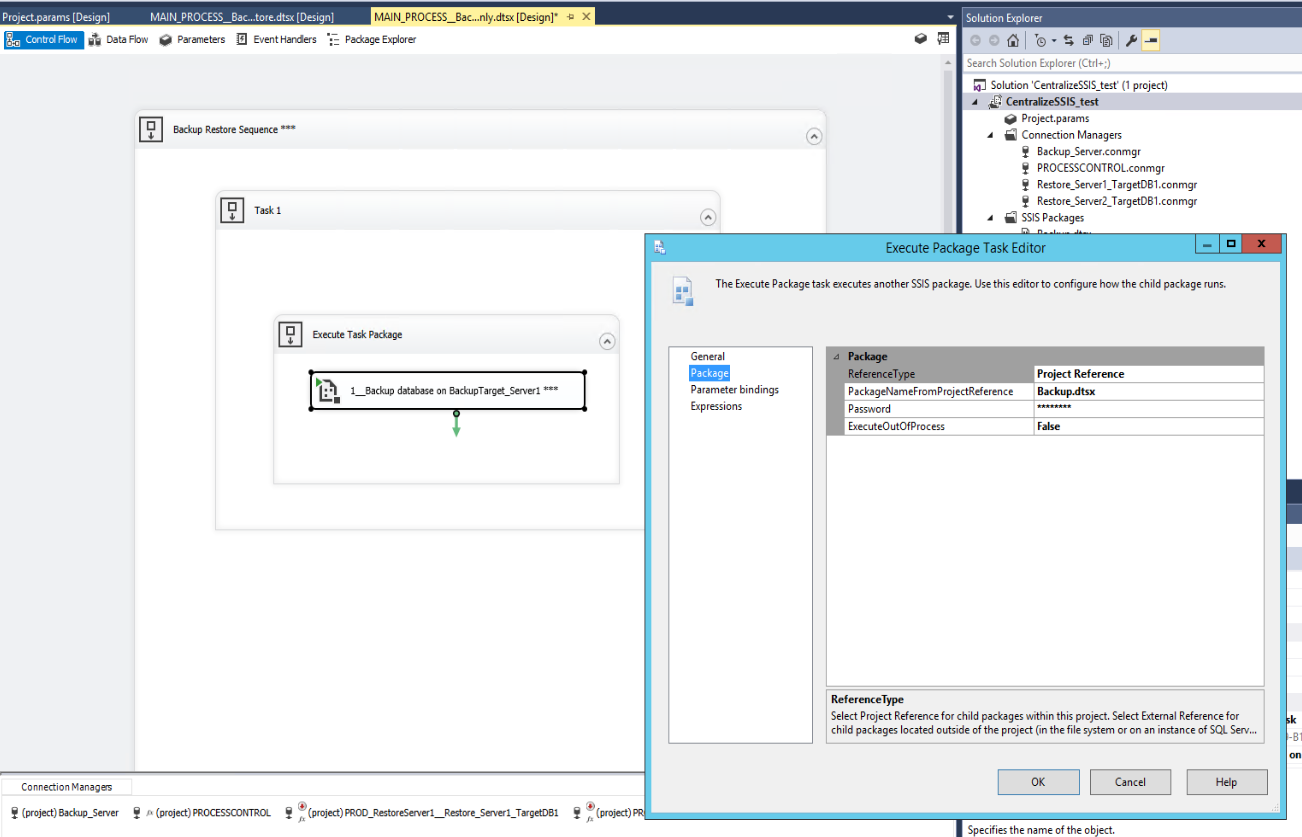
GO

* 1. In the SSIS Project, create a main template with event handlers.
  2. Each event handler will call a procedure in the database you created to store the logs.
  3. The Procedure will insert a record into the process log table.

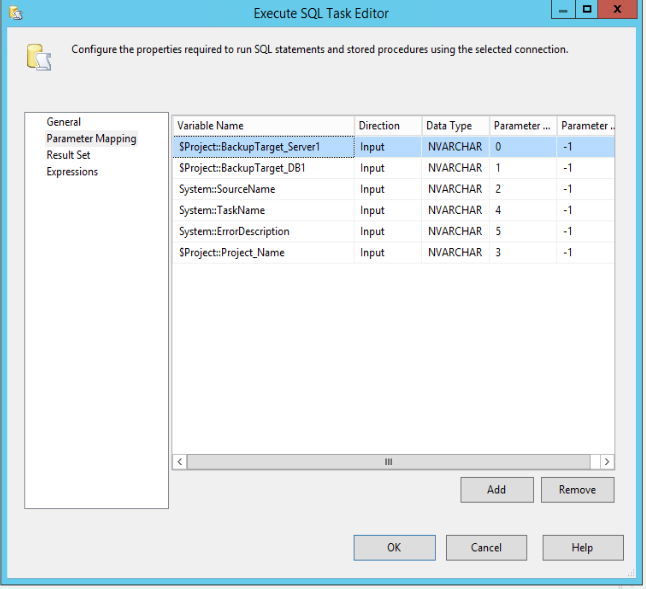
See Procedures in Appendix A

* Demo
  1. In your SSIS package, create a main template.
  2. In the template, call the package from the main package.





* 1. In the Main package, add the following event handlers.
     + OnError
     + OnInformation
     + OnPostExecute
     + OnPreExecute
     + OnWarning
  2. Why each event handler is used?
     + OnError - When the calling package fails the error will bubble up to the main template and use the onError EventHandler.
     + OnInformation – This event is raised during the validation and execution of an executable to report information.
     + OnPostExecute – Executed before the package ends
     + OnPreExecute – Executed before the package starts
     + OnWarning – When the calling package has a warning, the warning will bubble up to the main template and use the OnWarning EventHandler.
  3. Configure the Each event handler to execute a SQL Task from the centralized log database you created. The SQL Task will be a procedure stored in that database.
     + OnError – exec [dbo].[OnError2] ?, ?, ?, ?, ?, ?
     + OnInformation - exec [dbo].[OnInformation2] ?, ?, ?, ?, ?, ?
     + OnPostExecute - exec [dbo].[OnPostExec2] ?, ?, ?, ?, ?, ?
     + OnPreExecute - exec [dbo].[OnPreExec2] ?, ?, ?, ?, ?, ?
     + OnWarning - exec [dbo].[OnInformation2] ?, ?, ?, ?, ?, ?
  4. Map the ? parameters to a project or system variable.



# Appendix A

## OnError

Create procedure [dbo].[OnError2]

@serverName nvarchar(100),

@DBName nvarchar(100),

@JobPackageName nvarchar(100),

@ProjectName nvarchar(500),

@TaskName nvarchar(100),

@ErrorMessage nvarchar(max)

as

begin

declare @EventName nvarchar(100), @Notification\_Code nvarchar(100)

select @EventName = 'OnError'

select @Notification\_Code = rtrim(@ProjectName) + '\_' + @EventName

INSERT INTO [dbo].[ProcessStatusLog]

([Type]

,[Notification\_Code]

,[ServerName]

,[DBName]

,[JobPackageName]

,[ExecutionPath]

,[TaskName]

,[EventName]

,[ErrorMessage]

--,[EntryTimeStamp]

)

VALUES

('SSIS' --<Type, char(10),>

,@Notification\_Code

,@serverName --<ServerName, nvarchar(100),>

,@DBName --<DBName, nvarchar(100),>

,@JobPackageName --<JobPackageName, nvarchar(100),>

,@ProjectName --<ExecutionPath, nvarchar(500),>

,@TaskName --<TaskName, nvarchar(100),>

,@EventName --<EventName, nvarchar(100),>

,@ErrorMessage --<ErrorMessage, nvarchar(max),>

--,<EntryTimeStamp, datetime,>

)

end

## OnInformation

create procedure [dbo].[OnInformation2]

@serverName nvarchar(100),

@DBName nvarchar(100),

@JobPackageName nvarchar(100),

@ProjectName nvarchar(500),

@TaskName nvarchar(100),

@Note nvarchar(max)

as

begin

if @JobPackageName like '%\*\*\*' or @TaskName like '%\*\*\*' or @JobPackageName = 'MAIN\_PROCESS'

begin

declare @EventName nvarchar(100), @Notification\_Code nvarchar(100)

select @EventName = 'OnInformation', @JobPackageName = replace(@JobPackageName,'\*\*\*',''), @TaskName = replace(@TaskName,'\*\*\*','')

select @Notification\_Code = rtrim(@ProjectName) + '\_' + @EventName

INSERT INTO [dbo].[ProcessStatusLog]

([Type]

,[Notification\_Code]

,[ServerName]

,[DBName]

,[JobPackageName]

,[ExecutionPath]

,[TaskName]

,[EventName]

,[Note]

--,[EntryTimeStamp]

)

VALUES

('SSIS' --<Type, char(10),>

,@Notification\_Code

,@serverName --<ServerName, nvarchar(100),>

,@DBName --<DBName, nvarchar(100),>

,@JobPackageName --<JobPackageName, nvarchar(100),>

,@ProjectName --<ExecutionPath, nvarchar(500),>

,@TaskName --<TaskName, nvarchar(100),>

,@EventName --<EventName, nvarchar(100),>

,@Note --<ErrorMessage, nvarchar(max),>

--,<EntryTimeStamp, datetime,>

)

end

end

## OnPreExec

CREATE procedure [dbo].[OnPreExec2]

@serverName nvarchar(100),

@DBName nvarchar(100),

@JobPackageName nvarchar(100),

@ProjectName nvarchar(500),

@TaskName nvarchar(100),

@Note nvarchar(max)

as

begin

if @JobPackageName like '%\*\*\*' or @TaskName like '%\*\*\*' or @JobPackageName = 'MAIN\_PROCESS'

begin

declare @EventName nvarchar(100), @Notification\_Code nvarchar(100)

select @EventName = 'OnPreExec', @JobPackageName = replace(@JobPackageName, '\*\*\*', ''), @TaskName = replace(@TaskName, '\*\*\*','')

select @Notification\_Code = rtrim(@ProjectName) + '\_' + @EventName

INSERT INTO [dbo].[ProcessStatusLog]

([Type]

,[Notification\_Code]

,[ServerName]

,[DBName]

,[JobPackageName]

,[ExecutionPath]

,[TaskName]

,[EventName]

,[ErrorMessage]

,[Note]

--,[EntryTimeStamp]

)

VALUES

('SSIS' --<Type, char(10),>

,@Notification\_Code

,@serverName --<ServerName, nvarchar(100),>

,@DBName --<DBName, nvarchar(100),>

,@JobPackageName --<JobPackageName, nvarchar(100),>

,@ProjectName --<ExecutionPath, nvarchar(500),>

,@TaskName --<TaskName, nvarchar(100),>

,@EventName --<EventName, nvarchar(100),>

,'' --<ErrorMessage, nvarchar(max),>

, @Note

--,<EntryTimeStamp, datetime,>

)

end

## end

## OnPostExec

CREATE procedure [dbo].[OnPostExec2]

@serverName nvarchar(100),

@DBName nvarchar(100),

@JobPackageName nvarchar(100),

@ProjectName nvarchar(500),

@TaskName nvarchar(100),

@Note nvarchar(max)

as

begin

if @JobPackageName like '%\*\*\*' or @TaskName like '%\*\*\*' or @JobPackageName = 'MAIN\_PROCESS'

begin

declare @EventName nvarchar(100), @Notification\_Code nvarchar(100)

select @EventName = 'OnPostExec', @JobPackageName = replace(@JobPackageName,'\*\*\*',''), @TaskName = replace(@TaskName,'\*\*\*','')

select @Notification\_Code = rtrim(@ProjectName) + '\_' + @EventName

INSERT INTO [dbo].[ProcessStatusLog]

([Type]

,[Notification\_Code]

,[ServerName]

,[DBName]

,[JobPackageName]

,[ExecutionPath]

,[TaskName]

,[EventName]

,[ErrorMessage]

,[Note]

--,[EntryTimeStamp]

)

VALUES

('SSIS' --<Type, char(10),>

,@Notification\_Code

,@serverName --<ServerName, nvarchar(100),>

,@DBName --<DBName, nvarchar(100),>

,@JobPackageName --<JobPackageName, nvarchar(100),>

,@ProjectName --<ExecutionPath, nvarchar(500),>

,@TaskName --<TaskName, nvarchar(100),>

,@EventName --<EventName, nvarchar(100),>

,'' --<ErrorMessage, nvarchar(max),>

,@Note

--,<EntryTimeStamp, datetime,>

)

end

end

GO