Updating Deltas to Amazon RDS from on-premises MS SQL Server

By Tabish Sayed

In my previous blog post I discussed about few options to migrate our on-premises databases to Amazon RDS. In this blog we will move on to next step after migration. For those who have not read my previous blog I recommend going back and reading it here [Migration Process of On-Prem Databases to Amazon RDS](https://ipd.informafi.com/blogs/337-migration-process-of-on-prem-databases-to-amazon-rds).

So, you have successfully moved all your required databases from on-premises SQL Server to your Amazon RDS service, and you have also set up your applications (web or client/server) to communicate with these RDS Instance. What is next, the “deltas”. Please keep in mind that this blog is not about how we could setup our corporate policies, as to how we are going to expose our RDS instances to our local or public network. The main target of this blog post is to how we can get our deltas (the incremental changes on our on-prem databases) to our Amazon RDS instance.

Case Background:

I have multiple on-prem databases which are OLTP in nature. Typically, these databases are updated throughout the day and the tables may have gone thru many CRUD operations multiple times in a day. There is heavy analysis, reporting, and entitlements are involved on this data, so we decided to move the data to another database server which we call reporting databases. These reporting databases are moved on Amazon RDS, so the data can be served with all the goodies that Amazon RDS offers.

In our case, RDS Instance sits behind the load balancer with 2 instances. Now the big question is how do we update these database with new changes, the deltas. There are many ways to get deltas to RDS, my preferred way is using import/export utility from MS SQL Server along with integration services. To simplify my approach, I have divided my delta transfer process into 3 easy steps:

Step 1: Preparing Deltas on Source Database:

My main target in this step was to recognize the changes in the relevant production tables and create the new delta tables to bring to RDS. These delta tables could be thinner and shorter, meaning less columns and filtered rows from its predecessors. For simplicity, I’m going to talk only about one database, let’s call it **SalesDB**. I have created a stored procedure which creates the fresh delta table. Please note that this is just a sample script, you will have to create similar tables as per your own requirements.

USE SalesDB

GO

Create Procedure [dbo].[RDS\_SalesDB\_Prepare\_Deltas]

as

BEGIN

set nocount on

BEGIN TRY

----------------------------------------

-- Start Preparing Deltas for Customers

----------------------------------------

if not exists(select \* from sysobjects where name = 'zzzCustomers' and type = 'U')

begin

--drop table zzzCustomers

create table [dbo].[zzzCustomers](

[CustID] [int] null,

[CustName] [varchar](250) null,

[LastUpdated] [datetime] null default getdate()

)

end

else

begin

truncate table zzzCustomers

end

--The below data is the intended data to bring to RDS

Insert Into zzzCustomers

select

CustID

,CustName

,LastUpdated

from Customers with (nolock)

-------------------------

--Finish Preparing Deltas

-------------------------

END TRY

BEGIN CATCH

declare @ErrorSeverity int, @ErrorState int, @ErrorMessage nvarchar(4000);

select

@Errorseverity = error\_severity(),

@Errorstate = error\_state(),

@ErrorMessage = error\_procedure() + ': line#' + cast(error\_line() as varchar)

+ '-' + error\_message();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

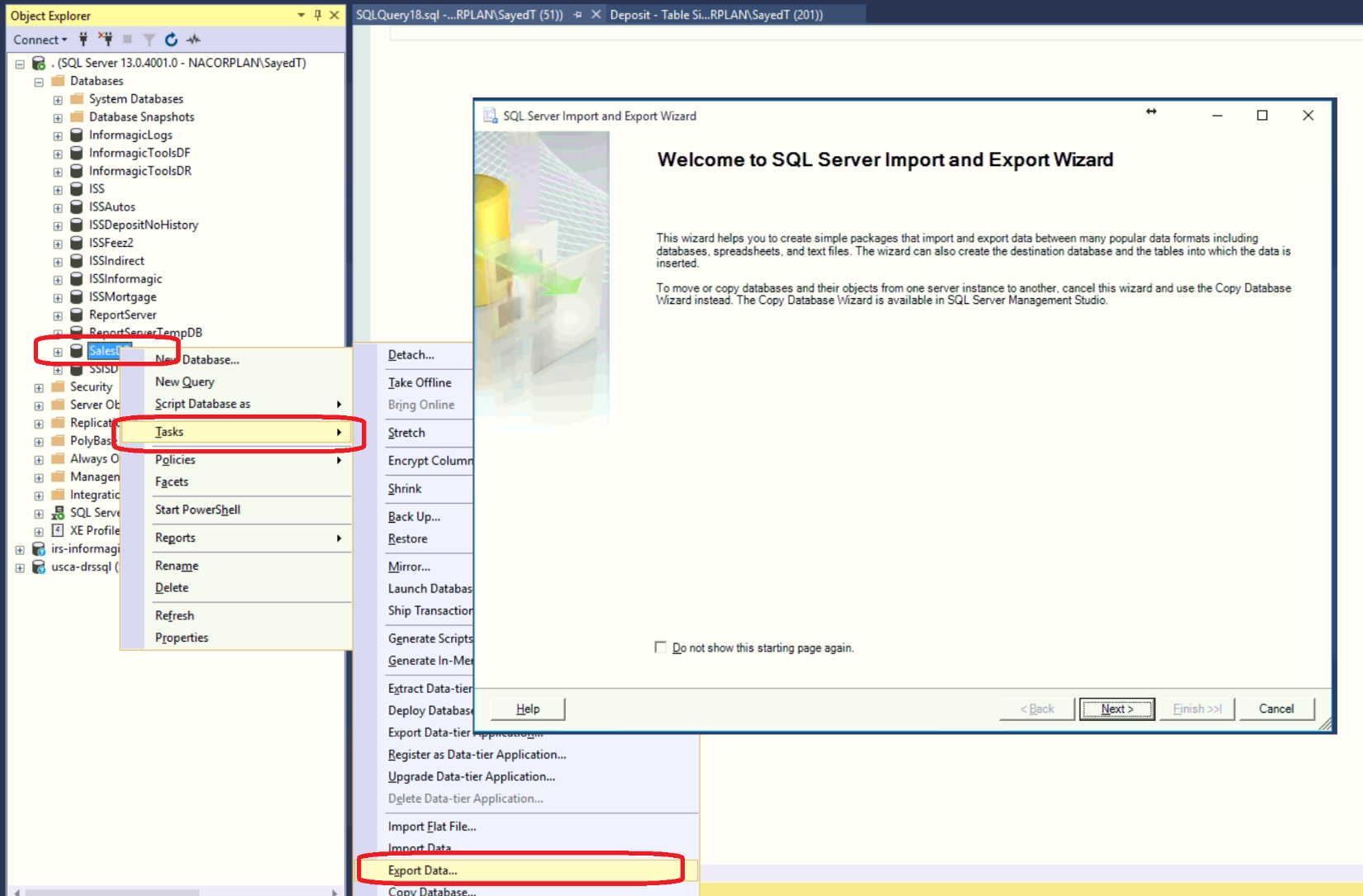
END CATCH

END

Go

Step 2: Push Deltas to RDS Instance using SSIS Package:

Create the SSIS package using SSMS to export all the delta tables you created in Step 1 and exported to RDS Endpoint. How to create SSIS package is beyond the scope of this blog. You can find tons of articles and videos on the web for this reason. Following is a link of simple example by Ray Barley which exports from SQL Server to Excel spreadsheet. <https://www.mssqltips.com/sqlservertutorial/202/simple-way-to-export-data-from-sql-server/>. The following image shows the starting point for creating SSIS package:



In our case, we would be exporting from SQL Server to SQL Server. However, just remember to take care of your identity columns (auto increment columns) by turning on the checkbox for “Enable Identity Insert On” and in my case I also use “Drop and Recreate Destination Table” at the time of transformation. Test your package to confirm that the package transfers the relevant tables to the destination RDS Instance properly.

In our case I prefixed delta tables with “zzz” and take all the zzzTables to RDS. Once the zzzTables are available on destination RDS move to step 3.

Step 3: Restore Deltas into RDS Instance:

In this step, you will create the stored procedure on RDS database which will restore the zzzTables to its actual destination tables. My procedure is as below:

USE [SalesReportingDB]

GO

Create Procedure [dbo].[RDS\_SalesReportingDB\_Restore\_Deltas]

as

Begin

set nocount on

BEGIN TRY

---------------------------------------

-- Refresh Customers from zzzCustomers

---------------------------------------

BEGIN TRAN

delete Customers

insert into Customers(CustID, CustName, LastUpdated)

select

CustID

,CustName

,LastUpdated

from zzzCustomers WITH (TABLOCKX)

if (@@error <> 0)

begin

raiserror('Delta Updates for Customers Failed...', 16, 1)

end

COMMIT TRAN;

END TRY

BEGIN CATCH

ROLLBACK TRAN;

declare @ErrorSeverity int, @ErrorState int, @ErrorMessage nvarchar(4000);

select

@Errorseverity = error\_severity(),

@Errorstate = error\_state(),

@ErrorMessage = error\_procedure() + ': line#' + cast(error\_line() as varchar)

+ ' - ' + error\_message();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

End

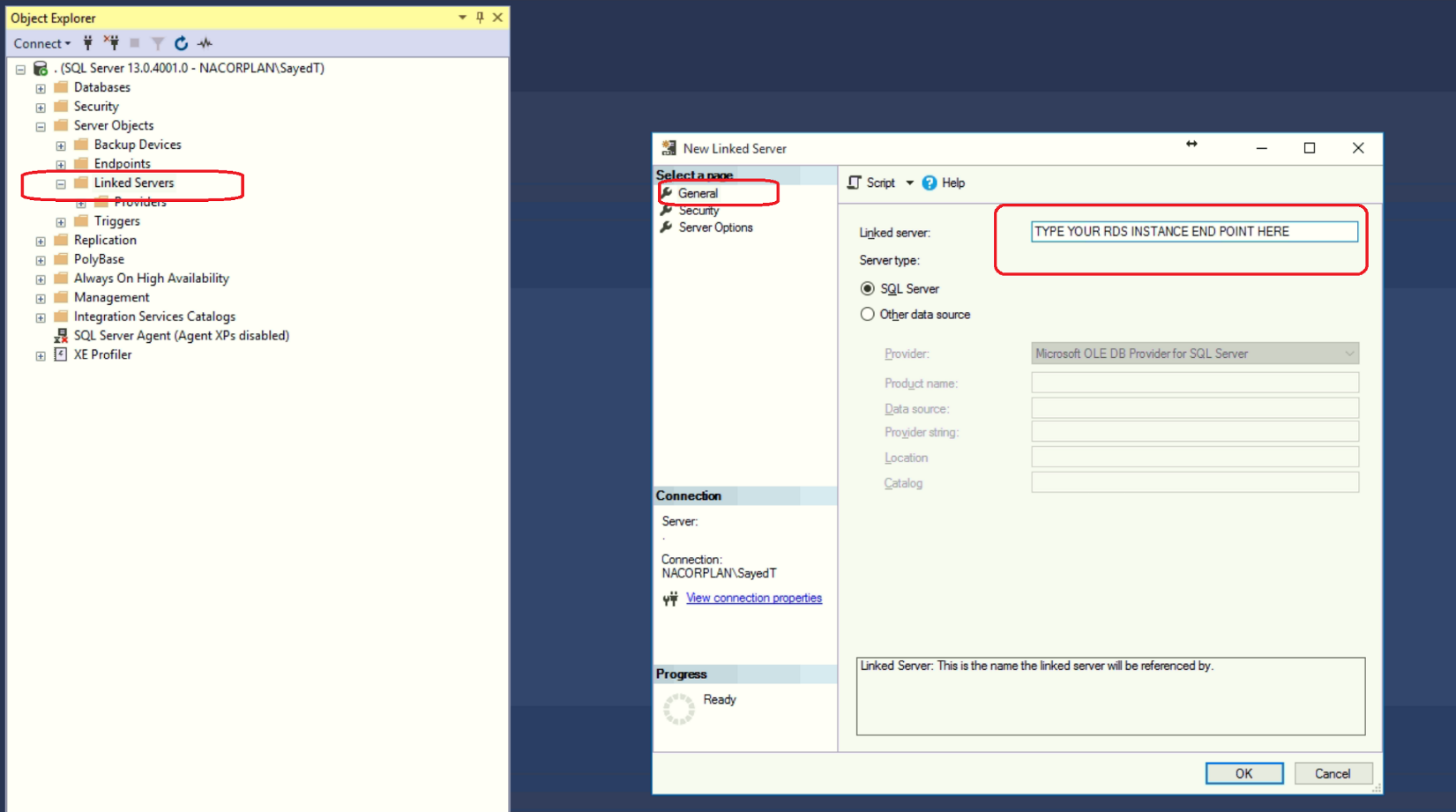
go

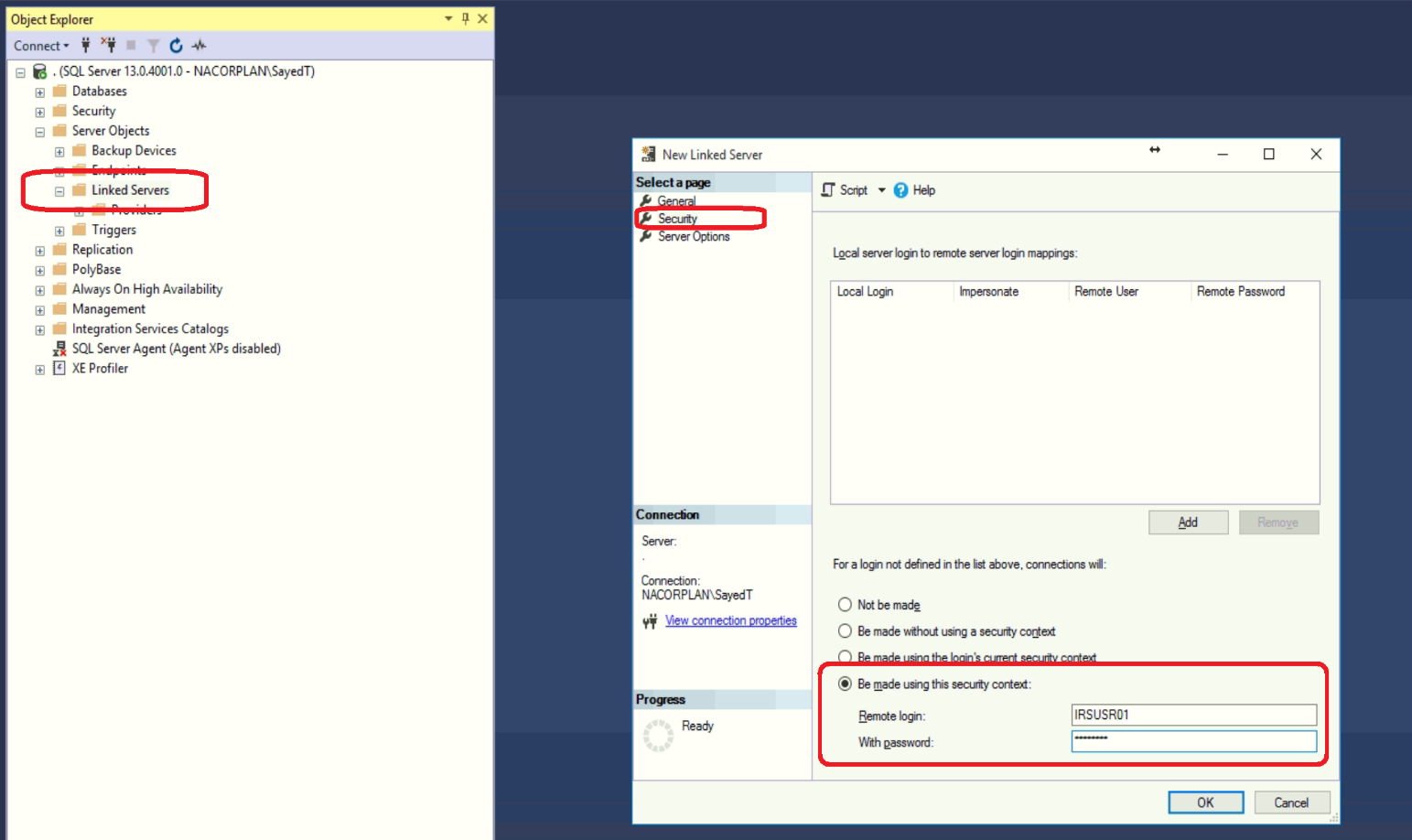
Executing the above procedure will bring the data in sync with production, but the ideal situation would be to schedule this procedure to execute once the deltas (zzzTables) are transferred to RDS. Let’s see how to do that in the next step.

Step 4 (Optional): Create and Schedule the above 3 steps as a SQL Server Job:

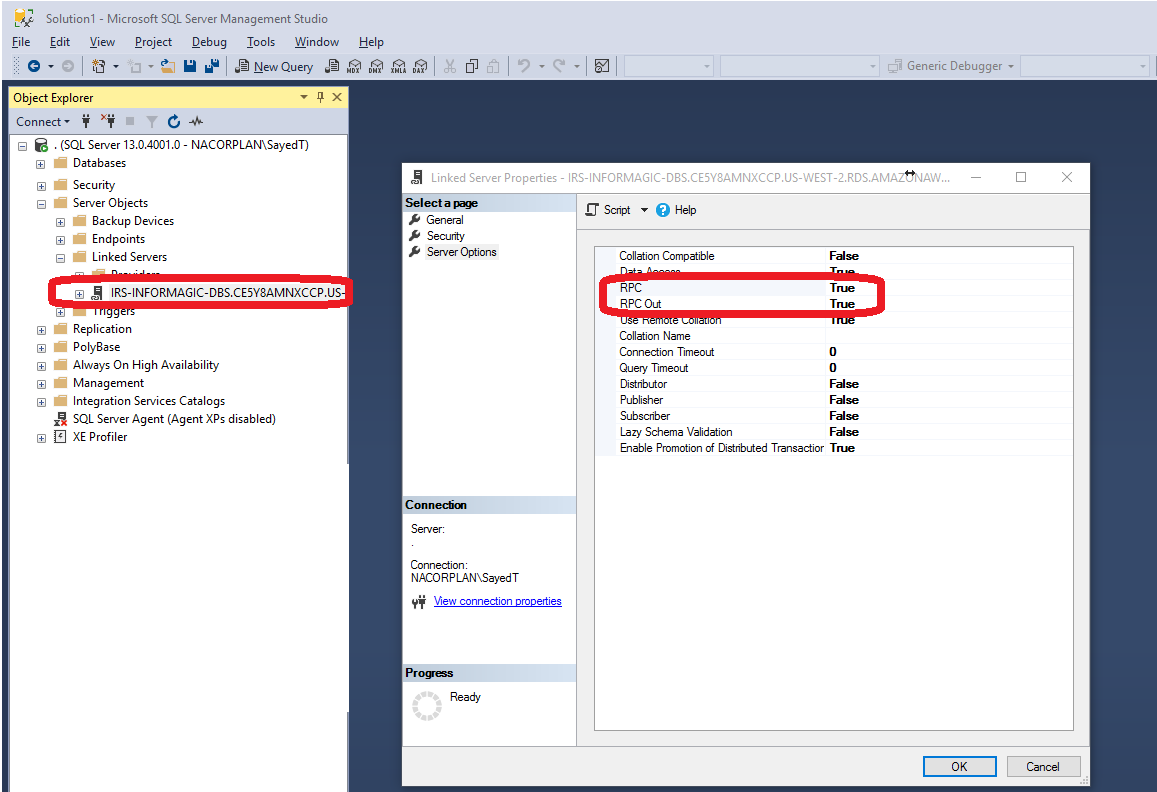
This step has 2 parts to it. At first, we will create the Linked server using SSMS for our RDS Endpoint and then we will create a SQL Job, so we could schedule it as per our needed frequency.

Start SSMS and create a Linked server under Server Objects as shown below.

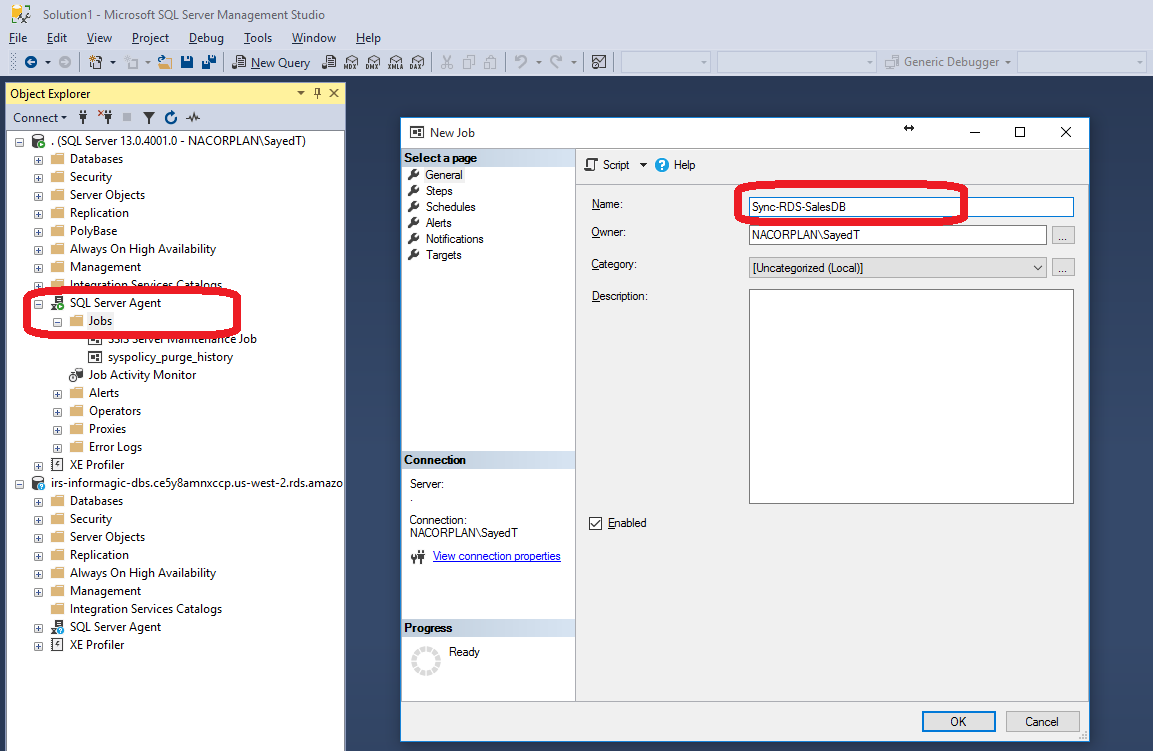


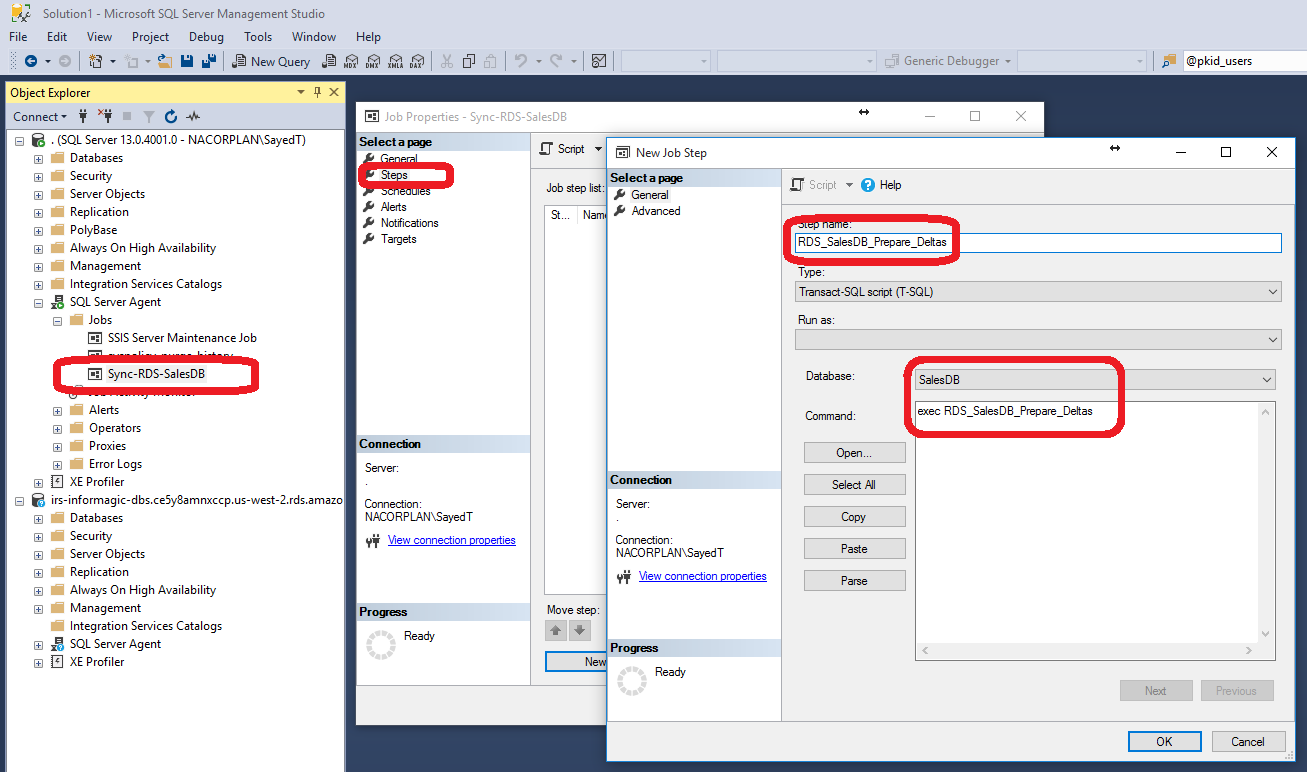


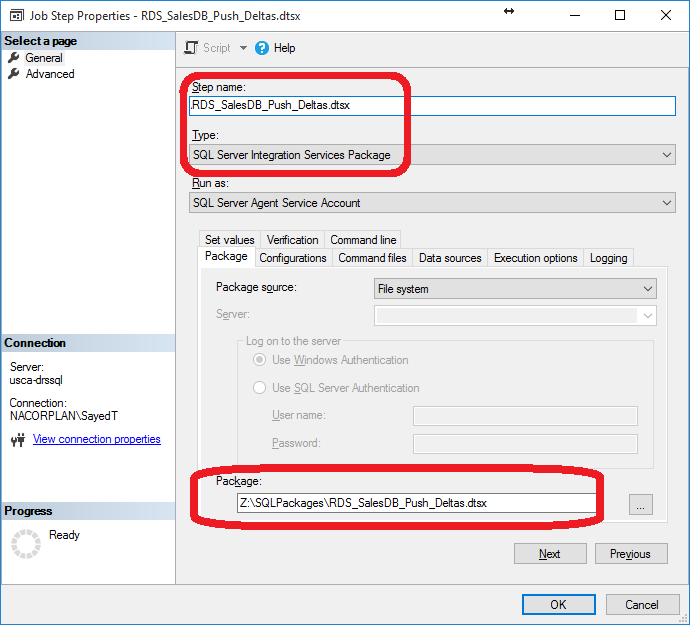
Right click on the linked server we just created and go to its properties, choose “Server options” tab and set “RPC” and “RPC Out” to true, as shown below.

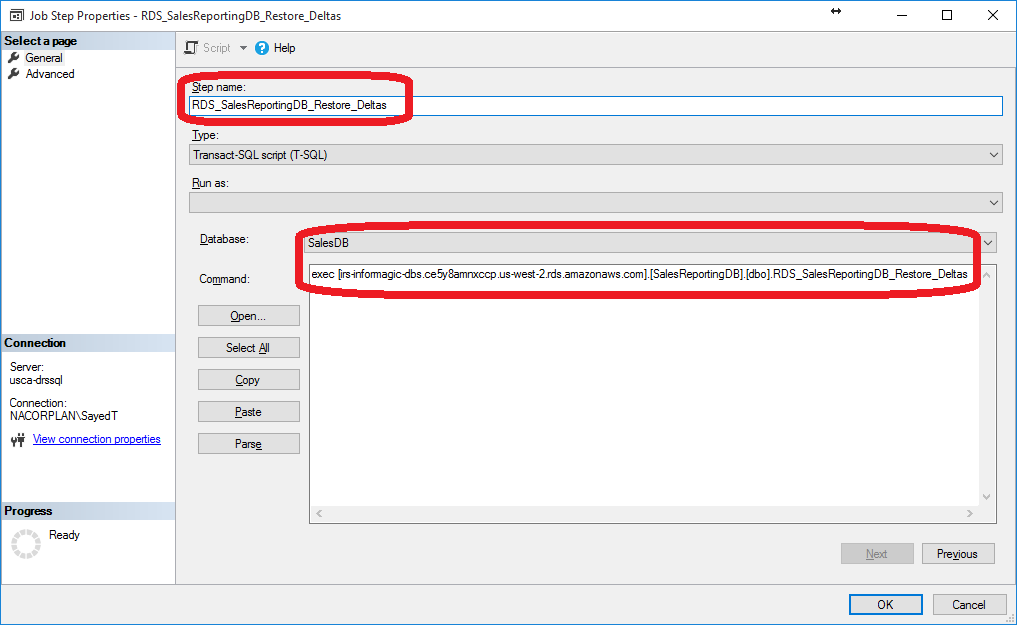


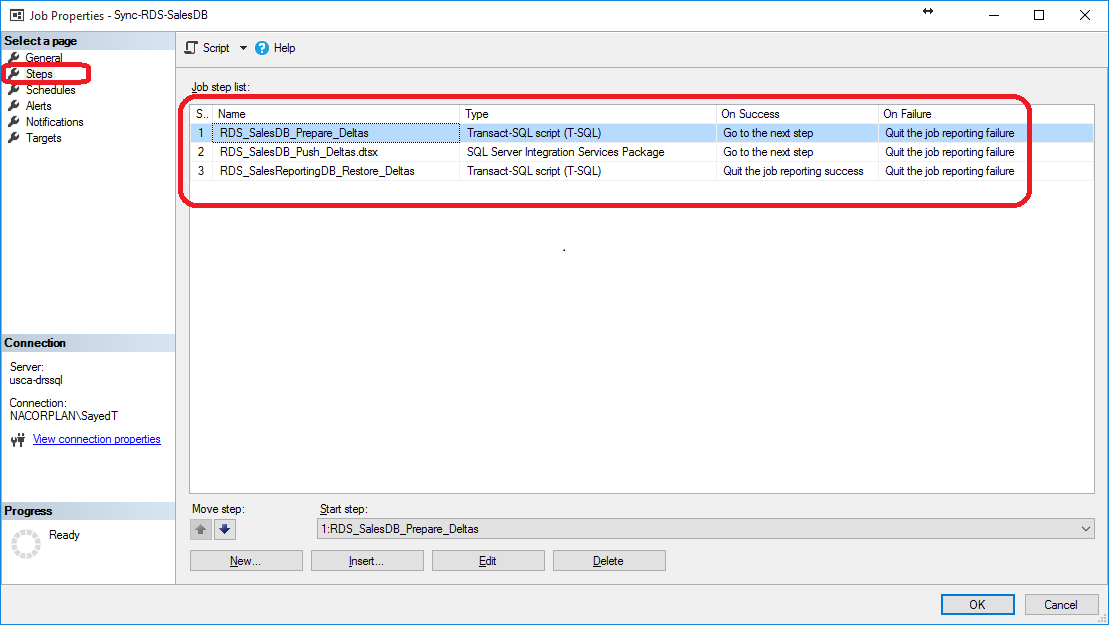
Once we have tested all the steps we will create a SQL Server job for them and schedule it as per our required frequency. SQL Server job is created using SQL Server Agent Services via SSMS as shown in the below images:











For more information on transferring deltas from on-prem SQL Server to Amazon RDS, you can reach me at [tabish.sayed@informa.com](mailto:tabish.sayed@informa.com).