Draft RGRTA User Guide

By: Michael Fung

Intricity Inc.

Table of Content

Contents

[Introduction 3](#_Toc468634793)

[Source Extraction 4](#_Toc468634794)

[Truncate and Load In Detail 5](#_Toc468634795)

[Incremental Load In Detail 6](#_Toc468634796)

[Source Extraction ETL Framework 7](#_Toc468634797)

# Introduction

ETL stands for Extraction, Transformation and Load. This process extracts data from various source systems and in the process of combining these disparate sources together is called transforming. Then the final step is called loading the data into a storage area called the Enterprise Data Warehouse.

The scope of this document encompasses the subject of Ridership. Therefore, to begin our discussion, we must examine the systems of data which provides for Ridership information. Please refer to [System of Sources](#_Source_Extraction)

The ETL code which extracts the source is written in two separate ways. Majority of the source extraction is called ‘Truncate And Load’. This process truncates or deletes all the data in the target data tables first and then copies the data from the source to the target table. The second process of extraction is called incremental load. The difference between Truncate and Load versus Incremental Load is that the latter does not truncate the target table but extracts and loads data within a date range from the source to the target table. Please refer to the [Truncate and Load In Detail](#_Truncate_and_Load) for further information.

The ETL code for the incremental load are done using a separate package from the truncate and load. Incremental packages have a prefix such as ‘Inc’ in front of the table name. For the table router\_block\_log, the prefix is E\_Inc means Extraction Incremental. Please refer to [Incremental Load In Detail](#_Incremental_Load_In).

The source extraction ETL process has a framework which tracks the processing status when it is running. This framework handles the following functions:

* Skip over successful steps during a rerun process
* Track error steps
* Show the status of the current run
* Keep historical details of process runs

Please refer to [Source Extraction ETL Framework](#_Source_Extraction_ETL) for further details.

Once all the data lands in the Staging table, the next step is to kick off the EDW process. The process contains its own framework and status control. Please refer to [Enterprise Data Warehouse](#_Enterprise_Data_Warehouse) for full details.

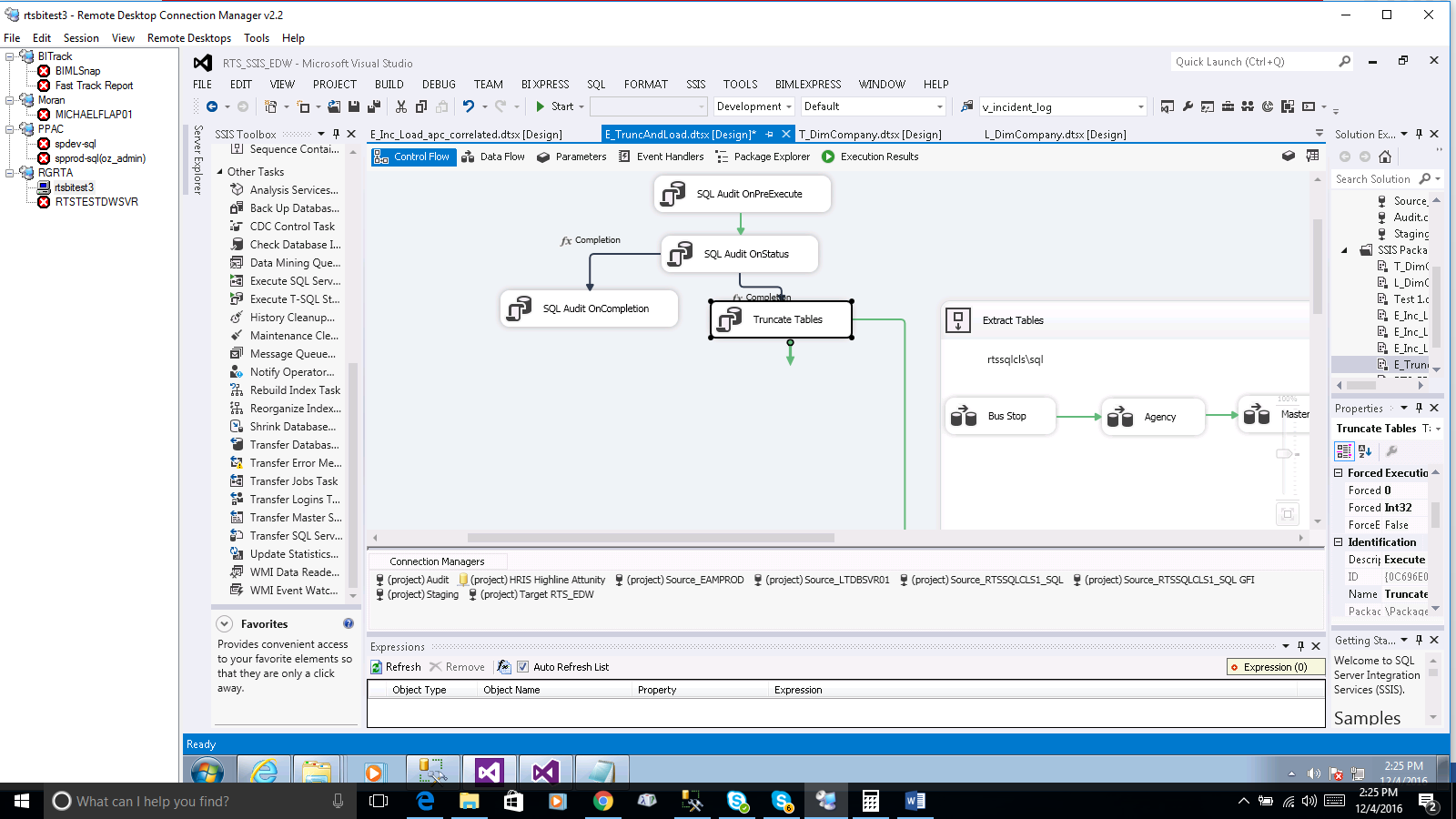
# Source Extraction

There are various systems to which we extract the data to compile into our Ridership ETL. These systems are:

|  |  |  |
| --- | --- | --- |
| **Server** | **Database** | **Description** |
| HRIS Highline Attunity | HRIS | Id: Scalability pw:active123, Oracle Server |
| EAMDBSVR | EAMPROD | SQL Server |
| LTDBSVR01 | orbcad\_ltdb | SQL Server |
| RTSSQLCLS1\SQL | Rochester\_PASS\_FX\_data | SQL Server |
| RTSSQLCLS1\SQL | GFI | SQL Server |

# Truncate and Load In Detail

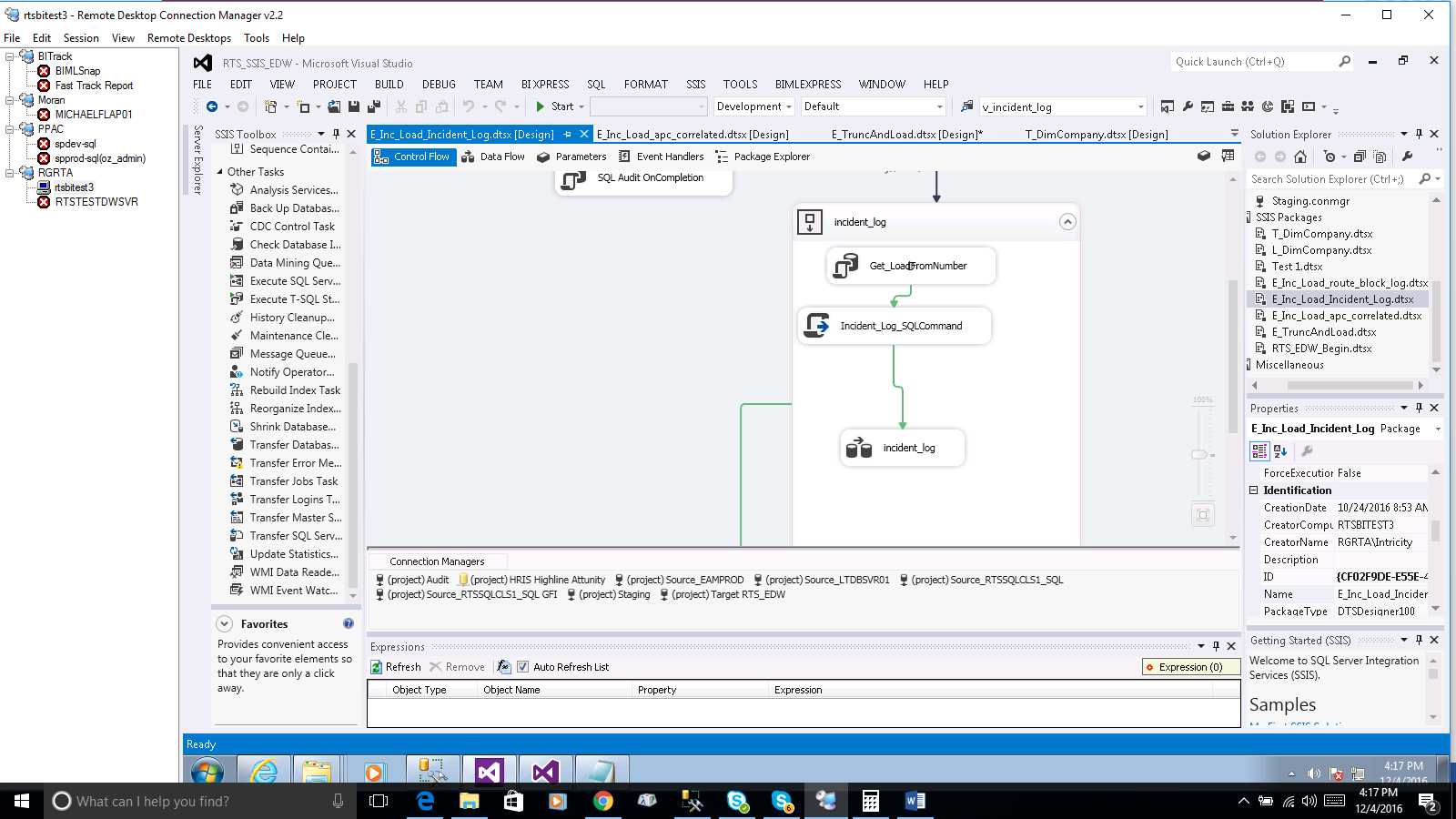
The Truncate table step is a call to a stored procedure (sp) EXEC sp\_TruncateTables. This stored procedure is located in RTS\_Staging database. It will delete all the tables that the parameters passed into this stored procedure. The table names in which the values are passed into this stored procedure is a table in the SQL Server database: SSISAudit, table called: TruncateTables. This table contains two fields: Chain and Table. The Chain describes the subject in which these tables are related to. For example, the tables Stops is used for the subject ‘Ridership’



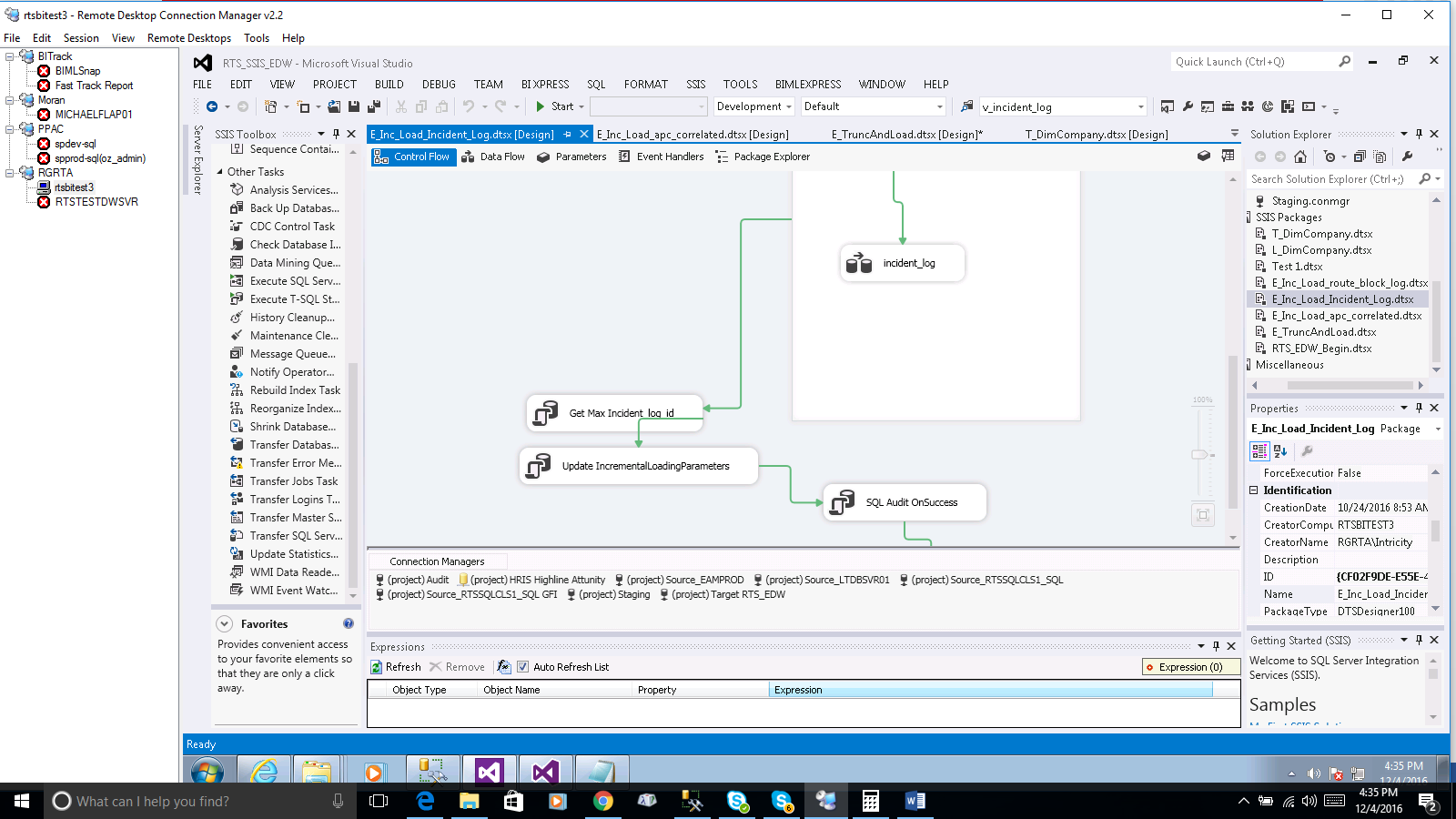
As you can see from the diagram, all ‘Truncate and Load tables’ are truncated first and then each source systems tables are loaded to the source tables in sequence. By looking at the green arrows, one can easily determine which source table is loaded first and which one is loaded next.

# Incremental Load In Detail

In order for incremental load, it requires tracking of the period in which the loading should start. There is a table called IncrementalLoadingParameters in SSISAudit which tracks the last value in which it was loaded. Please note that the value may not always be a date. In the case of Incident\_Log, there is no such date by which the data is ordered. Therefore, we track last loaded log by the last log number by which it was loaded.



The first step is to get the last loaded value from the IncrementalLoadingParameters table. This value is passed as a parameter into a sql statement inside the ‘Incident\_Log\_SQLCommand’ script task. Once the SQL Command is formulated, it is stored in a variable called user::SQLCommand. Next step is using a data flow to call the SQL variable as source extract to be loaded into the target table. When the dataflow completes loading the data to target table, we increment the IncrementalLoadingParameters table to the last or max value of the last inserted data to the target table.



# Source Extraction ETL Framework

To keep track of the current Source extraction ETL process, there are two tables which allows you to monitor the current state. The SSISAudit table holds two tables:

* StatusLog
* ExecutionLog

**StatusLog** gives you the current state of the where in the ETL the process is at. It contains only contains one run; either it is the current run which is still in progress or it contains the last run. This table contains the following fields:

Chain: Used for differentiating different process groupings. For example, Ridership, Finance…

PackageName: The Name of the Package as it is called

ProcessRunDate: The date of which this step started the run

Status: 1: Running (Incomplete)

2: Complete

3: Failed

DateLastUpdate: The date and timestamp in which it was last updated

**ExecutionLog** gives you the historical records of the ETL run.

Chain: Used for differentiating different process groupings. For example, Ridership, Finance…

LogID: Sequential number of ETL steps

Description: Additional information of the Package

PackageName: The Name of the Package as it is called

PackageGuid: Unique value for each Package

MachineName: Name of the Machine that ran the job

ExecutionGuid: Unique id which identifies the job run

ProcessRunDate: The date in which the jobran

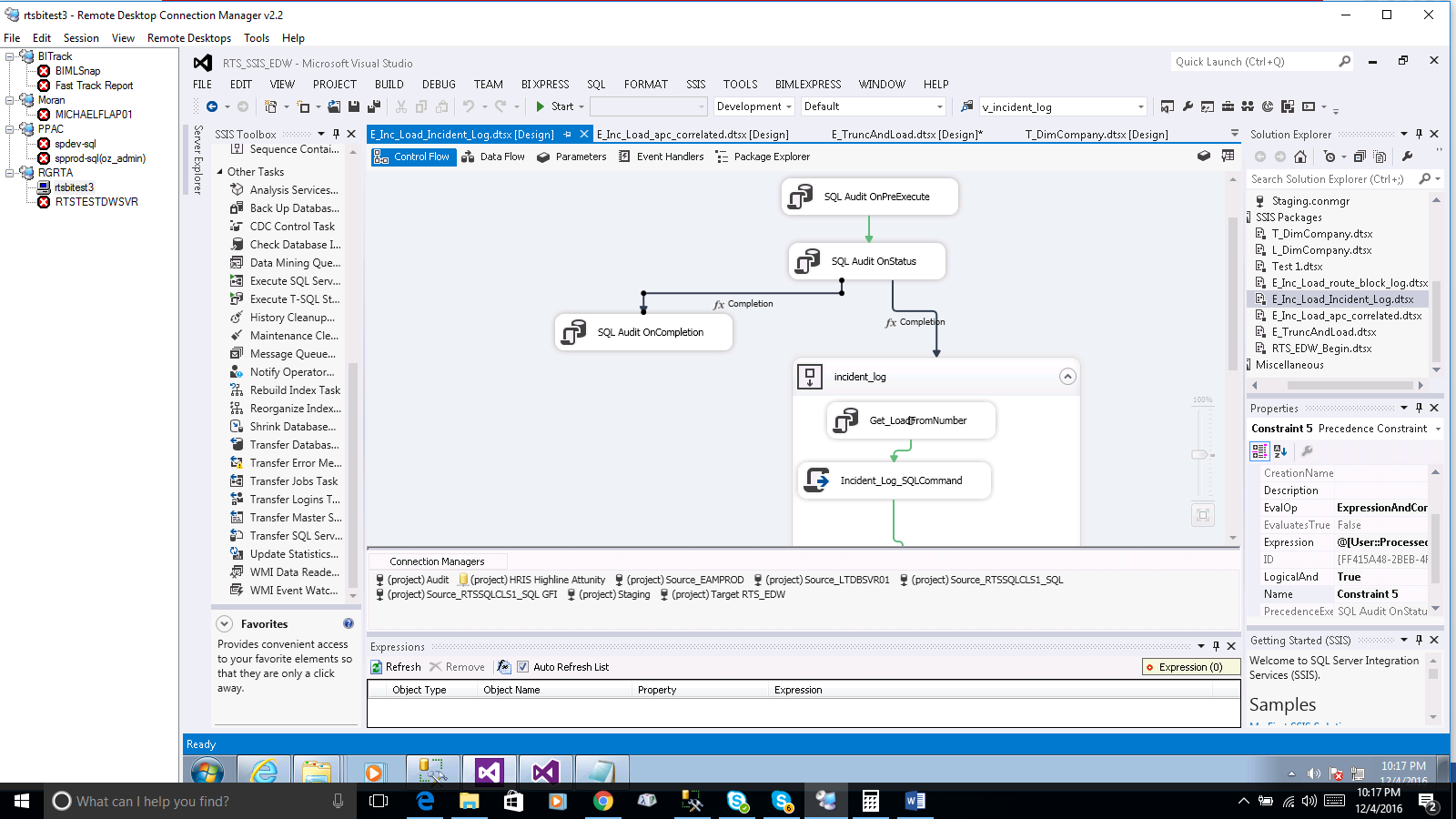
Operator: Name of owner who ran the job

StartTime: Start Date and Time in which the job ran

EndTime: End Date and Time in which the job ran

**Rerun in the event of an Error**

The framework detects that in the event of an error, those packages which previous ran successful will be skipped. The way it does that is to check the Status table for status value. If the job exists and the status is 2, a variable called user:ProcessedFlg will send the pipe execution ‘SQL Audit On Completion’ step.



# Enterprise Data Warehouse

Due to the repeating nature of the EDW load, an application has been used to generate the following packages to treat two groups of tables:

* Dimension
* Fact

The dimension tables are the first set of tables that are loaded into the (Enterprise Data Warehouse) EDW. The reason being that IDs to join to the fact tables must exist before the fact tables takes on the surrogate from the dimension table. All dimension tables are defined by the fields it contains are either (Slowly Changing Dimension ) SCD type 1 or type 2.

The dimension tables contains the following fields which identify changes in the dimension table and the metadata which

[DW\_Data\_Source] – source from which the data comes from

[DW\_Active\_Date] – Start date in which the row is valid

[DW\_Expiration\_Date] – End date in which the row is valid

[DW\_Active\_Flag] - Identifies whether the field is active = 1, not active = 0

[DW\_Insert\_Date] - The date in which the row was inserted

[DW\_Update\_Date] - The date in which the row was modified such as in the case of slow changing dimension. In most cases, the DW\_Insert\_Date is the same as the DW\_Update\_Date

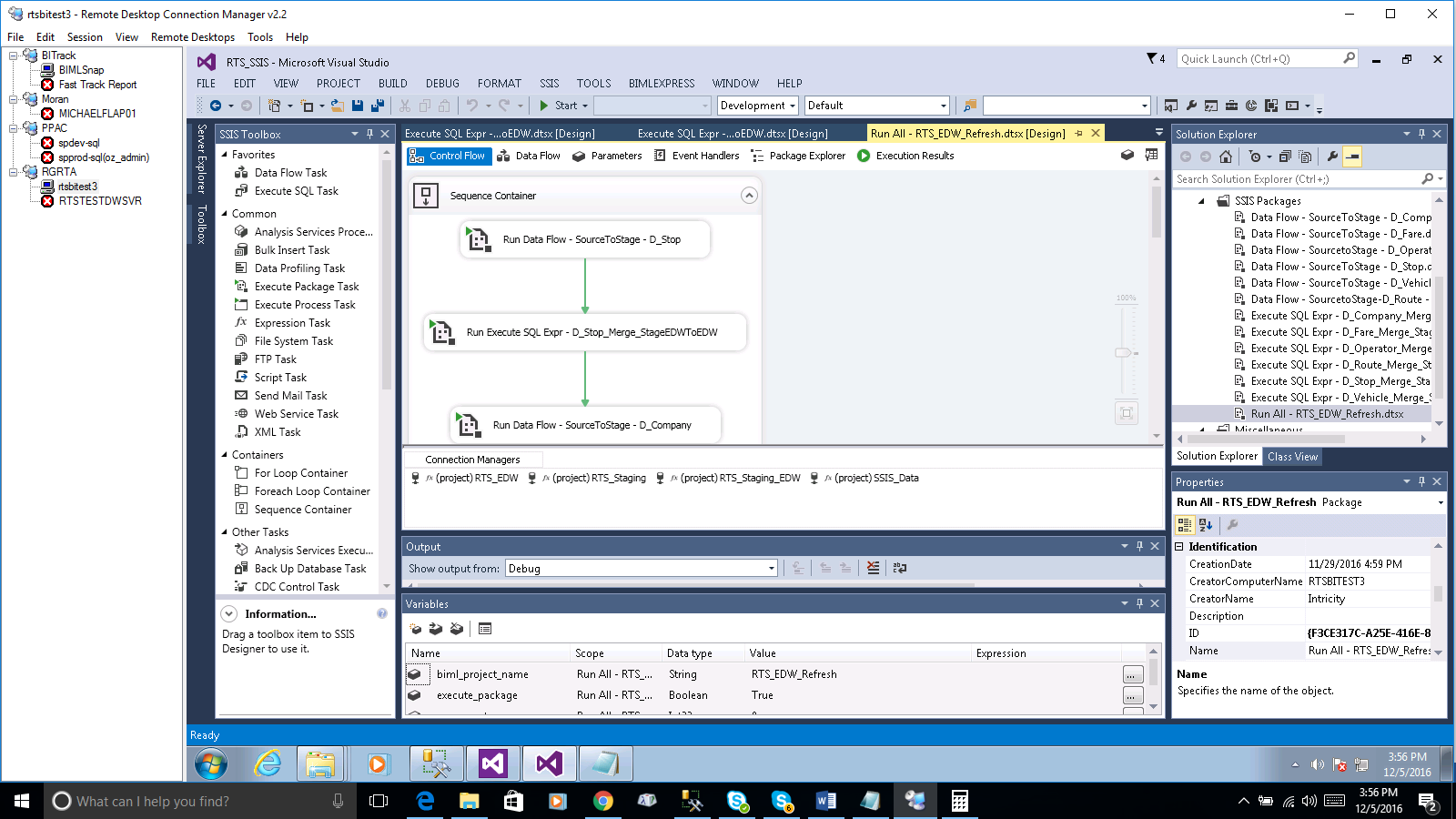
SCD Type 1 are those columns that do not keep versioning of any changes. Therefore, type 1 SCD fields are not concerned with any historical changes.

SCD Type 2 are those columns that do keep versioning of any changes. Therefore, type 2 SCD fields are track historical changes by:

1. Updating the ending date of the rows which type 2 fields have changed
2. Insert a new row fact table having the DW\_Expiration\_Date to ‘9999-12-31’

Looking at our sequence container, we see that there are two steps for each dimension. Referring to the diagram below, D\_Stop has two packages:

1. Source to Stage SQL Extraction
   1. Run Data Flow – SourceToStage – D\_Stop
2. SCD check
   1. Run Execute SQL Expr – D\_Stop\_merge\_StageEDWToEDW



**Source to Stage Extraction**

This step truncates the staging dimension table in the RTS\_Staging\_EDW and then load the data into staging table. From the staging table in THE RTS\_Staging\_EDW, the next step is to use a T-SQL merge statement to compare the SCD 1 and SCD 2 fields in the corresponding table in the RTS\_EDW database. Please refer to the code sample in the RGRTA\_ETL\_QueryAppendix.