Load TSIQ tables into Legacy EDW staging

Project Name –

Document Name – Development Document

Document Version –

Submission Date –

Prepared by SLK offshore  
Prepared for –

| Private & Confidential

1. Document History

**Change Record**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Revision Notes** |
|  |  |  |  |
|  |  |  |  |

**Reviewers**

| **Name** | **Position** | **Version Approved** | **Date** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

**Note to Holders**

If you receive an electronic copy of this document and print it out, please write your name on the equivalent of the cover page, for document control purposes.

If you receive a hard copy of this document, please write your name on the front cover, for document control purposes.

**Disclaimer**

The information within this document is solely for the purpose of the evaluation of the content and shall be kept confidential. This shall not be duplicated or disclosed in any form to any other party or used for any other purpose without prior written consent from SLK software.

Contents

[1.0. Document History 2](#_Toc106215303)

[2.0. Introduction 4](#_Toc106215304)

[3.0. SSMS-Source and Target Definitions 5](#_Toc106215305)

[4.0. Control Table 6](#_Toc106215306)

[5.0. Informatica- Source and Target connections 9](#_Toc106215307)

[6.0. Import Source and Target 19](#_Toc106215308)

[7.0. Mapping Development for Batch Control Tables 21](#_Toc106215309)

[8.0. Shortcuts for Staging Mappings 28](#_Toc106215310)

[9.0. Workflow Manager 29](#_Toc106215311)

[10.0. Workflow Creation 53](#_Toc106215312)

[11.0. Workflow Scheduler 54](#_Toc106215313)

1. Introduction

Document contains details of loading the data from TSIQ to EDW staging tables using Informatica Power Center 10.0.0 version.

TSIQ Source

EDW Staging Target

Views

List of all EDW Staging Tables are as follows:

1. vw\_data\_entities\_accounts
2. vw\_data\_entities\_producer\_businesses
3. vw\_data\_entities\_producer\_individuals
4. vw\_data\_entities\_submissions
5. vw\_policies\_com\_twosigmaiq\_protective\_auto\_AutoProjectedPeriod\_table
6. vw\_policies\_com\_twosigmaiq\_protective\_auto\_Vehicle\_table
7. vw\_policies\_com\_twosigmaiq\_protective\_PICAddress\_table
8. vw\_policies\_com\_twosigmaiq\_protective\_Underwriter\_table
9. vw\_policies\_commits
10. vw\_policies\_coveragecrossreferencedimension
11. vw\_policies\_deductibles
12. vw\_policies\_features
13. vw\_policies\_limits
14. vw\_policies\_primary\_named\_insured
15. vw\_policies\_protective\_top\_lines
16. vw\_policies\_rates
17. vw\_policies\_segments
18. vw\_workflows\_workflow\_commits
19. vw\_policies\_vehicletypecrossreferencedimension
20. vw\_policies\_com\_twosigmaiq\_protective\_PICSubmission\_table

**Here we have taken the example of policies\_commits table for demonstration of workflows.**

**Source System: TSIQ**

**Data Flow execution:**

1. The source database, all columns are currently defined as nvarchar (max) so created views on top of the tables has to match the target columns datatypes.
2. Created synonyms for the views and provide access using synonyms.
3. Populating TSIQ data from views into EDW Staging.
4. SSMS-Source and Target Definitions

**Source database:**

**Server:** pi-azsqlsr-mdp-dev.database.windows.net

**Database:** pi-azsqldb-tsiq

**Source Views (Creating a views in TSIQ)**

1. From TSIQ, identified 20 tables to load the data into EDW Staging.
2. All columns from 20 tables are defined as nvarchar(max)
3. Example in source date column is defined as nvarchar (max), so converted respective date column as DATE datatype in views.
4. Analyzed the data and convert datatype, length, and precision to every column, Created views and to match the target column datatypes.
5. Below for the reference creation of view for **vw\_policies\_commits.**

**Naming Convention followed:** **vw\_<table\_name> [vw\_policies\_commits]**



**Target database:**

**Server:** dbEntDWHDev

**Database:** dbEntDWHStageDev

Creating Target Tables:

**Naming Convention followed:** **TSIQ\_<table\_Name> [TSIQ\_Polices\_Commits]**



**Synonyms Creation:**

* Created synonyms in ENT for all 20 views in **dbEntDWHStageDev** database to access the views.
* Creation of synonyms for all views for reference below for the **vw\_polices\_commits**

CREATE SYNONYM [EntDWHStageDev]. [dbo].[vw\_policies\_commits] FOR [dbo].[vw\_policies\_commits]

Note: Additional of 3 new audit columns (LastUpdateDateTime, DWHCreateDate, SystemID) to all staging tables.

1. Control Table

**Control table details:**

1. **System Batch Control:**

This contains source system information like systemID, SystemShortDesription, SystemLongDescription, LastUpdateDate, LastUpdateDateTime, CompanyId, LineOfBusinessId, ParentSystemDescriptionand DWHCreateDate.

Initially insert row into SystemBatchControl with respect to SRC system, the same inserted record will be updated (date columns) during run.

**Insert script:**

INSERT INTO EntDWHDev.dbo.SystemBatchControl

(

SystemID,

SystemShortDesription,

SystemLongDescription,

LastUpdateDate,

LastUpdateDateTime,

CompanyId,

LineOfBusinessId,

ParentSystemDescription,

DWHCreateDate

)

VALUE (6,'TSIQ','Two Sigma - TSIQ','1900-01-01 00:00:00.000','1900-01-01 00:00:00.000', 3, 0, NULL, GETDATE ());

**Select script:**

SELECT \*

FROM dbo.SystemBatchControl

WHERE SystemShortDesription = 'TSIQ';



1. **DWH Batch Control:**

DWH Batch Control table contains details of batch status and source system information.

**Insert Script:**

INSERT INTO EntDWHDev.dbo.DWHBatchControl

(

SystemID,

BatchId,

SystemShortDesription,

SystemDescription,

BatchDescription,

BatchFromDateTime,

BatchToDateTime,

BatchScheduleSuccessfulCompleteFlag,

DWHCreateDate

)

VALUE (66, 1,'TSIQ','Two Sigma - TSIQ','TSIQ Nightly Batch','1900-01-01 00:00:00.000','1900-01-01 00:00:00.000', NULL, GETDATE ());

**Select script:**

SELECT \*

FROM dbo.DWHBatchControl

WHERE SystemShortDescription='TSIQ';

**C:\Users\PChikkarangaiah\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\A39C7636.tmp**

1. **DWH Batch Table Control :**

DWH Batch Table Control table contains table-level audit information and will be used to capture the incremental data from each source table.

**Insert Script:**

INSERT INTO EntDWHDev.dbo.DWHBatchTableControl

(

SystemID,

BatchId,

SystemShortDescription,

SystemDescription,

BatchDescription,

TableName,

BatchFromDateTime,

BatchToDateTime,

BatchScheduleSuccessfulCompleteFlag,

DWHCreateDate

)

VALUES (66,1,'TSIQ','Two Sigma - TSIQ','TSIQ Nightly Batch','vw\_policies\_commits','1900-01-01 00:00:00.000','1900-01-01 00:00:00.000',NULL,GETDATE());

**Select Script:**

SELECT \*

FROM dbo.DWHBatchTableControl

WHERE TableName = 'vw\_policies\_commits';

**C:\Users\PChikkarangaiah\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F2AED7B4.tmp**

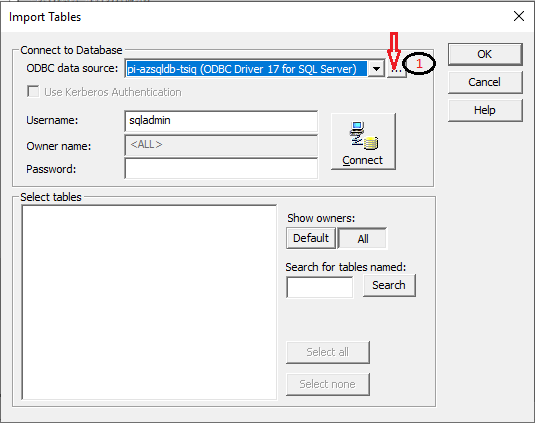
1. Informatica- Source and Target connections

Use Power Center Designer and connect to the DEV repository **PC\_DEV\_REP**

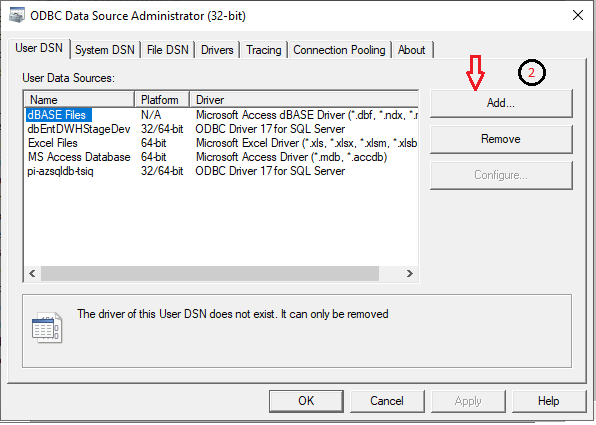
**Source database connection:**

* To import Source use ODBC Driver 17 for SQL Server or higher to connect to the Azure database.
* To import Target use respective windows authentication.

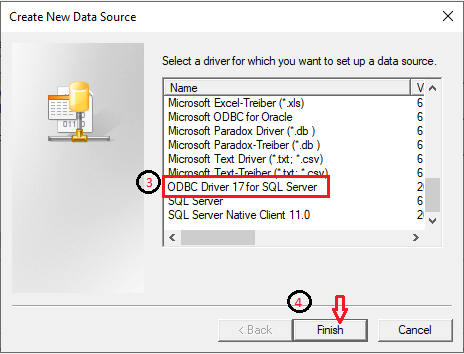
Set up for database connection for Source is shown below:



1. Click (…) three dot button to add or configure or remove database connection.

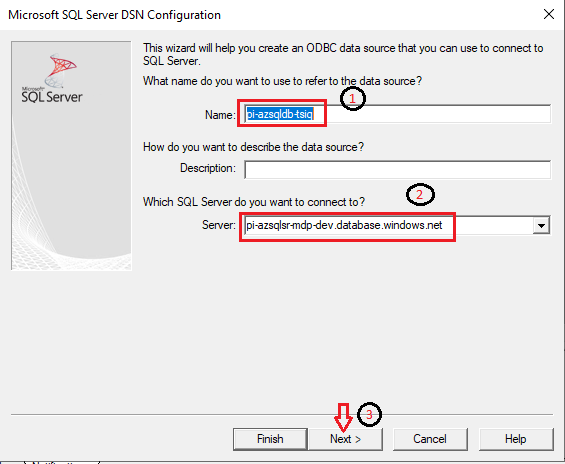


2. It will ask for driver which we want to set up a data source.

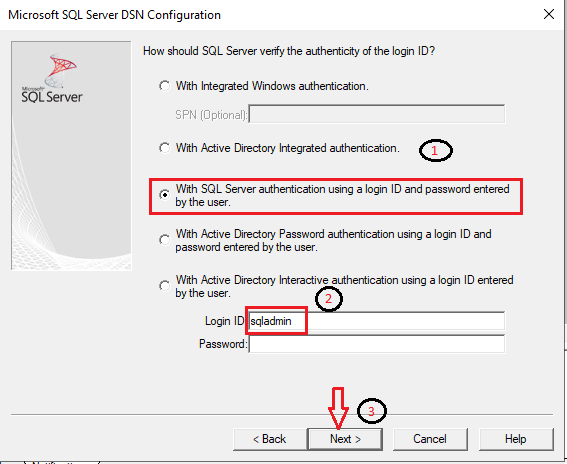


3. Select **ODBC Driver 17 for SQL Server** for souce.

4. After Clicking the **Finish** button it will ask create New Data Source to SQL Server.



1. Give reference **Name** for data source, this will identify the source.
2. Use **pi-azsqlsr-mdp-dev.database.windows.net** asServer Name to connect source data.
3. Click **Next** to continue**.**

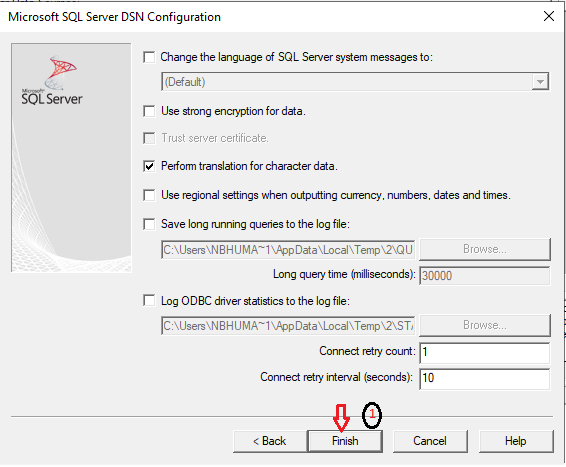
****

1. Select **with SQL Server authentication using a login ID and password entered by the user** option**.**
2. Give SQL Server Source Credentials **Login ID** and **Password.**
3. Click **Next** to continue.

Graphical user interface, text, application

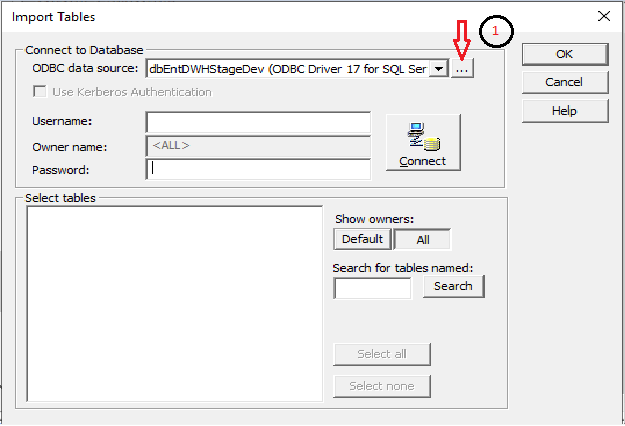
Description automatically generated

1. Change default database to respective database **pi-azsqldb-tsiq.**
2. Enable the ANSI quoted identifiers.
3. Enable the Use ANSI nulls, padding and warning.
4. Enable the **READWRITE** permissions.
5. Enable the Transparent Network IP Resolution.
6. Click **Next** to continue.

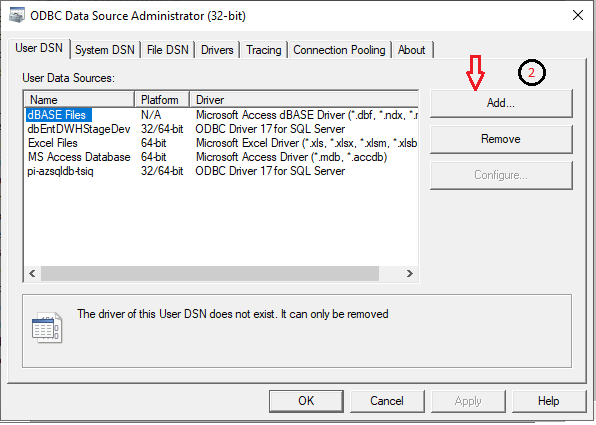


1. Keep all the fields by default and click **Finish** button.

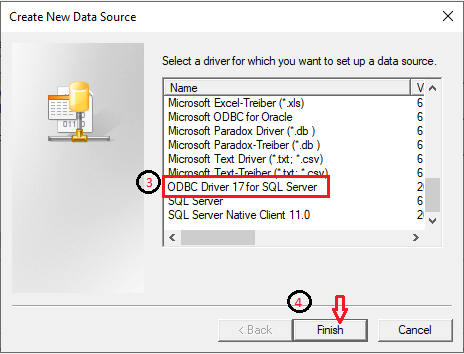
**SQL Server target DB Configuration**



1. Click (…) three dot button to add or configure or remove database connection.



2. It will ask for driver which we want to set up a data source.



3. Select **ODBC Driver 17 for SQL Server** for souce.

4. After Clicking the **Finish** button it will ask create New Data Source to SQL Server.

Set up for database connection for Target is shown below:

Graphical user interface, text, application

Description automatically generated

1. Provide reference name for data source as **dbEntDWHStageDev.**
2. Use **dbEntDWHDev as** Server Name to connect target database.
3. Click **Next** to continue.

Graphical user interface, text, application, email

Description automatically generated

1. Select **With integrated Windows authentication** for using windows credentials.
2. Provide windows credentials for **Login ID** and **Password.**
3. Click **Next** to continue**.**

Graphical user interface, text, application

Description automatically generated

1. Change the default database to respective **EntDWHStageDev** database.
2. Enable the ANSI quoted identifies.
3. Enable the Use ANSI nulls, paddings and warnings.
4. Enable the **READWRITE** permissions.
5. Enable the Transparent Network IP Resolution.
6. Click **Next** to continue.

Graphical user interface, text, application

Description automatically generated

1. Keep all the fields by default and click **Finish.**
2. Import Source and Target

**Folder Name:** **TSIQ\_Shared**

**Import Source Tables**:

**Step 1:** Import Source from respective database by giving **username** and **password.**

Graphical user interface, application

Description automatically generated

1. Select appropriate ODBC connection **pi-azsqldb-tsiq (ODBC Driver 17 for SQL Server).**
2. Enter the **Username**.
3. Enter the **password.**
4. Click on **Re-connec**t for validating the credentials and then we will get the list of tables.
5. Select **Table**, Here selecting **vw\_policies\_commits** table.
6. Click the **OK** button.

**Step 2:** After importing the source need to **save (ctrl+s)** definitions.

**Import Target Tables:**

**Step1:** While importing target table need to pass **username** and **password** here, we have to pass **windows credentials**

Graphical user interface, application

Description automatically generated

1. Select appropriate ODBC connection **dbEntDWHStageDev (ODBC Driver 17 for SQL Server).**
2. Enter the **Username**.
3. Enter the **password.**
4. Click on **Re-connec**t for validating the credentials and then we will get the list of tables.
5. Select **Table**, Here selecting the **TSIQ\_Policies\_Commits** table.
6. Click the **OK** button.

**Step 2:**

After importing the target table need to **save (ctrl+s)** definitions.

1. Mapping Development for Batch Control Tables

1. **m\_SystemBatchControl\_TSIQ\_10\_upd**
2. SystemBatchControl conatin every source system information Like SystemID, SystemShortDesription, SystemLongDescription, LastUpdateDate, LastUpdateDateTime, CompanyId, LineOfBusinessId, ParentSystemDescription, DWHCreateDate
3. SystemBatchControl table will get information like when the last run happened for a specific source system.
4. This mapping will be filtering only for **TSIQ** Source systemlike **SystemShortDesription = 'TSIQ**'
5. This mapping will update the **LastUpdateDate** and **LastUpdateDateTime** columns, Based on **SystemID.**

Graphical user interface, application

Description automatically generated

1. In Mapping Designer when we drag a source as **SystemBatchControl** we will get the Source Qualifier automatically
2. In Source Qualifier filter **SystemBatchControl.SystemShortDesription = 'TSIQ**'.
3. In **EXP\_timestamps** expression transformation creates two variable ports
4. SystemID = SystemID
5. o\_LastUpdateDate = **TRUNC**(**SYSTIMESTAMP**())
6. o\_LastUpdateDateTime = **SYSTIMESTAMP**()
7. The following fields to the target table
8. SystemID
9. LastUpdateDate
10. LastUpdateDateTime
11. **m\_DWHBatchControl\_TSIQ\_10\_upd**
12. DWHBatchControl table contains details of batch status and source system information.
13. This mapping will be updating **BatchScheduleSuccessfulCompleteFlag as N, BatchToDateTime, and BatchFromDateTime** in DWHBatchControl table based on key columns are SystemID and BatchID.

Graphical user interface, application, Teams

Description automatically generated

1. In Mapping Designer when we drag a source as **as DWHBatchControl** we will get the Source Qualifier automatically
2. In **Source Qualifier** filter **DWHBatchControl.SystemShortDescription = 'TSIQ'** and **DWHBatchControl.BatchId = 1**
3. In **EXP\_Derive\_Dates** expression transformation create three variable ports
4. o\_FromDate = **iif**(BatchScheduleSuccessfulCompleteFlag = 'Y', BatchToDateTime, BatchFromDateTime)
5. o\_ToDate = **sysdate**
6. o\_rest\_to\_N = 'N'
7. The following fields to Target table
8. SystemID
9. BatchId
10. BatchFromDateTime
11. BatchToDateTime
12. BatchScheduleSuccessfulCompleteFlag
13. **m\_DWHBatchTableControl\_TSIQ\_10\_upd\_roll\_values**
14. **DWHBatchTableControl** table contains table-level audit information and will be used to capture the incremental data from each source table.
15. This mapping will update **BatchFromDate** and **BatchScheduleSuccessfulCompleteFlag** Columns Based on Key columns.
16. The key columns are SystemID, BatchID and TableName.

Graphical user interface, application

Description automatically generated

1. In Mapping Designer when we drag a source as **DWHBatchTableControl** we will get the Source Qualifier automatically
2. In **Source Qualifier** filter the following:

DWHBatchTableControl.BatchToDateTime IS NOT NULL and

DWHBatchTableControl.SystemShortDescription = 'TSIQ' and

DWHBatchTableControl.BatchId = 1

1. In **EXP\_determine\_if\_to\_roll\_dates** expression transformation create two variable ports
2. o\_Date = **if**(BatchScheduleSuccessfulCompleteFlag = 'Y', BatchToDateTime, BatchFromDateTime)
3. o\_reset\_to\_N = 'N'.
4. The following fields to the Target table
   1. SystemID
   2. BatchID
   3. TableName
   4. BatchFromDateTime
   5. BatchScheduleSuccessfulCompleteFlag

1. **m\_DWHBatchTableControl\_TSIQ\_20\_upd\_update\_max\_timestamp**
2. **DWHBatchTableControl** table contains table-level audit information and will be used to capture the incremental data from each source table.
3. This mapping will update the control record with its max process date- **BatchToDateTime**

Graphical user interface, application

Description automatically generated

1. In Mapping Designer when we drag a source as **DWHBatchTableControl** we will get Source Qualifier automatically
2. In **Source Qualifier** filter
3. DWHBatchTableControl.SystemShortDescription = 'TSIQ' & DWHBatchTableControl.BatchId = 1
4. In **EXP\_Default\_date** Expression transformation create one variable port
5. Creating variable as default date
6. o\_Default\_Date = **to\_date**('01-JAN-1900', 'DD-MON-YYYY')
7. **LKP\_vw\_policies\_commits\_MAX\_TimeStamp** Lookup transformation
8. In **Lookup Sql Override**

SELECT

Max (case when dwhupdateddatetime is not null then dwhupdateddatetime else dwhcreateddatetime end) as BatchDate

FROM vw\_policies\_commits

Condition:

**BatchDate** >= **o\_Default\_Date**

1. BatchDate will pick the value from **vw\_policies\_commits** table
2. o\_Default\_Date used in this lookup to perform joins.
3. In this expression transformation
   1. **BatchDate** renamed as a **vw\_policies\_commits\_TIME\_STAMP**
   2. **o\_To\_DateTime = if**(TableName = 'vw\_policies\_commits', vw\_policies\_commits\_TIME\_STAMP, NULL)

Filter all other tables based on the above logic

1. The following fields to the Target table
   1. SystemID
   2. BatchID
   3. TableName
   4. BatchToDateTime
2. **m\_TSIQ\_policies\_commits\_10\_ins\_and\_upd**

Once source(s) and target(s) are imported, created a mapping to populate them. One mapping per stage table. Following mapping development. Policies commits mapping to load data from view to target table with adding audit columns.

Graphical user interface, application

Description automatically generated

1. In Mapping Designer drag a source **as DWHBatchControl** will get the Source Qualifier automatically
2. In Mapping Designer when we drag a source **as vw\_policies\_commits** will get the Source Qualifier automatically
3. Will place it all required columns in one source qualifier to join both tables

**Source Qualifier Logic:**

DWHBatchTableControl. TableName = 'vw\_policies\_commits' and DWHBatchTableControl.SystemShortDescription = 'TSIQ' and DWHBatchTableControl.BatchId = 1 and COALESCE (vw\_policies\_commits.DWHUpdatedDateTime, vw\_policies\_commits.DWHCreatedDateTIme) > DWHBatchTableControl.BatchFromDateTime

Order by case when vw\_policies\_commits.DWHUpdatedDateTime is not null

then vw\_policies\_commits.DWHUpdatedDateTime

else vw\_policies\_commits.DWHCreatedDateTime end

1. **EXP\_SystemID\_and\_UpdateTimestamp** is Reusable Expression Transformation have 3 ports
2. i\_char
3. SystemID = 66
4. LastUpdateDateTime = **SYSTIMESTAMP**()
5. Pass all respective fields to the target.
6. **m\_DWHBatchTableControl\_TSIQ\_30\_upd\_batch\_success\_flag**

This mapping will have two pipelines for below tables, updating **BatchScheduleSuccessfulCompleteFlag**as **Y.**

1. DWHBatchTableControl
2. DWHBatchControl

Graphical user interface, application

Description automatically generated

1. **DWHBatchTableControl(First Pipeline)**
   1. In Mapping Designer drag a source as **DWHBatchTableControl**, will get the Source Qualifier automatically
   2. **In Source Qualifier filter**

DWHBatchTableControl.BatchToDateTime IS NOT NULL and DWHBatchTableControl.SystemShortDescription = 'TSIQ' and

DWHBatchTableControl.BatchId = 1

* 1. **EXP\_batch\_complete\_flag** expression transformationcreating one variable port **o\_batchcomplete\_flag =** 'Y'
  2. The following fields to the Target table

1. SystemID
2. BatchID
3. TableName
4. BatchScheduleSuccessfulCompleteFlag
5. **DWHBatchControl(First Pipeline)**
   1. In Mapping Designer drag a source as DWHBatchControl, then automatically source qualifier has come.
   2. **In Source Qualifier filter**

DWHBatchControl.SystemShortDescription = 'TSIQ' and

DWHBatchControl.BatchId = 1

* 1. **EXP\_batch\_complete\_flag** expression transformationcreating one variable port **o\_batchcomplete\_flag =** 'Y'
  2. The following fields to the Target table

1. SystemID
2. BatchID
3. BatchScheduleSuccessfulCompleteFlag
4. Shortcuts for Staging Mappings

**Folder Name:** **0100\_Enterprise\_App\_Schedule**

1. Shortcuts to mappings, sessions, worklet, and workflows no mappings will exist in this folder, simply shortcuts to the mappings in the **0100\_Enterprise\_App\_Schedule** folder.

**Shortcut\_to\_m\_TSIQ\_policies\_commits\_10\_ins\_and\_upd**

Graphical user interface, application

Description automatically generated

1. In Mapping Designer drag a source **as DWHBatchControl** will get the Source Qualifier automatically
2. In Mapping Designer when we drag a source **as vw\_policies\_commits** will get the Source Qualifier automatically
3. Will place it all required columns in one source qualifier to join both tables

**Source Qualifier Logic:**

DWHBatchTableControl. TableName = 'vw\_policies\_commits' and  DWHBatchTableControl.SystemShortDescription = 'TSIQ' and  DWHBatchTableControl.BatchId = 1 and COALESCE (vw\_policies\_commits.DWHUpdatedDateTime,vw\_policies\_commits.DWHCreatedDateTIme) > DWHBatchTableControl.BatchFromDateTime

order by  case when vw\_policies\_commits.DWHUpdatedDateTime is not null

then vw\_policies\_commits.DWHUpdatedDateTime

else vw\_policies\_commits.DWHCreatedDateTime end

1. **EXP\_SystemID\_and\_UpdateTimestamp** is Reusable Expression Transformation have 3 ports
2. i\_char
3. SystemID = 66
4. LastUpdateDateTime = **SYSTIMESTAMP**()
5. Pass all respective fields to the target table
6. Workflow Manager

Open Power Center Workflow Manager and connect to the DEV repository.

**Folder Name: 0100\_Enterprise\_App\_Schedule**

**Dependency**: On success of previous session the following session will trigger.

**Creation of Worklet:**

**wklt\_TSIQ\_Staging\_Nightly\_Batch**

Graphical user interface, application

Description automatically generated

Worklet in Informatica is a set of multiple tasks connected with start task link and **triggers the proper sequence to execute a process**. When a workflow in Informatica is executed, it triggers a start task and other tasks connected in the workflow, **wklt\_TSIQ\_Staging\_Nightly\_Batch** is combination of all below mention **sessions.**

**The workflow of sessions as follows:**

1. **s\_m\_SystemBatchControl\_TSIQ\_10\_upd**
2. **s\_m\_DWHBatchControl\_TSIQ\_10\_upd**
3. **s\_m\_DWHBatchTableControl\_TSIQ\_10\_upd\_roll\_values**
4. **s\_m\_DWHBatchTableControl\_TSIQ\_20\_upd\_update\_max\_timestamp**
5. **s\_m\_TSIQ\_policies\_commits\_10\_ins\_and\_upd**
6. **s\_m\_DWHBatchTableControl\_TSIQ\_30\_upd\_batch\_success\_flag**

Each session is explained below in detail.

1. **s\_m\_SystemBatchControl\_TSIQ\_10\_upd**

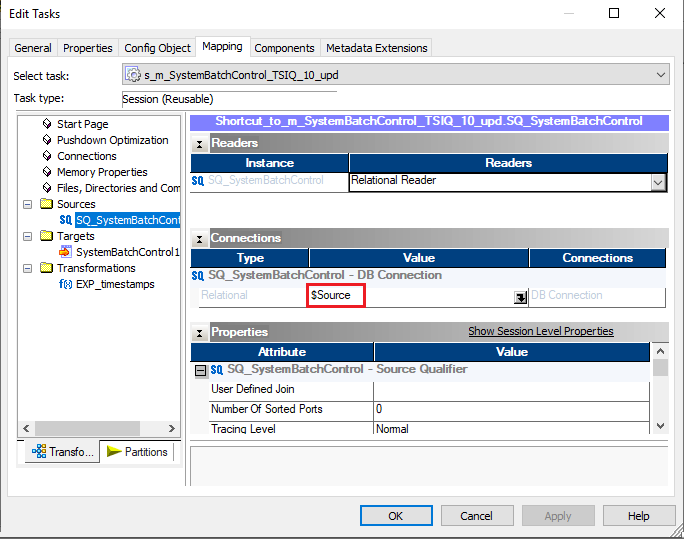
Creation of reusable session for mapping, set properties as mentioned below screenshot.

Graphical user interface, text, application, email

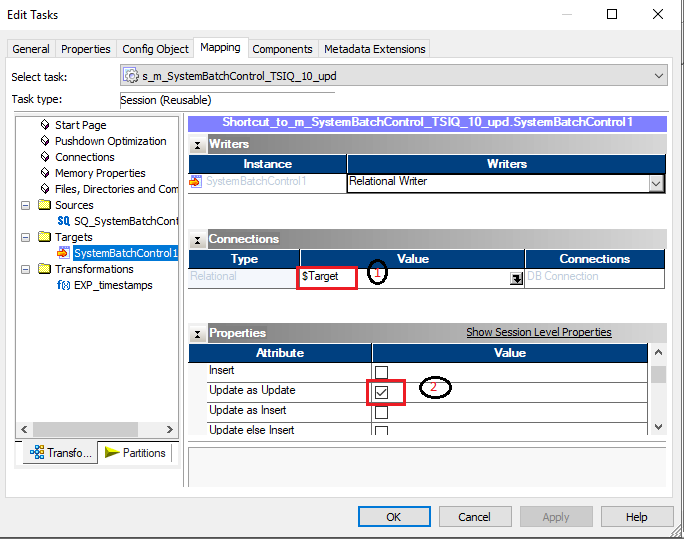
Description automatically generated

1. **Write Backward Compatible Session Log File**: It will write the workflow or session logs into a text file (find the Session Log under server directory).
2. **$Source connection value** – set to $DBConnectionWarehouse, This variable is set by a parameter file.
3. **$Target connection value** – set to $DBConnectionWarehouse, This variable is set by a parameter file.
4. **Treat source rows as: Update** Integration Service marks all rows to update the target.
5. **Recovery Strategy:** Restart task - When the Integration Service recovers a workflow, it restarts each recoverable task that is configured with a restart strategy.

**Config Objects Tab –**Default Value

****

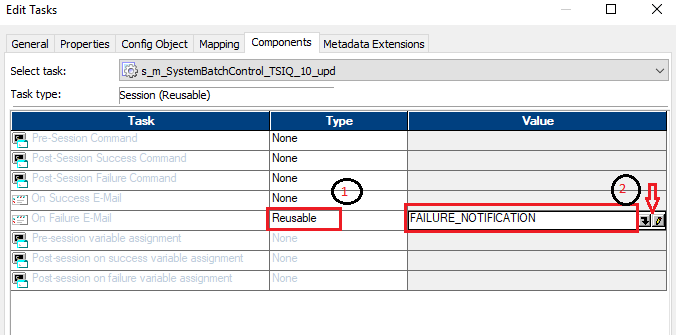
**$Source:** This variable is set by a parameter file



1. **$Target** :This variable is set by a parameter file
2. **Update as Update :** Update the row in the target table

**Components:**

**Reusable EMAIL task:** FAILURE\_NOTIFICATION



1. Use **Reusable** Type.
2. Click 🖉 Button (refer to the below screenshot)

Graphical user interface, text, application, email

Description automatically generated

1. **Email User Name:** $$FailureNotificationEmailGroup - This variable is set by a parameter file
2. **Email Subject** : Error %s Failed in %d - %n – Here
   1. %s - Session name.
   2. %d - Name of the repository containing the session.
   3. %n - Name of the folder containing the session.
3. Click button(refer below screenshot)

Graphical user interface, text, application

Description automatically generated

**Email Text:** Using the below text

Folder: %n

Session: %s

Mapping: %m

Target Details: %t

Records Loaded: %l

Records Rejected: %r

See attached log file for details on failure.

%g

1. **s\_m\_DWHBatchControl\_TSIQ\_10\_upd**

After creation of reusable session for mapping, set properties as mentioned in below screenshot.

Graphical user interface, text, application, email

Description automatically generated

1. **Write Backward Compatible Session Log File :** It will write the workflow or session logs into a text file (find the Session Log under server directory)
2. **$Source connection value** – set to $DBConnectionWarehouse, This variable is set by a parameter file
3. **$Target connection value** – set to $DBConnectionWarehouse, This variable is set by a parameter file
4. **Treat source rows as : Update** Integration Service marks all rows to update the target
5. **Recovery Strategy:** Restart task - When the Integration Service recovers a workflow, it restarts each recoverable task that is configured with a restart strategy.

Graphical user interface, application

Description automatically generated

1. **$Target:** This variable is set by a parameter file.
2. **Insert:** If enabled, the Integration Service uses the target update mode (Update else Insert) to update rows.
3. **Update else Insert:** Integration Service updates existing rows and inserts other rows as if marked for insert, must also select the Insert target option.

Graphical user interface, application, table

Description automatically generated

1. Use **Reusable** Type
2. Click 🖉 Button (refer to the below screenshot)

Graphical user interface, text, application, email

Description automatically generated

1. **Email User Name:** $$FailureNotificationEmailGroup - This variable is set by a parameter file
2. **Email Subject :** Error %s Failed in %d - %n – Here
   1. %s - Session name.
   2. %d - Name of the repository containing the session.
   3. %n - Name of the folder containing the session.
3. Click button(refer below screenshot)

Graphical user interface, text, application

Description automatically generated

Email Text: Using the below text

Folder: %n

Session: %s

Mapping: %m

Target Details: %t

Records Loaded: %l

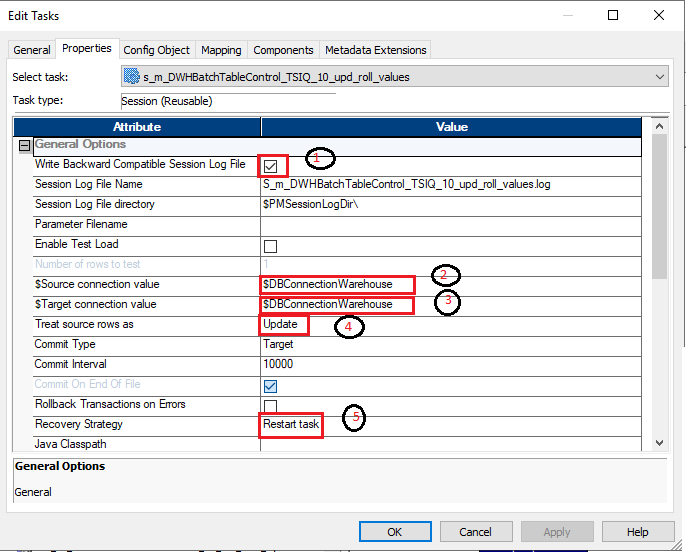
Records Rejected: %r

See attached log file for details on failure.

%g

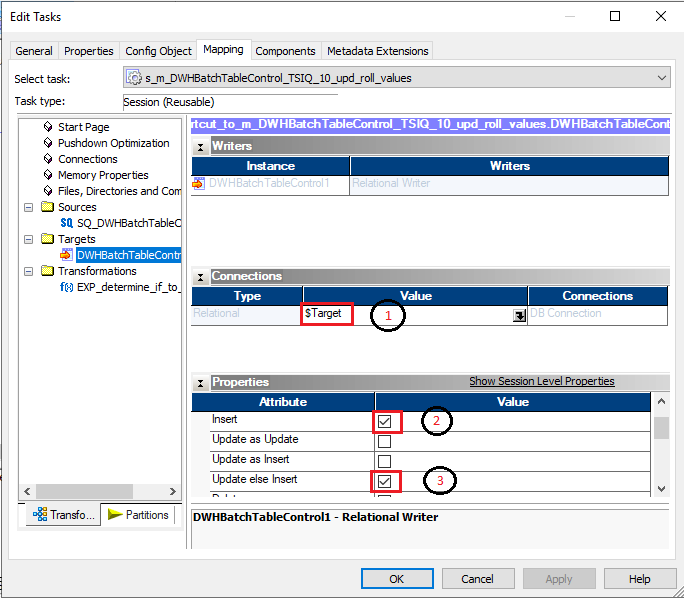
1. **s\_m\_DWHBatchTableControl\_TSIQ\_10\_upd\_roll\_values**

After creation of reusable session for mapping, set properties as mentioned in below screenshot.



1. **Write Backward Compatible Session Log File**: It will write the workflow or session logs into a text file (find the Session Log under server directory).
2. **$Source connection value** – set to $DBConnectionWarehouse, This variable is set by a parameter file.
3. **$Target connection value** – set to $DBConnectionWarehouse, This variable is set by a parameter file.
4. **Treat source rows as: Update** Integration Service marks all rows to update the target.
5. **Recovery Strategy:** Restart task - When the Integration Service recovers a workflow, it restarts each recoverable task that is configured with a restart strategy.

**Config Objects Tab –**Default Value



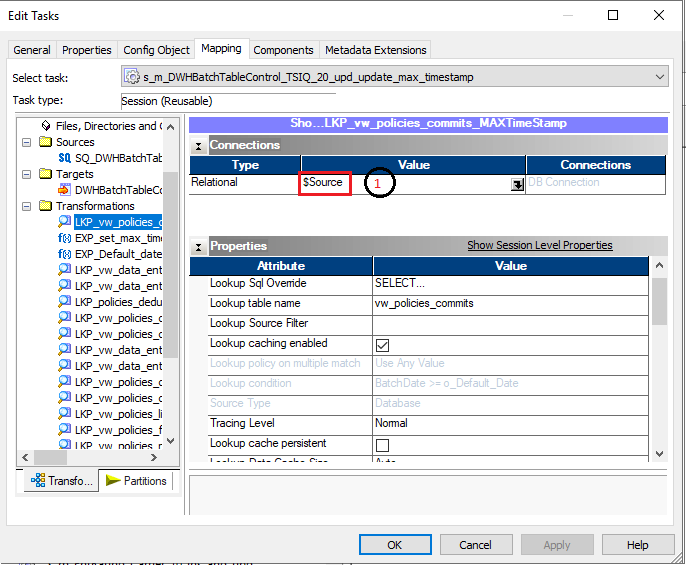
1. **$Target**: This variable is set by a parameter file.
2. **Insert:** If enabled, the Integration Service uses the target update mode (Update else Insert) to update rows.
3. **Update else Insert:** Integration Service updates existing rows and inserts other rows as if marked for insert, Must also select the Insert target option.
4. **s\_m\_DWHBatchTableControl\_TSIQ\_20\_upd\_update\_max\_timestamp**

After creation of reusable session for mapping, set properties as mentioned in below screenshot.

Graphical user interface, text, application, email

Description automatically generated

1. **Write Backward Compatible Session Log File:** it will write the workflow or session logs into a text file (find the Session Log under server directory).
2. **$Source connection value** – set to $DBConnectionStage, This variable is set by a parameter file.
3. **$Target connection value** – set to $DBConnectionWarehouse, This variable is set by a parameter file.
4. **Treat source rows as:** Update Integration Service marks all rows to update the target.
5. **Recovery Strategy**: Restart task - When the Integration Service recovers a workflow, it restarts each recoverable task that is configured with a restart strategy.



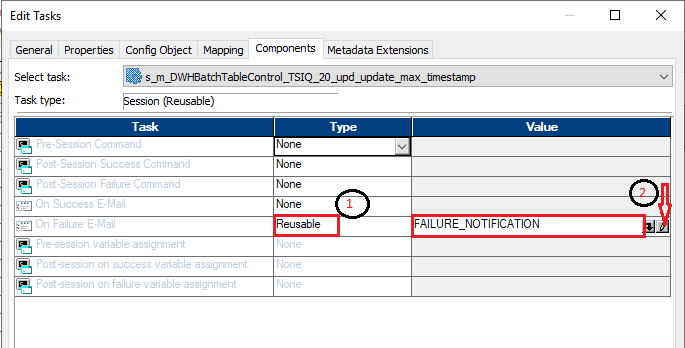
**1. $Source:** this variable is set by a parameter file

Graphical user interface, application

Description automatically generated

1. **$Target:** This variable is set by a parameter file
2. **Insert:** If enabled, the Integration Service uses the target update mode (Update else Insert) to update rows.
3. **Update else Insert:** Integration Service updates existing rows and inserts other rows as if marked for insert, must also select the Insert target option.

**Reusable EMAIL task:** FAILURE\_NOTIFICATION



1. Use **Reusable** Type

2. Click 🖉 Button (refer to the below screenshot)

Graphical user interface, text, application, email

Description automatically generated

1. **Email User Name:** $$FailureNotificationEmailGroup - This variable is set by a parameter file
2. **Email Subject :** Error %s Failed in %d - %n – Here
   1. %s - Session name.
   2. %d - Name of the repository containing the session.
   3. %n - Name of the folder containing the session.
3. Click button(refer below screenshot)

Graphical user interface, text, application

Description automatically generated

**Email Text:** Using the below text

Folder: %n

Session: %s

Mapping: %m

Target Details: %t

Records Loaded: %l

Records Rejected: %r

See attached log file for details on failure.

%g

1. **s\_m\_TSIQ\_policies\_commits\_10\_ins\_and\_upd**

After creation of reusable session for mapping, set properties as mentioned in below screenshot.

Graphical user interface, text, application, email

Description automatically generated

* 1. **Write Backward Compatible Session Log File:** it will write the workflow or session logs into a text file (find the Session Log under server directory)
  2. **$Source connection value** – set to $DBConnectionStage, This variable is set by a parameter file
  3. **$Target connection value** – set to $DBConnectionStage, This variable is set by a parameter file
  4. **Treat source rows as :** Update Integration Service marks all rows to update the target
  5. **Recovery Strategy:** Restart task - When the Integration Service recovers a workflow, it restarts each recoverable task that is configured with a restart strategy.

Graphical user interface, application

Description automatically generated

1. **$Target:** This variable is set by a parameter file
2. **Insert:** If enabled, the Integration Service uses the target update mode (Update else Insert) to update rows.
3. **Update else Insert:** Integration Service updates existing rows and inserts other rows as if marked for insert, must also select the Insert target option.

**Reusable Email task:** FAILURE\_NOTIFICATION

Graphical user interface, table

Description automatically generated

* + - 1. Use **Reusable** Type
      2. Click 🖉 Button (refer to the below screenshot)

Graphical user interface, text, application, email

Description automatically generated

1. **Email User Name:** $$FailureNotificationEmailGroup - This variable is set by a parameter file
2. **Email Subject :** Error %s Failed in %d - %n – Here
   1. %s - Session name.
   2. %d - Name of the repository containing the session.
   3. %n - Name of the folder containing the session.
3. Click button(refer below screenshot)

Graphical user interface, text, application

Description automatically generated

**Email Text:** Using the below text

Folder: %n

Session: %s

Mapping: %m

Target Details: %t

Records Loaded: %l

Records Rejected: %r

See attached log file for details on failure.

%g

Note: The above are everything that needs to be set on a session and will be applicable to every stage table session we builded

1. **s\_m\_DWHBatchTableControl\_TSIQ\_30\_upd\_batch\_success\_flag**

After creation of reusable session for mapping, set properties as mentioned in below screenshot.

Graphical user interface, text, application

Description automatically generated

1. **Write Backward Compatible Session Log File:** it will write the workflow or session logs into a text file (find the Session Log under server directory).
2. **$Source connection value** – set to $DBConnectionStage, This variable is set by a parameter file.
3. **$Target connection value** – set to $DBConnectionStage, This variable is set by a parameter file**.**
4. **Treat source rows as**: Update Integration Service marks all rows to update the target.
5. **Recovery Strategy:** Restart task - When the Integration Service recovers a workflow, it restarts each recoverable task that is configured with a restart strategy.

**Config Objects Tab –**Default Value

Graphical user interface, application

Description automatically generated

1. **$Target:** This variable is set by a parameter file.
2. **Insert:** If enabled, the Integration Service uses the target update mode (Update else Insert) to update rows.
3. **Update else Insert:** Integration Service updates existing rows and inserts other rows as if marked for insert, must also select the Insert target option.

Graphical user interface, application

Description automatically generated

1. **$Target:** This variable is set by a parameter file.
2. **Insert:** If enabled, the Integration Service uses the target update mode (Update else Insert) to update rows.
3. **Update else Insert:** Integration Service updates existing rows and inserts other rows as if marked for insert, must also select the Insert target option.

**Reusable Email task:** FAILURE\_NOTIFICATION

Graphical user interface, application, table

Description automatically generated

1. Use **Reusable** Type
2. Use Reusable **FAILURE\_NOTIFICATION** task

Graphical user interface, text, application, email

Description automatically generated

1. **Email User Name:** $$FailureNotificationEmailGroup - This variable is set by a parameter file
2. **Email Subject :** Error %s Failed in %d - %n – Here
   1. %s - Session name.
   2. %d - Name of the repository containing the session.
   3. %n - Name of the folder containing the session.
3. Click button(refer below screenshot)

Graphical user interface, text, application

Description automatically generated

**Email Text:** Using the below text

Folder: %n

Session: %s

Mapping: %m

Target Details: %t

Records Loaded: %l

Records Rejected: %r

See attached log file for details on failure.

%g

1. Workflow Creation

Creating the workflow by adding **wklt\_TSIQ\_Staging\_Nightly\_Batch**to the below workflow.

**wkfl\_TEMPORARY**

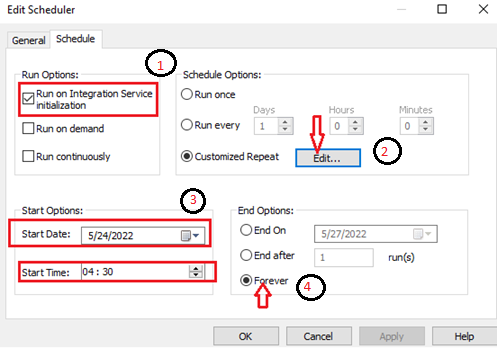
Graphical user interface, application

Description automatically generated

* + - 1. Workflow in Informatica is a set of multiple tasks connected with start task link and **triggers the proper sequence to execute a process**. When a workflow in Informatica is executed, it triggers a start task and other tasks connected in the workflow.
      2. **wklt\_TSIQ\_Staging\_Nightly\_Batch is** a workletofa set of multiple sessions connected with start task link.

1. Workflow Scheduler

Scheduled workflow will run every day at 4:30 AM EST except Saturday and Sunday for reference below.



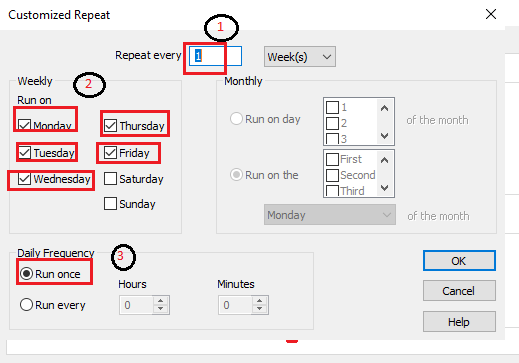
1. **Run On Integration Service Initialization:** The Integration Service runs the workflow as soon as the service is initialized. The Integration Service then starts the next run of the workflow according to settings in Schedule Options.
2. **Customized Repeat:** The Integration Service runs the workflow on the dates and times specified in the Repeat dialog box. When you choose Customized Repeat, you can schedule specific dates and times to run the workflow. The selected scheduler appears at the bottom of the page.
3. **Start Options:** Indicates when to start the workflow schedule. You can choose one of the following options:

**Start Date**: The date that the Integration Service begins the workflow schedule.

**Start Time**: The time when the Integration Service begins the workflow schedule.

1. **End Options**

**Forever**: The Integration Service schedules the workflow as long as the workflow does not fail.



1. **Repeat Every:** Enter the numeric interval would like the Integration Service to schedule the workflow Days.
2. **Weekly:** Run on Every Monday, Tuesday, Wednesday, Thursday, Friday
3. **Daily Frequency:** Run Once. The Integration Service runs the workflow one time on the selected day, at the time entered on the **Start Time** setting on the **Time** tab.