Shared Services Enterprise Data Warehouse

ETL Standards and Guidelines

|  |  |  |  |
| --- | --- | --- | --- |
| 1.0 | 2014-04-16 | Yogesh Pattapa | Initial Version |
| 2.0 | 2015-01-14 | Om Sachdev | Revised and added more sections |
|  |  |  |  |
|  |  |  |  |

Recommendations & Guidelines Summary

|  |  |  |
| --- | --- | --- |
| **Topic** | **Guideline** | **Location** |
| **Folder Structure** | The folder structure should take a data object / area (Subject Area/Project Name) approach. | 2.1.1 |
| **Naming Standards** | Mapping naming standard will be m\_{PROCESS}\_{SOURCE\_SYSTEM}\_{TARGET\_NAME}. The session naming standard is the same as the mapping name just replace the “m” with a “s”. The workflow naming standard is wf\_{DATA OBJECT DESCRIPTOR}. Please review section 2.2.5 for transformations names. | 2.1.4 |
| **Ports** | The ports on any transformation must be noted if altered especially in a lookup or expression. The input ports name must be suffixed with ‘\_in’. The output ports name must be suffixed with ‘\_out’. | 2.1.6 |
| **File Naming Standards** | A File must have a header with a specific pattern | 2.1.9 |
| **Parameters** | Developers should parameterize the code at mapping, session, and workflow levels but the parameter file location should only be stored at the workflow level. No hard coding is allowed in mapping code. A parameter file must be placed on the Informatica server. | 2.1.8 |
| **ETL Design** | The design phase should go through a business design, technical design, and a data mapping. | 3.1 |
| **SQL Override** | Only use SQL overrides if it will either result in a substantial performance gain or alter data types. If used Lookup and SQ Transformations must have “\_override” in the name. | 3.1.1 |
| **Migrations** | Migrating code from will follow the same change management process as any other application code migration. A code review with Data and Storage team is required for all new application or significant change in an existing application. | 3.1.2 |
| **Surrogate Keys** | Surrogate keys should be used to create a standard id. It is recommended to use Informatica instead of DB scripts for creating the surrogate key. | 3.1.3 |
| **Comments** | Code commenting should occur as often as possible and should be useful to other developers. | 3.1.4 |
| **Source Objects** | Source objects should be same structure of where they are sourced from and placed in Shared Folder. | 3.1.5 |
| **Target Objects** | Target objects should be same structure of target table environment and placed in Shared Folder. | 3.1.6 |
| **Data Objects & Transformations** | Expression Transformations should be used to “bookend” other transformations. Filter Transformations should not be used, use a Router instead. Aggregators should rarely be used and with very specific grouping criteria. When using a Lookup Transformations use connected lookups where possible. Rarely use a Joiner Transformation, use a SQL override in a Source Qualifier Transformation instead. | 3.1.7 |
| **Change Detection** | Change detection / MD5 - CRC (SCD-1 , SCD-2 ,SCD-3) | 3.1.8 |
| **Error Handling** | Developers will follow the Informatica error log to identify technical errors. The error log will be loaded to a table to be more useful. Specific Informatica error codes will need fixed on a case by case basis. | 3.3.1 |
| **Testing** | Recommended to Unit Test & Peer Review before moving code to project folder. | 3.3.2 |
| **Recovery** | A process to recover session/workflow from failure | 3.4 |
| **User Access** | Developers logging into the PowerCenter should have their own account. Do not use shared accounts. | 3.4.1 |
| **Versioning** | Versioning will need to be used during any code development. A tool will need to be used properly control versions. Code will be checked out when being developed and checked in when completed. | 3.5 |

[1 Introduction 5](#_Toc411435217)

[1.1 Purpose 5](#_Toc411435218)

[1.2 Scope 5](#_Toc411435219)

[1.3 Targeted Audience 5](#_Toc411435220)

[1.4 Assumptions 5](#_Toc411435221)

[2 ETL Standards & Procedures 6](#_Toc411435222)

[2.1 Coding Standards 6](#_Toc411435223)

[**2.1.1** **Folder Guidelines** 6](#_Toc411435224)

[**2.1.2** **Folder Structure** 6](#_Toc411435225)

[**2.1.3** **Workflow / Tasks naming standards** 7](#_Toc411435226)

[**2.1.4** **Mapping Naming Standards** 7](#_Toc411435227)

[**2.1.5** **Objects and Transformations Naming Standards** 8](#_Toc411435228)

[**2.1.6** **Ports naming standards** 9](#_Toc411435229)

[**2.1.7** **Connection Naming Standards** 9](#_Toc411435230)

[**2.1.8** **Use of parameter files** 10](#_Toc411435231)

[**2.1.9** **File naming standards** 12](#_Toc411435232)

[**2.1.10** **Out-bound Integrations for down streams** 13](#_Toc411435233)

[**2.1.11** **Job Control** 14](#_Toc411435234)

[**2.1.12** **Data Loading from Flat Files** 15](#_Toc411435235)

[**2.1.13** **Data Loading from Database Tables** 15](#_Toc411435236)

[**2.1.14** **Shell Scripts** 17](#_Toc411435237)

[3 Guidelines and Best Practices 18](#_Toc411435238)

[3.1 Mapping Design 18](#_Toc411435239)

[**3.1.1** **SQL Override** 19](#_Toc411435240)

[**3.1.2** **Code Migration Process** 19](#_Toc411435241)

[**3.1.3** **Surrogate Key Generation** 21](#_Toc411435242)

[**3.1.4** **Comments** 22](#_Toc411435243)

[**3.1.5** **Source Objects** 22](#_Toc411435244)

[**3.1.6** **Target Objects** 23](#_Toc411435245)

[**3.1.7** **Bulk Reader / Writer** 24](#_Toc411435246)

[**3.1.8** **Object Information and Usage** 27](#_Toc411435247)

[**3.1.9** **Change detection / MD5 - CRC (SCD-1, SCD-2, SCD-3)** 30](#_Toc411435248)

[3.2 Workflow/session best practices 32](#_Toc411435249)

[3.3 Error Handling & Testing 32](#_Toc411435250)

[**3.3.1** **Error/Failure Handling** 32](#_Toc411435251)

[**3.3.2** **Testing** 34](#_Toc411435252)

[3.4 Recovery 35](#_Toc411435253)

[3.5 Security 35](#_Toc411435254)

[**3.5.1** **User Access** 36](#_Toc411435255)

[3.6 Versioning 38](#_Toc411435256)

[**3.6.1** **Informatica** 38](#_Toc411435257)

[**3.6.2** **Microsoft Team Foundation Server (TFS)** 38](#_Toc411435258)

[4 Performance Optimization 38](#_Toc411435259)

[4.1 Performance Tuning Steps in Informatica 38](#_Toc411435260)

[4.2 Optimization Hints 39](#_Toc411435261)

[5 Teradata Connections 41](#_Toc411435262)

[6 Appendix 45](#_Toc411435263)

[6.1 Mapping Examples 45](#_Toc411435264)

[6.2 Recommended Fields 45](#_Toc411435265)

[**6.2.1** **Source Data Extract (Data Staging)** 46](#_Toc411435266)

[**6.2.2** **Debugging** 46](#_Toc411435267)

[**6.2.3** **Encryption** 48](#_Toc411435268)

[6.3 Metadata 48](#_Toc411435269)

[**6.3.1** **Informatica Technical Metadata** 48](#_Toc411435270)

[**6.3.2** **Business Metadata** 49](#_Toc411435271)

# Introduction

## Purpose

This document serves as the starting point for (ETL) developers. The document will provide the standard processes, methods and components which will be utilized in creating, testing and deploying ETL Integration interfaces. The guidelines in this document are ETL standards and therefore can be used across different ETL tools and processes

## Scope

The scope of this document is limited to the standards and guidelines for the ETL processes. It does not cover hardware setup, software installs, operations support and other activities not directly related to development.

## Targeted Audience

* ETL Developers
* ETL Administrators
* Database Administrators

## Assumptions

* Audience will have a basic knowledge of ETL processes
* Audience will have a basic understanding of Informatica PowerCenter
* The current architecture is relevant for Data Conversion, Data Staging, Operational Data Store, and Data Warehouse

# ETL Standards & Procedures

## Coding Standards

This section will focus on the coding standards specifically for using Informatica Power Center. These guidelines are to be used by ETL developers as the standards .The aim is to ensure that code conforms to corresponding standards and best practices.

### **Folder Guidelines**

Folders provide a way to organize the repository, including mappings, schemas, and sessions. The folder structure will be setup by the project Informatica Administrator. Each developer will work within their own respective development folder in the Informatica development environment. Source and target definitions will be created in a shared folder and shortcuts created into the developer folders. No source or target definitions will be created in the developer folders that will be transitioned to the next environment.

After setting the folder types there are many ways to organize the folders. A few of the best practices / approaches are: Development Environment, Object Type, and Location.

|  |  |
| --- | --- |
| **Folder Approach** | **Pro’s / Con’s** |
| **Development Environment** | The Development Environment approach creates one folder per environment (Dev, QA, etc.) and then Subfolders for specific projects/repositories. **Pros:** Good for organizing test code and areas. **Cons:** Does not easily list the source or data object. |
| **Object Type** | The Object Type approach creates one folder per object type being migrated (Vendor, Personnel, Benefits). This is the recommended approach. **Pros:** Allows numerous sources per area. Specific to each task. **Cons:** Doesn’t explicably list the source location. |
| **Location** | The Location approach creates folders based on the physical location of servers running the services. **Pros:** Know exact where data is sourced. **Cons:** Not useful if numerous servers are used. |

### **Folder Structure**

The folder structure comes from the start-up / instruction manual for Informatica. The approach taken aligns to the recommended Object Type approach. The following folder structure is planned to be used:

* Global Shared Folder
* Application Project Folder - This is planned to be broken down so that there is one folder per project (Line Of Business and Project). Deploy the corresponding Shared Folder first so that shortcuts can be made. Inside each Application Regular Folder there will be subfolders for: Sources, Targets, Transformations, Mapplets, Mappings, Sessions / Tasks, Workflows.
* Application Working Folder – Build team will have Application level working folders where the build will initiate. Once the build is completed, code will merge into “Application Project Folder” before migrating to Next environment. This folder will only exists in the development environment.

### **Workflow / Tasks naming standards**

* Workflow name: ***wf\_<session\_name>***

Example : ***wf\_s\_STG\_PERSON***

Standards for naming various tasks in the workflow:-

Please make sure that all your objects including have proper naming and comments.

|  |  |
| --- | --- |
| **Workflow Objects** | **Naming Conventions** |
| **Worklet Name** | wklt\_<meaningful name> |
| **Command Line Name** | cmd\_<meaningful name> |
| **Event Name** | evtw\_<meaningful name> for Event Wait Task and evtr\_<meaningful name> for Event Raise Task |
| **Decision Name** | dscn\_<meaningful name> |
| **Control Name** | cntrl\_<meaningful name> |
| **Email Name** | email\_<meaningful name> |
| **Assignment Name** | asgmt\_<meaningful name> |
| **Timer Name** | tmr\_<meaningful name> |

### **Mapping Naming Standards**

Please make sure that all your objects including have proper naming and comments.

* Example of mapping naming standard: ***m\_{PROCESS}\_{SOURCE\_SYSTEM}\_{TARGET\_NAME}***
* Example of mapping name: ***m\_STG\_WO\_PERSON***
* Process can be STG, ODS, LOAD, DM, RPT

### **Objects and Transformations Naming Standards**

|  |  |
| --- | --- |
| **Data Object / Transformation** | **Naming Standard** |
| **Source Object** | **Naming Standard:** A source object should be named src\_(SYSTEM\_ABRV)\_(TABLE\_NAME). **Example:** For source SAP vendor table would be src\_SAP\_LFA1. |
| **Target Object** | **Naming Standard:** A target object should be named trgt\_(SYSTEM\_ABRV)\_(TABLE\_NAME). **Example:** If the target is the personnel table in the data warehouse would be: trgt\_DW\_PERSONNEL. |
| **Expression Transformation** | **Naming Standard:** An Expression Transformation is named exp\_(description\_of what\_it\_does). **Example:** The name should leverage the expression and/or a name that describes the processing being done. Ex: exp\_FORMAT\_CUSTOMERS |
| **Lookup Transformation** | **Naming Standard:** A lookup transformation is named lkp\_TABLE\_NAME. **Example:** If a lookup transformation has sql override used within the object a suffix of “override” should be used to let other developers know that code is changing the function of the object. |
| **Aggregator Transformation** | **Naming Standard:** agg\_(FUNCTION/DESCRIPTION) that leverages the expression and/or a name that describes the processing being done.  **Example:** agg\_SUM\_BY\_MONTH |
| **Source Qualifier Transformation** | **Naming Standard:** sq\_(TRANSFORMATION) \_(SOURCE\_TABLEx) represents data from a source.  **Example:** sq\_STG\_PERSON. If a source qualifier has sql override used within the object a suffix of “override” should be used to let other developers know that code is changing the function of the object. |
| **Router Transformation** | **Naming Standard:** rtr\_(DESCRIPTOR) describes the process being filtered. **Example:** rtr\_EMPLOYEE\_TYPE |
| **Sequence Generator Transformation** | **Naming Standard:** seq\_(DESCRIPTOR) if using keys for a target table entity, then refer to that |
| **Normalizer Transformation** | **Naming Standard:** NRM\_{FUNCTION} that leverages the expression or a name that describes the processing being done. |
| **Mapplet** | **Naming Standard:** mplt\_(DESCRIPTION). Example: mplt\_Sales\_Summaries |
| **MQ Source Qualifier Transformation** | **Naming Standard:** mqsq\_(DESCRIPTOR) defines the messaging being selected. |
| **Custom Transformation** | **Naming Standard:** ct\_(TRANSFORMATION) name that describes the processing being done.  **Example:** ct\_FORMAT\_PERSON\_NAME |
| **Filter Transformation** | **Naming Standard:** fil\_ or filt\_(FUNCTION / DESCRIPTION) that leverages the expression or a name that describes the processing being done.  **Example:** fil\_VALID\_CUSTOMERS |
| **Joiner Transformation** | **Naming Standard:** jnr\_(DESCRIPTION) that can describe a homogeneous or heterogeneous join for a specific type. **Example:** jnr\_SALES\_FACTS\_To\_STG\_PRODUCT |
| **Rank Transformation** | **Naming Standard:** RNK\_(FUNCTION) that leverages the expression or a name that describes the processing being done. |
| **Sorter Transformation** | **Naming Standard:** srt\_(DESCRIPTOR).  **Example:** srt\_EMPLOYEE\_ID\_DESC |
| **Union Transformation** | **Naming Standard:** uni\_(DESCRIPTOR).  **Example:** uni\_PRODUCT\_SOURCES |
| **Update Strategy Transformation** | **Naming Standard:** UPD\_(UPDATE\_TYPE(S)) or UPD\_(UPDATE\_TYPE(S))\_(TARGET\_NAME) if there are multiple targets in the mapping.**Example:**upd\_UPDATE\_EXISTING\_EMPLOYEES |
| **Transaction control Transformation** | tct\_<DESCRIPTOR> |
| **XML Source Qualifier Transformation** | XMLsq\_<meaningful name> |
| **Application Source Qualifier Transformation** | Appsq\_<meaningful name> |
| **Midstream XML Parser Transformation** | XMLpr\_<meaningful name> |
| **Midstream XML Generator Transformation** | XMLgn\_<meaningful name> |
| **HTTP Transformation** | http\_<meaningful name> |
| **SQL Transformation** | sql\_<meaningful name> that describes the processing being done. |
| **Java Transformation** | java\_<meaningful name> that describes the processing being done. |
| **Identity Resoultion Transformation** | ir\_<meaningful name> that describes the processing being done. |
| **UnStructured Data Transformation** | unsdt\_<meaningful name> |
| **Data Masking Transformation** | dm\_<meaningful name> |
| **Session name** | s\_<mapping\_name> ….’s’ replaces the ‘m’ in mapping . example  m\_STG\_PERSON s\_STG\_PERSON |
| **Workflow name** | wf\_<session\_name> example wf\_s\_STG\_PERSON |

### **Ports naming standards**

|  |  |
| --- | --- |
| **Input Port** | **prefix ‘in\_’ example – in\_AMOUNT** |
| **Output port** | prefix ‘out\_’ example – out\_AMOUNT |
| **Variable Port** | prefix ‘v\_’ example – v\_ AMOUNT |

### **Connection Naming Standards**

Connections are managed through Informatica PowerCenter manager. The naming standards for connections are:

|  |  |
| --- | --- |
| **Connection Type** | **Naming Standard** |
| **Oracle connections** | ora\_ |
| **SQL Server connections** | sql\_ |
| **DB2 connections** | db2\_ |
| **PowerExchange connections** | pwx\_ |
| **PowerExchange for Databases** | pwx\_ora\_ / pwx\_sql\_ / pqx\_db2 / … |
| **Teradata connections** | td\_ |

### **Use of parameter files**

Developers should parameterize the code at mapping, session, and workflow levels but the parameter file location should only be stored at the workflow level. No hard coding is allowed in mapping code, even in the SQL Qualifier. If there is a need to hard code such as setting default values for a target table, use the parameter variables to hard code the values. In the mapping itself, a parameter connection will need to be used anywhere there is an override, lookup overrides and source qualifier overrides. Other reasons to use parameter files are that they are portable across environments (Dev, SIT, UAT, and Prod).

***Use of one parameter file per project is highly recommended***. Group related mappings and sessions within a workflow section in the parameter file, sort of nested approach.

A parameter file can have following sections

* [Global]

The parameters which are used by multiple workflows in a Project Folder are grouped and termed

as global parameters.

Examples:-

Database connections - $DBConnection\_Src=TD\_HRREPO\_STG

$DBConnection\_Tgt=TD\_HRREPO\_DM

Target File location – $OutputFilePath=/aaa/bbb/ccc -- full path

$InputFilePath=/aaa/bbb/ccc -- full path

* [Service: service name]

When there are multiple Integration services configured and used by objects within a single project folder, parameters can also be grouped based on Integration service that is going to use them.

Example –

* [folder name.WF:workflow name]

The parameters which are used by multiple sessions within a single workflow are

Grouped under this section. The parameters declared in this section are local to the folder and workflow specified.

Examples are default/Standard values and conventions

* [folder name.WF:workflow name.ST:session name]

Parameters that are local to a particular mapping and session are grouped in this section. Other sessions / workflows cannot use these parameters.

These parameters are customized for a particular session i.e they have situational use in the mapping flow.

Examples - suffix / prefix strings in string data types

Reference dates for CDC

A general parameter file should look in like this - 

* General guidelines for using Parameters

1- Keep the Parameter files as compact and precise as possible.

2 - Use parameters if you foresee the values of a variable changing in the future

3 - Values / entities that could be parameterized in general are Database connections,

Target / Source file paths, Global default values used across the application, Cut off - cut in dates for CDC.

4 - Maintain variables that need to be overridden as parameters.

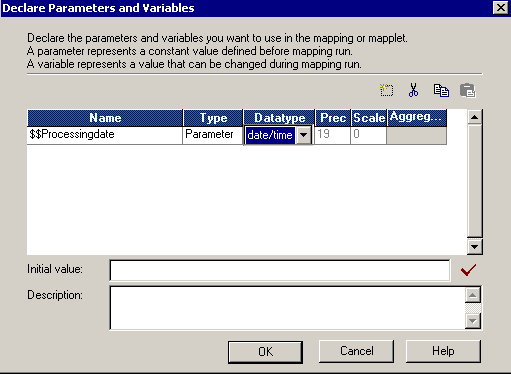
5 - Values like prefix strings eg: ‘AOLTW’, table names to populate audit fields need not be parameterized.

6 – File name should include source and target info. Ex – WB\_EDWSTG\_param.txt, EDWSTG\_WD\_param.txt

7 – Based on Project scope, check with ETL architect on param file decision.

Maintain separate param file for each integration layer. One file for each layer below -

* Source to EDW Staging
* EDW Staging to ODS
* ODS to EDW
* ODS to Reporting layer
* ODS to Data Mart
* ODS to Downstream



Ex: To declare a variable in a mapping

If initial values are not declared for the mapping parameter or variable then default values will be assigned based on the data type. The value that is defined for the parameter remains constant throughout the entire session. Create parameter files using a text editor such as WordPad or Notepad. Parameter files can contain only the following types of parameters and variables:

* Workflow variable
* Session parameter
* Mapping parameter and variables

### **File naming standards**

* The inbound and outbound file extracts should follow these guidelines
* The extension of a file can be .csv, .txt or .dat .
* The delimiter in the file can contain comma (,), pipe (|) or tilde (~ ) with an emphasis of quotes (“”) for all text fields
* The file must contain header. The header should contain the following information:
* Company
* Functional Area
* Timestamp
* Count of Total rows in the file
* Comments (Optional)
* The file name is broken down with the sections identified below with an underscore (\_) as a separator and filled with pound sign (#) if not applicable, lower case only.
* Company Name (6 Characters)
* Business Area (4 Characters)
* Function (10 Characters)
* Parts (2 Numeric)
* Miscellaneous (4 Characters)
* Date and Time Stamp (YYYYMMDDHHMM)

Example: xyzmod\_corp\_payrollded\_00\_0001\_201210010715.ext

pruavi\_ltc#\_remittance\_01\_####\_201204021130.ext

abcmed\_ltc#\_remittance\_02\_####\_201204021131.ext

### **Out-bound Integrations for down streams**

* **TBD -** Create a Log Table for outbound Integration. Each OB (outbound integrations for downstream systems) is required to log data in integration specific OB (outbound log) table in ODS schema, irrespective of incremental or full file extracts.
* OB tables will reside in ODS schema
* OB table naming format would be OB\_INTXXX, where XXX is integration number. ex- INT100 (outbound) will have OB table as OB\_INT100
* OB table will always have new rows inserted based on records extracted in an outbound file and row count will match
* OB table structure will follow outbound file structure (include all the data elements) in addition will have standard fields as follows

MD5\_CHCKSUM\_VAL

CREATE\_TS                                            - load date timestamp

LAST\_MOD\_PROC\_ID                         - Last run proc\_id

* OB table may be used as lookup to determine changes since last run for outbound extracts
* OB table will have historical extracts
* OB table may also be used for auditing purposes to determine what records were extracted in an outbound  file at any given point of time
* OB Archive/ purge strategy to be implemented based on the need

Example: - INT568 (wf\_INT568\_OUTBOUND\_YTD) generates an output file which has 3 columns and 2 rows as below -

1-INPUT\_TRANSACTION

2- PRIMARY\_TAXING

3- WORK\_STATE\_%

A separate outbound specific table has to be created in ODS schema with the following structure

***OB\_INT568***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| INPUT\_TRANSACTION | PRIMARY\_TAXING | WORK\_STATE\_WITHHOLDING\_% | MD5\_CHKSUM\_VAL | CREATE\_TS | LAST\_MOD\_PROC\_ID |
| ABCD | 20 | 2 | HYGF1234… | 2013-11-21 | 81 |
| WXYZ | 24 | 7 | POUH681… | 2013-11-21 | 81 |

***So there will be an additional OB\_INT Target in the ETL map of each outbound interface which will have a similar field to that of the actual Target along with the audit fields as shown above.***

### **Job Control**

All workflows Inbound / Outbound are subject to populate audit log information in job control

Tables-

There are 2 tables to store metadata information about the job runs:-

1 – PROJ\_MAS

This table has information on Project Name, Division Name, and Domain name

|  |  |
| --- | --- |
| ***Column Name*** | ***Definition / meaning*** |
| PROJ\_KEY | Sequence number used for reference in PROC\_CTRL\_TBL |
| PROJ\_NM | Name of the Project Folder in the ETL repository |
| DIVN\_NM | Name of the division – example ‘ABC’ |
| DOMAIN\_NM | Name of the domain or subject area--example HR |
| SBJCT\_AREA\_NUM | This is the number provided to each Integration.  Example – INIT191 - GL has number 191 |
| SUB\_SBJCT\_AREA\_NUM | This is sub subject area number attached with the integration like INT191\_1 , INT191\_2…these are the numbers (1,2,3tc) which says that there are multiple files coming in for the same integration |
| SBJCT\_AREA\_NM | It is the name of the interface. For the example ‘GL’, ‘POET’. |
| SRC\_SYS\_NM | Source system which provides input data |
| ERROR\_DETL\_CREATE\_TS | Timestamp |

This is a Prepopulated table.

2 – PROC\_CTRL\_TBL

The Table stores information about a workflow run

|  |  |
| --- | --- |
| ***Column Name*** | ***Definition / meaning*** |
| PROC\_ID | Sequence number |
| PROJ\_KEY | Foreign key referencing PROJ\_MAST table |
| PROC\_NM | Workflow name |
| PROC\_SUB\_NM | Mapping / session name |
| PROJ\_NM | Project Name |
| PROC\_TYP\_CD | Process that Populates the record. Example Informatica |
| PROC\_START\_TS | Process start Timestamp |
| PROC\_END\_TS | Process end Timestamp |
| PROC\_STAT\_CD | Defines whether Process was completed successfully or not |
| PROC\_STAT\_DESC | Description |
| COMMENT\_TXT | Comments |
| LAST\_MODFY\_TS | Timestamp |

For more details please refer attached documents –examples rows for the above tables:-



### **Data Loading from Flat Files**

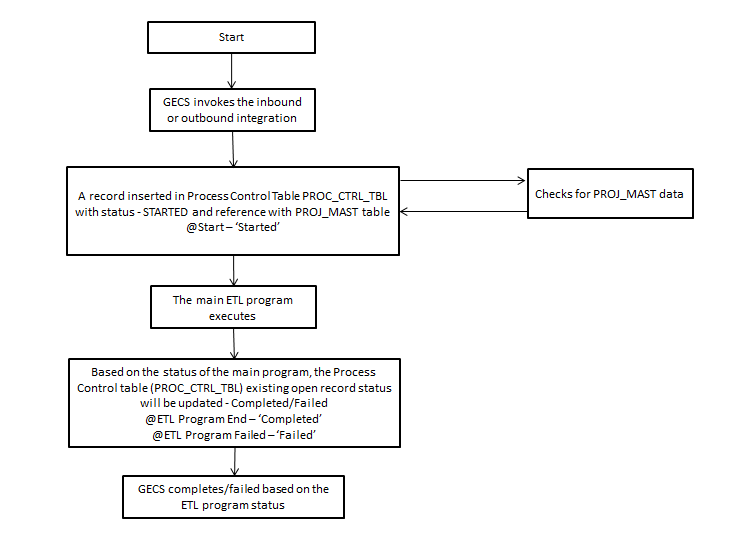
It’s an accepted best practice to always load a flat file into a staging table before any transformations are done on the data in the flat file.

* Always use LTRIM, RTRIM functions on string columns before loading data into a stage table.
* You can also use UPPER function on string columns but before using it you need to ensure that the data is not case sensitive (e.g. ABC is different from Abc)
* If you are loading data from a delimited file then make sure the delimiter is not a character, which could appear in the data itself. Avoid using comma-separated files. Tilde (~) is a good delimiter to use.

### **Data Loading from Database Tables**

Mappings which run on a regular basis should be designed in such a way that you query only that data from the source table which has changed since the last time you extracted data from the source table.

If you are extracting data from more than one table from the same database by joining them then you can have multiple source definitions and a single source qualifier instead of having joiner transformation to join them as shown in the figure below. You can put the join conditions in the source qualifier. If the tables exist in different databases you can make use of synonyms for querying them from the same database

In nut shell – Process lifecycle can be summarized as below

### **Shell Scripts**

Admin team has following a few generic scripts available and more generic scripts will be built upon request. If you have a need to build project specific custom script, then please make sure to consult with Admin team and get the design approval before building any new scripts. In general, please keep in mind the following points

* Try to make the scripts generic so that it can be used across the projects.
* No hardcoding in scripts
* No exposing passwords in scripts
* Make sure that Script saves the execution log for few runs at least.
* Script should have brief description on the functionality and proper indentation and comments throughout.
* There should be proper error handling and notification in your script.

# Guidelines and Best Practices

## Mapping Design

When designing mappings the developer should draw out a rough draft of the mapping that resembles a data flow diagram in DI spec (DLD). Draw out the different paths that the data flow can take and the different actions it will take on the target table. This diagram can act as a template for future mappings that will perform similar tasks. Example below:

Source Employee Table

Expression Transformation

Router Transformation

Data Validation Errors

Valid Employees

Data Error Target

Valid Data Target

An STM data map is an excel file which describes in detail where a field is sourced from and the exact target destination of the field.



***Downstream impact analysis***

* Identify the systems that are impacted by the ETL process and ETL loads.
* When ETL outbound data is unavailable
* ETL outbound data is erroneous
* There is delay in ETL outbound data and various other factors depending upon the nature of the project.

***Exit criteria for ETL deliverables:-ETL code reviewed and tested along with STM , DI Tech spec and UTC (3.4)***

Ex: Templates for Design Checklist, Technical Design Documents and Data Maps

### **SQL Override**

* Only use SQL overrides if it will result in a substantial performance gain
* SQL over rides should be made at mapping level and not session level to have SQL override in sync
* Whenever using a SQL override alter the transformation name to have “\_override” on the suffix so that another developer will know to look at the code
* Replace large Lookup tables (huge data) with joins in SQL overrides wherever possible
* Make sure the SQL override is generated by the transformation and other parts like where clause can be added later on for easy validation purpose
* Make sure to bring all the sources joined in SQL override SQL in the mapping for better visibility and code maintenance

### **Code Migration Process**

Migrating code from development to stage or production will follow the same change management process as any other application code migration. A code review with internal QA team is required for all new application or significant change in an existing application. Please follow these steps for code migration:

In case of a production target, we need approved change request otherwise we need approval with service now request.

Please not that we don’t generally support any manual change request in QA or Production for example, changing session property or changing a mapping or adding a command task etc… hence please take care of these kind of requirements in your deployment.

1. For new application or significant change in existing application, setup a meeting ***with DA Arch, DA Platform and Run teams*** for code review.
2. Open a change request with detail about code migration. This should include details on:
   1. Mention the source and target environments
   2. Source and Target Folders
   3. Parent object name (for example workflow name(s))
   4. Informatica deployment objects
      1. Label information
      2. Connection request information – please ensure connections are requested in proper format only. Requestor is responsible for providing all the information including username and password.
      3. Folder request information
      4. OS Profile request information
      5. Scheduling information
      6. Special instruction if any
   5. Server deployment objects
      1. Folder structure setup information
      2. Source/target files
      3. Parameter files
      4. Configuration files
      5. Parameter files
3. Open a related Remedy ticket for JOB SCHEDULER team with scheduling details if required

**Connection Request Format**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Connection  Type** | **Connection  Name** | **Username** | **Password** | **Host, Database, Port, SID** | **Code page** | **additional details for TPT  and Application connections** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Guidelines to use deployment groups -**

•**Static**: Static deployment is used in the scenarios where objects are not expected to change. Objects are added manually to the deployment group object.

•**Dynamic**: Dynamic deployment group is used where object change too often. A query is used in this case which can dynamically be associated with the latest version of the objects.

* 1. **Steps to create Dynamic Deployment Group:**
* Create Dynamic Deployment Group
* Create the Label
* Assign Objects to label
* Create query using the label
* Assign query to Deployment Group
* Copy deployment group to Target repository
  1. **Pre-Requisite: Creating labels**

* A label is a versioning object that you can associate with any versioned object or group of versioned objects in a repository
* Note: By default, the latest version of the object gets labeled
  1. **Advantages:**
     + Tracks versioned objects during development
     + Improves query results
     + Associates groups of objects for deployment
     + Associates groups of objects for import and export

### **Surrogate Key Generation**

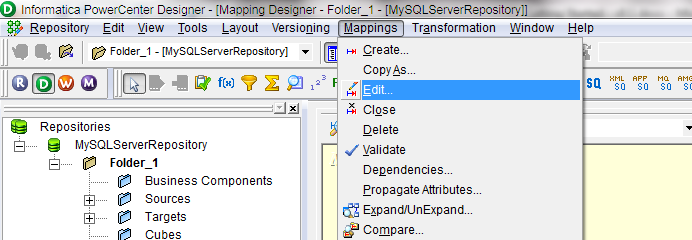
All primary surrogate keys are trigger-based sequence numbers. The new sequence numbers are generated programmatically when records are inserted into either through the Informatica PowerCenter or through the database (using something similar to Oracle’s sequence objects or a basic SQL script).

This method allows for a simple means to ensure that every record is unique in the table. Furthermore, it becomes easier to insert large volumes of data quickly as no lookup on the target table is needed to see if it already exists.

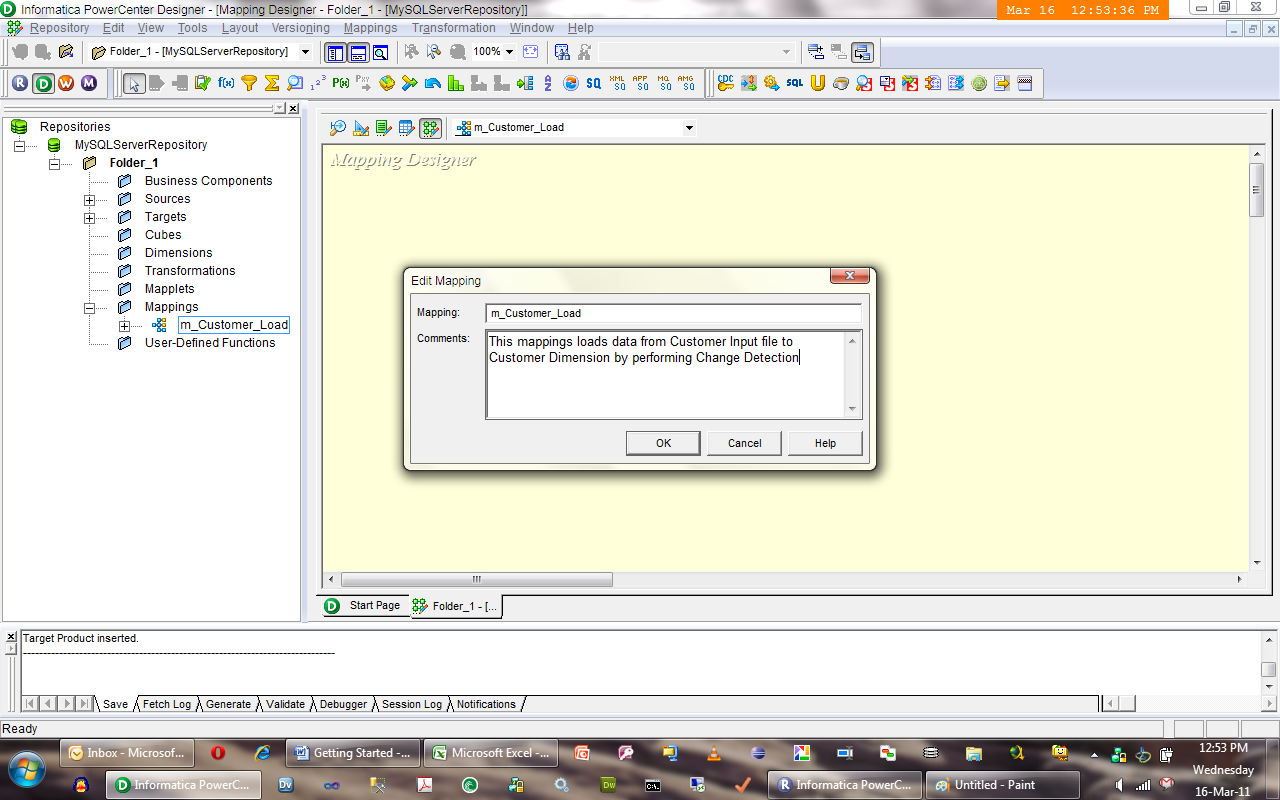
During the ETL load for parent and child relationship tables, parent tables are loaded with the surrogate key first and a separate ETL process retrieves the primary key from the parent table to load child tables.

### **Comments**

* Proper commenting in code is essential to developing an effective module. An Informatica mapping which is commented clearly and concisely is easier to read, analyze, debug, modify and test. While commenting code is an art rather than a science these guidelines should be followed to establish a system standard



Ex: Comments within Designer starts by clicking “Mappings” then “Edit”.



Ex: After selecting edit mapping the developer can add comments.

|  |
| --- |
| EXAMPLE 1: --BH – 4/29/2011 – Added a column and updated the name field so that a flag could be set which will be used in rtr\_PERSONNEL\_SCOPE. |
| EXAMPLE 2 if the SIR or defect number is known: // BH - 04/29/2011 - SIR1234 - ‘rounding of AT\_OPEN\_MO\_BAL\_v is changed from 4 to 2 to ensure accuracy |

### **Source Objects**

Source objects are the representations of sources within PowerCenter. These can be direct connections to a systems table, a flat file, or even just the source staging table. This section covers the guidelines for the table structure as well as the best practices when using source objects.

#### Table Structure

* The structure of the source objects should be identical from where the data is sourced
* The columns should be the same names and length whenever possible

**Informatica Source Objects**

* The source objects should be imported into a shared folder and all developers should create shortcuts to the table in the shared folder
* Sourcing table objects from source system will be the ETL Administrator responsibility
* Only the ETL Administrator should have read-write permission on the shared folder
* Using a shared folder allows numerous developers to work on the same tables without causing problems like unwanted editing, table definition out of sync
* Utilize single-pass reads
  + NOTE: Single-pass reading is the server’s ability to use one Source Qualifier to populate multiple targets
  + NOTE: For any additional Source Qualifier, the server reads this source. If there are different Source Qualifiers for the same source (e.g., one for delete and one for update/insert), the server reads the source for each Source Qualifier
* If processing intricate transformations, consider loading source flat file first into a relational database. This allows the PowerCenter mappings to access the data in an optimized fashion by using filters and custom SQL Selects where appropriate

### **Target Objects**

#### Table Structure

The structure of the target objects should look nearly identical to where and how the target systems are set up. In the case of the staging tables, the target tables will more reflect the source system as this is where the extracts will be populated. For the ODS, the target tables will more reflect the eventual data warehouse. Additional columns may be added for tracking purposes which is why the target table structure may not be exactly the same as the target environment (Ex: Adding a unique key, Timestamps, Version Stamps, etc.).

#### Informatica Target Objects

* The target objects should be imported into a shared folder and all developers should create shortcuts to the target tables in the shared folder
* All target table objects must be extracted into the shared object folder before they may be used for any mapping
* Sourcing table objects from the target database will be the ETL Administrator responsibility
* Only the ETL Administrator should have read-write permission on the shared folder
* A target instance must be named based on the operation it is subjected to

|  |
| --- |
| Example:, if a mapping has four instances of CUSTOMER table according to update strategy (Update, Insert, Reject, and Delete), the tables should be named as follows: CUSTOMER\_UPD, CUSTOMER\_INS, CUSTOMER\_DEL, CUSTOMER\_REJ. |

Target table scenarios:

|  |
| --- |
| Session Property Insert or Update – If the target table volume is large, use the session property “insert” or “update” and route one of the targets to a flat file. The flat file is used to insert or update within a separate mapping.  If the incoming source records usually require more updates than inserts into the target table, create two target instances: the target database instance for updating and the flat file instance for inserting. Then set the session property “target treat row as” to update (this can also be performed by using session partitions on this session since update is more expensive than insert). A second mapping will use the flat file from the first mapping to insert to the same target table. |
| If the incoming source records usually require more inserts than updates into target table time, create two target instances: the target database instance for inserting and the flat file instance for updating. Then set the session property “target treat row as” to insert. A second mapping will use the flat file from the first mapping to update the same target table. |
| Update Strategy– If the target table volume is small, use the “insert else update” update strategy (with or without target table lookup). |

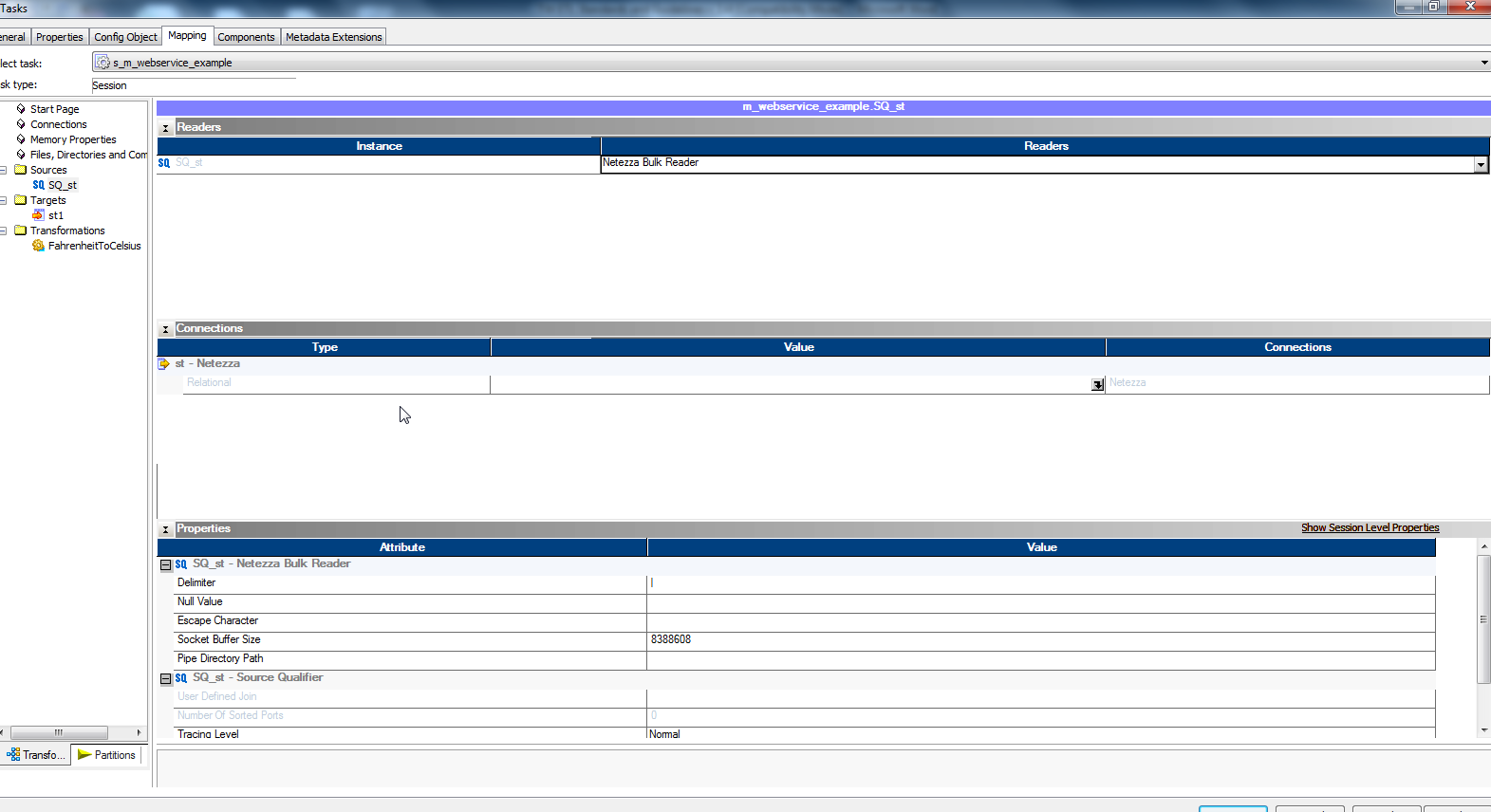
* The value for ’Treat source rows as’ session property must be = **DATA DRIVEN**
* The properties insert, update, delete etc pertaining to target instance must be checked /unchecked appropriately

### **Bulk Reader / Writer**

* Use Bulk Reader & Loader for Optimized performance while dealing with data of substantial size.
* Bulk Writer can be used to Insert , Update to the targets giving optimized performance as compared to the conventional Relational connection \*

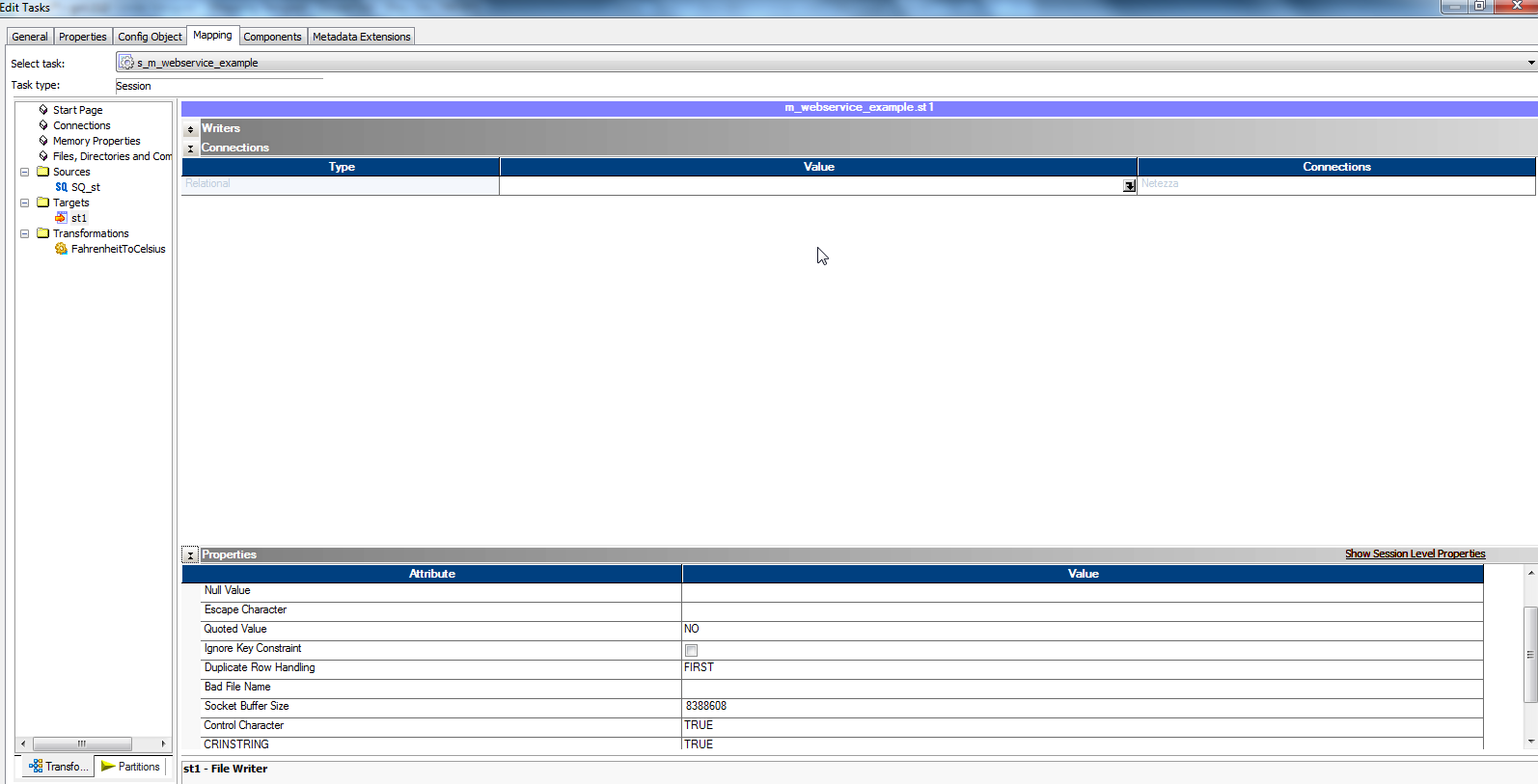
#### Configuring a Session with a Source

|  |  |
| --- | --- |
| ***Attribute Name*** | ***Description*** |
| Socket Buffer Size | Set the socket buffer size to 25 to 50 % of the DTM buffer size to increase session performance. You might need to test different settings for optimal performance. Enter a value between 4096 and 2147483648 bytes. Default is 8388608 bytes. |
| EscapeCharacter | Escape character of an external table. If the data contains NULL, CR, and LF characters in the Char or Varchar field, you need to escape these characters in the source data before extracting. Enter an escape character before the data. The supported escape character is backslash (\). |



#### Configuring a Session with a Target

|  |  |
| --- | --- |
| ***Attribute Name*** | ***Description*** |
| Socket Buffer Size | Set the socket buffer size to 25 to 50 % of the DTM buffer size to increase session performance. You might need to test different settings for optimal performance. Enter a value between 4096 and 2147483648 bytes. Default is 8388608 bytes. |
| EscapeCharacter | Escape character of an external table. If the data contains NULL, CR, and LF characters in the Char or Varchar field, you need to escape these characters in the source data before extracting. Enter an escape character before the data. The supported escape character is backslash (\). |
| Ignore Key Constraints | Ignores constraints on primary key fields. When you select this option, the PowerCenter Integration Service can write duplicate rows with the same primary key to the target. Default is disabled. The PowerCenter Integration Service ignores this value when the target operation is “update as update” or “update else insert.” |
| Duplicate Row Handling Mechanism | Determines how the PowerCenter Integration Service handles duplicate rows. Select one of the following values: - First Row. The PowerCenter Integration Service passes the first row to the target and rejects the rows that follow with the same primary key. - Last Row. The PowerCenter Integration Service passes the last duplicate row to the target and discards rest of the rows. Default is First Row. |



### **Object Information and Usage**

#### Frequently Used Data Objects

***Only Required ports should be used across the mapping .Unused Ports must be deleted .***

***The data types and lengths for the mapped fields should be consistent through out the mapping***

##### Expression Transformation

* Placing expressions after each object allows for easier code editing in the future, Having the ability to make changes to a mapping with little work is a good design
* A recommended practice is to always place an expression after the source qualifier to allow for the mapping to be edited later without disconnecting ports
* Use an Expression Transformation as a gathering location to make the mappings easier to read
* Create an Expression Transformation to bring all the ports together before going to the next transformation or target
* Note that Informatica processes the ports based on the priority as follows

1-Input ports 2-variable ports 3-output ports

And then in Top to down order in each of the above groups.

**Best Practices:**

* Calculate once, use many times. Avoid calculating or testing the same value over and over. Calculate a formula once in an expression and then set a True/False flag
* Use local variables to simplify complex calculations. Use variables to calculate a value used several times
* Watch the data types of fields and implicit conversions involved. Excessive data type conversions will slow the mapping

##### Filter Transformation

* If a filter is absolutely needed then use it early in the mapping as possible as it is an active transformation

##### Router Transformation

* Router Transformation tests data for one or more conditions and gives the option to route rows of data that do not meet any of the conditions to a default output group
* A Router Transformation should be used in place of the Filter Transformation since routers redirect unwanted data but allows filtered data to be stored if needed. Use a Router Transformation to separate data flows instead of multiple Filter Transformations
* Use a Router Transformation if more than one target requires some kind of filter condition

##### Aggregator Transformation

* Factor out aggregate function calls where possible. SUM (A) + SUM (B) can become SUM (A+B). Therefore, the server only searches through and groups the data once
* Minimize aggregate function calls by using “group by”
* Do place aggregator as early in the mapping as possible
* Provide Sorted input for better performance

##### Lookup Transformation

* Unconnected lookups should be used if the table being looked up is needed numerous times
* Check for look up policy on multiple match with data analyst
* Use dynamic lookup in scenarios when a single data set pulled in a run has multiple records wrt natural keys
* To ensure a match on a Lookup Transformation the developer may need to generate a SQL override and trim values in the lookup condition of leading and trailing spaces (Ex: RTRIM(LTRIM(fieldname))
* Size the Lookup Data and Index Cache Sizes and specify them as part of a tuning exercise

**Best Practices:**

* When using a Lookup Table Transformation, improve lookup performance by placing all conditions that use the equality operator = first in the list of conditions under the condition tab
* Avoid date time comparisons in lookup: replace with string
* When the source is large, cache lookup table columns for those lookup tables of 500,000 rows or less. This typically improves performance by 10 to 20 percent
* Use connected lookups where possible
* If caching lookups and performance is poor, consider replacing with an unconnected, un-cached lookup
* If the same lookup is used in multiple mappings or the same lookup is used more than once in the same mapping, take advantage of reusable transformation. In the case of using the same lookup multiple times in the same mapping, the lookup will only be cached once and both instances will refer to the same cache

##### Sequence Generator Transformation

* The Informatica Sequence Generator Transformation object should not be used when a table’s sequence numbers are also populated by another application.
* Do not reset the value unless the logic of the mapping demands
* Do not overwrite sequence generator values during migration from one environment to other unless mentioned explicitly
* Make sure that the start value of sequence is a higher value than 0 leaving decent number of holes for default records and space for unexpected future exceptions. (Example start value in HR reporting Wave 1 Project was 100 for each sequence generator)
* If only DB refresh is performed in DEV/STAGE from PROD then reset sequence generator equal to PROD
* Set optimum cache size for the sequence generator for better performance.

##### Joiner Transformation

* If using a Joiner Transformation, be sure to make the source with the smallest amount of data the master source
* When joining two sources, if both sources have same amount of records, select the master table as the one having more unique values in join column
* If the use of a joiner is necessary when loading parent and child tables then separate mappings must be developed: one to load the parent table and one to load the child table(s)

##### Source Qualifier Transformation

* Join data originating from the same source database
* Filter rows when the PowerCenter Server reads source data
* Specify sorted ports

##### Normalizer Transformation

* Use a Normalizer Transformation in a mapping to normalize a data stream where columns can be flipped into multiple rows
* A normalize is good for creating one-to-many records which is useful to break out a table to its individual columns
* Use a Normalizer Transformation to pivot rows rather than multiple instances of the same target

#### Rarely Used Objects

##### Update Strategy Transformation

* Do not code update strategies when all the rows to the target are update or insert
* Rejected rows from an update strategy are logged to the bad file. Consider filtering before the update strategy. Retaining these rows is not critical because logging causes extra overhead on the engine. Choose the option in the update strategy to discard rejected rows
* If an update override is necessary in a load, consider using a Lookup Transformation just in front of the target to retrieve the primary key. The primary key update will be much faster than the non-indexed lookup override

##### Sorter Transformation

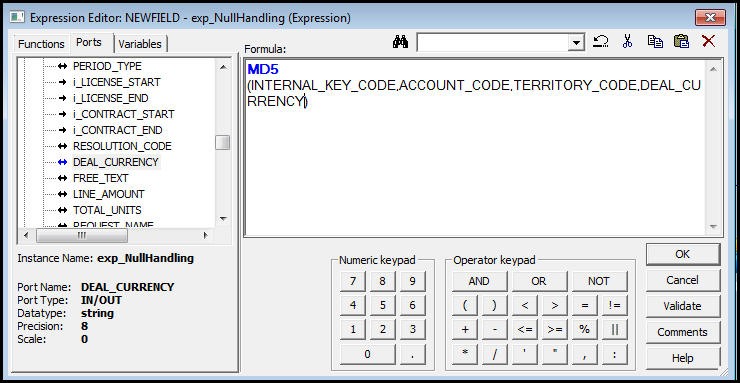
* Use a Sorter Transformation or hash-auto keys partitioning before an Aggregator Transformation to optimize the aggregate.

### **Change detection / MD5 - CRC (SCD-1, SCD-2, SCD-3)**

Challenge:-A record with lot of columns needs to be checked for changes over period of time.

Target dimension tables or similar tables should have a column to store MD5 checksum value which is populated with a hash value.

Use Informatica MD5 function to achieve this by passing all the column(s) that we intend to check for changes.



Change detection: - calculate MD5() for the same columns in the ETL map and perform a lookup on the Target table comparing the 2 MD5 values for a each key combination.

SCD 1 :- Update the existing record using update strategy (insert or update)

SCD 2:- Insert the current record and flag existing record in the table as inactive (depending on the design) . (This features Insert OR Insert and Update)

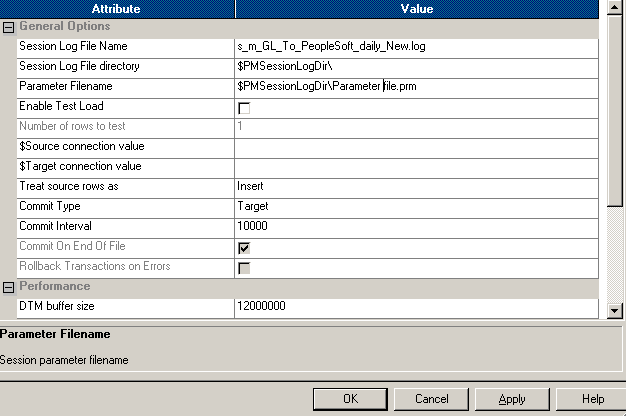
SCD 3:- Insert for a new record and Update the multiple columns that contain previous and current values.

## Workflow/session best practices

* Do not create reusable objects unless they are really needed to be reusable; especially sessions.
* Use stop on errors = 1
* Set commit interval appropriately for high performance
* Set ‘Fail parent if child fails’.
* Link sessions in parallel as much as possible
* Treat rows = data driven
* Check / uncheck target instance properties for insert , update , delete etc. accordingly
* Tracing level at session / transformation level must be ‘NORMAL’ which is the default.
* Do not use TRUNCATE TABLE option available in the session properties, rather create a separate dummy session with a pre / post SQL to truncate tables.
* Save workflow and session logs for at least 5 runs.
* Workflow log directory is set correctly as $pmrootdir/workflowlogs/
* Make sure that your workflow is HA aware; please select following properties at Workflow level.
  + Enable HA Recovery
  + Automatically recover terminated tasks – check this if you wish workflow to get automatically recovered, else not.
* There should be no hardcoded path in script/parameter file/workflow objects.
* For CDC Workflows, make sure that each workflow has only one session. If possible, logically group multiple CDC sources in one mapping. Following models are supported
  + One mapping with one source and target – One workflow
  + One mapping with multiple sources and multiple targets – One workflow
  + ~~Multiple mappings – One workflow~~ This is not technically possible, but not recommended
* For CDC sessions – make sure that following is true
  + Commit: source based
  + Commit on end of file: do not check
  + Recovery: resume from last checkpoint

Developers should request DBAs for truncate privilege over only certain required tables for data testing

The following sections explain how a developer would parameterize a workflow as well as properly name a workflow.



Ex: Parameter setting in a workflow

## Error Handling & Testing

### **Error/Failure Handling**

* Session logs and bad files
  + Create session logs / workflow logs and bad files with timestamp suffix by check marking the option in session properties.
  + Retain the logs for 15 days and purge them after the timeframe to maintain server space
* In the case of a record failure the developer should:
  + Identify the record(s) which failed.
  + Identify if the error was caused by database constraints or invalid data
  + Send on to appropriate contact for correction (Data Governance, DBA)
* In the case of a session or workflow error the developer should try to investigate and fix in order:
  + Workflow Level – Generally an issue with the param file
  + Session Level – Generally DB connection or owner is off
  + Mapping Level – Not usually a quick fix and takes time to investigate. The reason why is that the user will need to go through the debugging process

Common errors that a developer would need to walk through. For a complete listing please review the Informatica troubleshooting guide or help menu.

|  |  |
| --- | --- |
| **Error description** | **Possible Fix** |
| **Target field [XXXXX] does not exist in the object [XXXXX].** | The target table has been edited. Verify the changes and then reimport the target table into the shared folder. |
| **The connection test failed. The directory [XXXXX] does not exist.** | Check the parameter file to see if the connection is listed correctly. If the connection is listed correctly then check to see if the server is down. |
| **Execution terminated unexpectedly** | Generic error. Generally specific to issues with Informatica settings such as the log or cache is maxed out. |
| **Connection Error** | Check to see if parameter connections are set up correctly in the session. The error should list which transformation failed. Occasionally the error will just be “0”. This is a definite sign that the connection is wrong in the session or the parameter. |
| **Sequence Generator Transformation: Overflow Error** | The sequence generator has reached the end of its user specified value. The developer should look into the sequence generators length to verify the issue. Then the developer must decide to either expand the maximum value for the sequence or reset the sequence generator back to its initial value. Either decision will have a large impact so this issue should be raised up. |
| **Unique Record Constraint Failure** | Target tables id has to be unique but a duplicate ID attempted to load. The session will not fail but the individual record will be dropped. The record should be found in the error log. |
| **Invalid Mapping** | The saved mapping is invalid. Normally this occurs when a transformation is not connected to anything or in the case of an active transformation not all of the ports are connected. The error should specify the transformation. |
| **Data Value Overflowed or Too Large** | The precision settings cannot handle the amount of data being processed. The developer will need to edit the declared variable in the mapping to have a higher precision.  User defined lookup override query contains invalid characters – The listed lookup transformation contains an invalid SQL character. The developer will just need to go to the lookup transformation and correct the query. |
| **Error Truncating Target Table** | This is a user permissions error or the table being truncated is locked. In either scenario the DBA will have to fix it. |
| **Performance Error** | A session task takes a longer time than it should to load a small amount of data. A developer should check with the DBA of the source to see the DB performance. After that the developer should review any SQL used in a mapping. The developer should also try to run the mapping in another environment to see if their environment is down. |
| **Cannot find parameter file for the session** | The parameter file name has changed, has been moved or deleted, or didn’t exist at all. This will cause a workflow to fail immediately. The developer should check the parameter in workflow manager and verify that the file is correct on the server. |
| **Invalid lookup connect string** | The lookup transformations has an invalid location for its lookup table. The developer should check to see if the parameters are set up correctly in the lookup transformation as well as the session. If everything appears to be correct then the developer should verify that the table is still available within the Informatica Shared folder. If those appear to be fine then the developer should check the database to see if the table has been altered or dropped. |
| **Conversion from source type to target type is not supported** | There is an invalid data type conversion. This occurs when an objects data type is altered but there was not a valid conversion. Ex: A number field is changed to varchar in an expression transformation drop down menu but no command (TO\_CHAR) is called out |

### **Testing**

This section explains the standards and methodology for a developer to finalize their code. Before moving code from a developer’s folder to the project folder the code will need to go through a series of unit tests.

#### Preparing Test Data

* The data required for testing would enable a full dataset to be defined, ensuring that all possible cases will be tested.
* The developer will need to select a number of specific cases that they are looking to test.
* Retain test data, test scripts and test results to perform regression testing and possibly reuse the same data for Integration test in a neat and readable manner.

#### Unit Testing Procedure

* Every mapping must go through unit testing and a peer review.
* The final step of unit test will be a review and signoff by the ETL team lead on the test checklist.
* All functionality within each mapping should be unit tested.
* Developers are responsible for unit testing. The developer should check the following before and during Unit testing:

1. Create test cases based on Business requirements
2. Ensure the required source files are available in a directory
3. Validate the actual result with the expected result
4. Negative testing

Once a mapping is completed, unit tested and peer reviewed it will be moved from the individual developer folder into a common subject area folder. The folder structure will be setup by the project Informatica admin. This folder will be the project folder and will contain only final code.



Ex: Unit test script and peer review template

## Recovery

* Define a process to restart session/workflow in case of a failure
* Clean up steps as pre-requisite, if applicable
* Point of contact in build team to help research failure in case run team needs assistance

## Security

* Handled by Informatica administrators
* Implement security with the goals of easy maintenance and scalability
* When establishing repository security, keep it simple
* Although PowerCenter includes the utilities for a complex web of security, the more simple the configuration, the easier it is to maintain
* Securing the PowerCenter environment involves the following basic principles:
* Create users and groups
* Define access requirements
* Grant privileges and permissions

Other forms of security available in PowerCenter include permissions for connections. Connections include database, FTP, and external loader connections. These permissions are useful to limit access to schemas in a relational database and can be set-up in the Workflow Manager when source and target connections are defined. Occasionally, restriction changes to source and target definitions are needed in the repository. A recommended approach to this security issue is to use shared folders, which are owned by an Administrator. Granting read access to developers on these folders allows them to create read-only copies in their work folders. When implementing a security model, keep the following guidelines in mind:

* Create groups with limited privileges
* Do not use shared accounts
* Limit user and group access to multiple repositories
* Customize user privileges
* Limit the Administer Repository privilege
* Restrict the Workflow Operator privilege
* Follow a naming convention for user accounts and group names

### **User Access**

Individuals logging into the PowerCenter repository should have their own unique user account as Informatica does not recommend creating shared accounts. The following steps provide an example of how to establish users, groups, permissions and privileges in an environment.

* Identify users and the environments they will support (development, UAT, QA,

Production, production support, etc.)

* Identify the PowerCenter repositories in the environment (this may be similar to the basic groups listed in Step 1, e.g., development, UAT, QA, production, etc).

Identify what users need to exist in each repository

* Define the groups that will exist in each PowerCenter Repository. Repository privileges work in conjunction with folder permissions to give a user or group authority to perform tasks. Consider the privileges that each user group requires, as well as folder permissions, when determining the breakdown of users into groups. It is recommended to create one group for each distinct combination of folder permissions and privileges
* Assign users to groups. When a user is assigned to a user group, the user receives all privileges granted to the group
* Define privileges for each group and assign folder permissions. Informatica PowerCenter can also assign privileges to users individually. When a privilege is granted to an individual user, the user retains that privilege even if his or her user group affiliation changes. Example: a user in a Developer group who has limited group privileges needs to act as a backup Administrator when the current admin is not available. To do so the user must have the Administrator privileges. Grant the Administrator privilege to the individual user, not the entire Developer group

#### User Accounts

**Domain Administrator** – This account is configured on the initial Install. No regular user will be using this account after the root admin accounts are set up.

**Root Administrator** – This account is an admin console user with domain admin access. This user has the ability to create and restrict other accounts. Most security will be run through this user and developers will be dependent on this user to be able to have access. Since this user grants access this account should be heavily restricted and essentially works as a security Administrator. To summarize, here are the security related tasks an Administrator should be responsible for:

* Creating user accounts
* Defining and creating groups
* Defining and granting folder permissions
* Defining and granting repository privileges
* Enforcing changes in passwords
* Controlling requests for changes in privileges
* Creating and maintaining database, FTP, and external loader connections in conjunction with database Administrator
* Working with operations group to ensure tight security in production environment

Domain user –This is the developer account and will only have access to objects they have been granted access to but have no user create/edit ability. The developer account will have read/write access to their own developer folder which is where majority of the coding will occur. It is possible to assign read only permission to a developer so that they will have access to view finalized code as well as other developers code.

Data and Storage Architects group manages the Informatica platform in development, stage and production environments.

Currently there are four security groups for the platform in each environment where individual users log into the active directory. The settings are:

|  |  |
| --- | --- |
| **Group Role** | **Description** |
| **Administrators** | Has full access in all environments. Data and Storage group members are administrators. |
| **App\_Developers\_Group**   * **Benefits App\_Developers\_Group** * **Ect.** | Has sub group for each application and exists in Development only. Members of this group have Read, Write and Execute privilege. This group is meant for build team. |
| **App\_Support\_Group**   * **Benefits App\_Support\_Group** * **Ect.** | Has sub group for each application and exists in Development, Stage and Production environments. Members of this group have Read and Execute privilege on runtime objects. This group is meant for Run team. |
| **Release\_Managers – Only one release manager** | Members of this group have Read, Write and Execute privilege on runtime objects as well as Read, Write and Execute permission on folder objects for code migration. This group is meant for person migrating code from one environment to another. |

#### User on Boarding Process

Informatica authentication is integrated with Windows Active directory, so you need to have a domain account setup in each environment before you can request access to Informatica.

Please follow these steps for on boarding new application or person.

1. Setup a meeting with Data and Storage Architect to discuss application and/or user access requirements. The following items need to be discussed in this meeting.
   1. Group membership
   2. Folder permissions
   3. Import and export directory requirement and transfer of files from/to Informatica Shares
   4. Database source and target connectivity requirements
   5. Code review
2. Open a Remedy ticket assigned to Data and Storage Architect group with the following information
   1. Windows Domain where access is requested (BENHRIS, STAGEPRD, TWIDPRD):
   2. Windows Account Name:
   3. New Application Name:
   4. File Share Requested:
   5. Existing Application Name for which access is requested:
   6. Type of Privilege requested (Developer, Support or Release Manager).
3. Install Informatica Client software from <[\\Install\DBMS\Informatica\Client910](file:///\\Install\DBMS\Informatica\Client910)>
4. Register domains in client tool as listed in Developer Guide document available at

< [\\ Install\DBMS\Informatica\Documents](file:///\\ad.corp.aoltw.net\Install\DBMS\Informatica\Documents)>

1. Follow naming conventions for Informatica objects listed in the Developer Guide.

## Versioning

### **Informatica**

* Track objects during development – for adding Label, User, Last saved, or Comments parameters to queries to track objects during development.
* Associate a query with a deployment group – For creating a dynamic deployment group, associate an object query with it.
* Find deleted objects that can be recovered
* Find groups of invalidated objects that can be validate

#### Check-out

* Identify mapping, sessions, and workflows that need to be modified for code changes within the Integration folder

A developer should only check out the code that they will be working on as it allows other users to continue working instead of waiting for the code to be checked back in. Last a developer should always check out code being developed. Doing so will keep code consistent and will help with versioning if an issue arises.

#### Check-in

Once the object has been modified for code changes, the developer will then have to use the “check-in” feature to commit the changes to the repository. This is done by right clicking the checked-out object and then selecting the check-in option under versioning. Whenever a developer is done working on a mapping they must check the code in to allow other developers to continue working.

### **Microsoft Team Foundation Server (TFS)**

Microsoft Team Foundation Server (TFS) is software that is specific to visual studio. The tool to access the repository is Team Explorer but this tool cannot be executed in isolation, Visual Studio must be installed before installing Team Explorer. From there Team Foundation Server can be used as a basic repository.

Use TFS as a version control repository for SQL (DDL, DML)) and Unix scripts.

# Performance Optimization

## Performance Tuning Steps in Informatica

The goal of performance tuning is to optimize session performance by eliminating performance bottlenecks. To tune the performance of a session, first you identify a performance bottleneck, eliminate it, and then identify the next performance bottleneck until you are satisfied with the session performance. You can use the test load option to run sessions when you tune session performance.

The most common performance bottleneck occurs when the Informatica Server writes to a target database. You can identify performance bottlenecks by the following methods:

* **Running test sessions.** You can configure a test session to read from a flat file source or to write to a flat file target to identify source and target bottlenecks.
* **Studying performance details.** You can create a set of information called performance details to identify session bottlenecks. Performance details provide information such as buffer input and output efficiency.
* **Monitoring system performance.** You can use system-monitoring tools to view percent CPU usage, I/O waits, and paging to identify system bottlenecks.
* Once you determine the location of a performance bottleneck, you can eliminate the bottleneck by following these guidelines:
* **Eliminate source and target database bottlenecks.** Have the database administrator optimize database performance by optimizing the query, increasing the database network packet size, or configuring index and key constraints.
* **Eliminate mapping bottlenecks.** Fine-tune the pipeline logic and transformation settings and options in mappings to eliminate mapping bottlenecks.
* **Eliminate session bottlenecks.** You can optimize the session strategy and use performance details to help tune session configuration.
* **Eliminate system bottlenecks.** Have the system administrator analyze information from system monitoring tools and improve CPU and network performance.

If you tune all the bottlenecks above, you can further optimize session performance by partitioning the session. Adding partitions can improve performance by utilizing more of the system hardware while processing the session.

Because determining the best way to improve performance can be complex, change only one variable at a time, and time the session both before and after the change. If session performance does not improve, you might want to return to your original configurations.

For more information check out the Informatica Help from any of the three informatica client tools.

## Optimization Hints

* Store all Sequence Generators as re-usable (even if they won’t be reused) so they end in the Transformation section of the Project Folder. This will make it easier to find and re-set the sequences if necessary. However, when a sequence is marked re-usable the cached value can’t be zero. Make sure you don’t keep the cached value at 1 because it will access the Repository for every row; instead make the cache value 100.
* Optimize Query. Give Hints, add indexes, analyze tables, Create index on order by and group by columns.
* Filter data in source side.
* Single-pass reading. Use router, decode and other transformation.
* Consider more shared memory for large number of transformations. Session shared memory at 40MB should suffice.
* Calculate once, use many times.
* Only connect what is used.
* Watch the data types.
* The engine automatically converts compatible types.
* Sometimes conversion is excessive, and happens on every transformation.
* Minimize data type changes between transformations by planning data flow prior to developing the mapping.
* Facilitate reuse.
* Plan for reusable transformations.
* Use variables.
* Use mapplets to encapsulate multiple reusable transformations.
* Only manipulate data that needs to be moved and transformed.
* Delete unused ports particularly in Source Qualifier and Lookups. Reducing the number of records used throughout the mapping provides better performance
* Use active transformations that reduce the number of records as early in the mapping as possible (i.e., placing filters, aggregators as close to source as possible).
* Select appropriate driving/master table while using joins. The table with the lesser number of rows should be the driving/master table.
* When DTM bottlenecks are identified and session optimization has not helped,
* Use tracing levels to identify which transformation is causing the bottleneck (use the Test Load option in session properties).
* Utilize single-pass reads.
* Single-pass reading is the server’s ability to use one Source Qualifier to populate multiple targets.
* For any additional Source Qualifier, the server reads this source. If you have different Source Qualifiers for the same source (e.g., one for delete and one for update/insert), the server reads the source for each Source Qualifier.
* Remove or reduce field-level stored procedures.
* If you use field-level stored procedures, Power Center has to make a call to that stored procedure for every row so performance will be slow.
* Lookup Transformation Optimizing Tips.
* Indexing on lookup tables.
* In LOOKUP never pass NULL value for input port instead use default value like –999.
* Use SQL Overrides whenever possible to limit the number of rows returned.
* Only Cache lookup tables if the number of lookup calls is more than 10-20% of the lookup table rows. For fewer number of lookup calls, do not cache if the number of lookup table rows is big. For small lookup tables, less than 5,000 rows, cache for more than 5-10 lookup calls. Remove Cache if low number of rows coming in (if high rows in LKP). When caching is required, only select the data needed for the lookup. For example, only select current records when caching tables.
* Reuse cache when used by 3 or more sessions in a single job stream AND it takes greater than 15 minutes to create the cache file
* When your source is large, cache lookup table columns for those lookup tables of 500,000 rows or less. This typically improves performance by 10-20%. Do this by add condition logic to the SQL override whenever possible.
* The rule of thumb is not to cache any table over 500,000 rows. This is only true if the standard row byte count is 1,024 or less. If the row byte count is more than 1,024, then the 500k rows will have to be adjusted down as the number of bytes increase (i.e., a 2,048 byte row can drop the cache row count to 250K 300K, so the lookup table will not be cached in this case).
* When using a Lookup Table Transformation, improve lookup performance by placing all conditions that use the equality operator = first in the list of conditions under the condition tab.
* Replace lookup with decode or IIF (for small sets of values).
* If caching lookups and performance is poor, consider replacing with an unconnected, uncached lookup.
* UN-connected lookups should be used when less than 30% of the input rows need to be looked up for a value.
* For overly large lookup tables, use dynamic caching along with a persistent cache. Cache the entire table to a persistent file on the first run, enable update else insert on the dynamic cache and the engine will never have to go back to the database to read data from this table. It would then also be possible to partition this persistent cache at run time for further performance gains (Caution: Use only with approval).
* Review complex expressions.
* Examine mappings via Repository Reporting and Dependency Reporting within the mapping.
* Minimize aggregate function calls.
* Operations and Expression Optimizing Tips
* Numeric operations are faster than string operations.
* Optimize char-varchar comparisons (i.e., trim spaces before comparing).
* Operators are faster than functions (i.e., || vs. CONCAT).
* Optimize IIF expressions.
* Avoid date comparisons in lookup; replace with string.
* Test expression timing by replacing with constant.
* Use Flat Files
* Using flat files located on the server machine loads faster than a database source located in the server machine.
* Fixed-width files are faster to load than delimited files because delimited files require extra parsing.
* If processing intricate transformations, consider loading first to a source flat file into a relational database, which allows the Power Center mappings to access the data in an optimized fashion by using filters and custom SQL selects where appropriate.
* If working with data that is not able to return sorted data (e.g., Web Logs) consider using the Sorter Advanced External Procedure.
* Use a Router Transformation to separate data flows instead of multiple Filter Transformations.
* Use a Sorter Transformation or hash-auto keys partitioning before an Aggregator Transformation to optimize the aggregate. With a Sorter Transformation, the Sorted Ports option can be used even if the original source cannot be ordered.
* Use a Normalizer Transformation to pivot rows rather than multiple Instances of the same Target.
* When using a Joiner Transformation, be sure to make the source with the smallest amount of data the Master source.
* If an update override is necessary in a load, consider using a lookup transformation just in front of the target to retrieve the primary key.
* The primary key update will be much faster than the non-indexed lookup override.
* Tune Session Parameters
* Buffer Block Size (at least 20 rows at a time)
* Enable or Disable lookup cache
* Increase cache size (data and index).
* For data (column (s) size + 8) \* Number of Rows.
* For index (column (s) size + 16) \* Number of rows
* Increase commit interval
* Remove any verbose settings made in transformations for testing. Also, avoid running the session on verbose for large set of data.
* Monitor the sessions and document it.

# Teradata Connections

The connection should be chosen as per the below standards and based on the DBA’s advice.

The Naming conventions to be followed are

ODBC - <call letter>\_<conn type>\_<user type> eg : PRIME\_REL\_LOAD01

TPT - <call letter>\_<conn type>\_<usertype>\_<operator> eg : CDAS\_TPT\_STAGE\_UPD

|  |  |
| --- | --- |
| **Connection** | **Suggested Load Ranges** |
| TPT LOAD (FLOAD) | > 100,000 rows on empty table |
| TPT UPDATE (MLOAD) | > 25,000 rows on populated table |
| TPT STREAM (TPUMP) | > 1,000 rows < 250,00 |
| ODBC | <1000 rows |

The Run strategy should be discussed with the DBA on how many sessions/connections to be run in parallel with the above set of combinations.

**Connection details Teradata**

##### Load (Fast Load)

* The Load operator is a consumer operator that uses Teradata FastLoad protocol to load a large volume of data at high speed into an empty table on the Teradata Database. Use this operator to initially load tables into the Teradata Warehouse. Multiple parallel instances can be used to improve the performance of the load.
* The Load operator does not support update, select, and delete operations on the target table. The data sources for the Load operator can come from anywhere, such as a flat file, a queue, an ODBC-compliant source, an access module provided by Teradata, a named pipe, or a customer access module created by an end user, to name a few.
* Features:
  + Fastest way to load data into an empty table
  + Moving data in bulk (block) fashion
  + Multi-session
  + Recommended for more data more than 100,000 rows
* Restrictions:
  + Target table must be empty
  + No secondary index on the target table
  + No join index on the target table
  + No foreign key on the target table
  + If the job fails the table needs to be re-created
* Considerations:  
  The LOAD operator (Teradata FASTLOAD utility) should be used when there is a need to load a very large volume of data quickly in the EDW into an empty table. This is generally for load strategy utilizing a ELT approach, where data is loaded into a staging table , transformed in the database, and then coalesced with data from the production table, via a rename process.

##### Update (Multi Load)

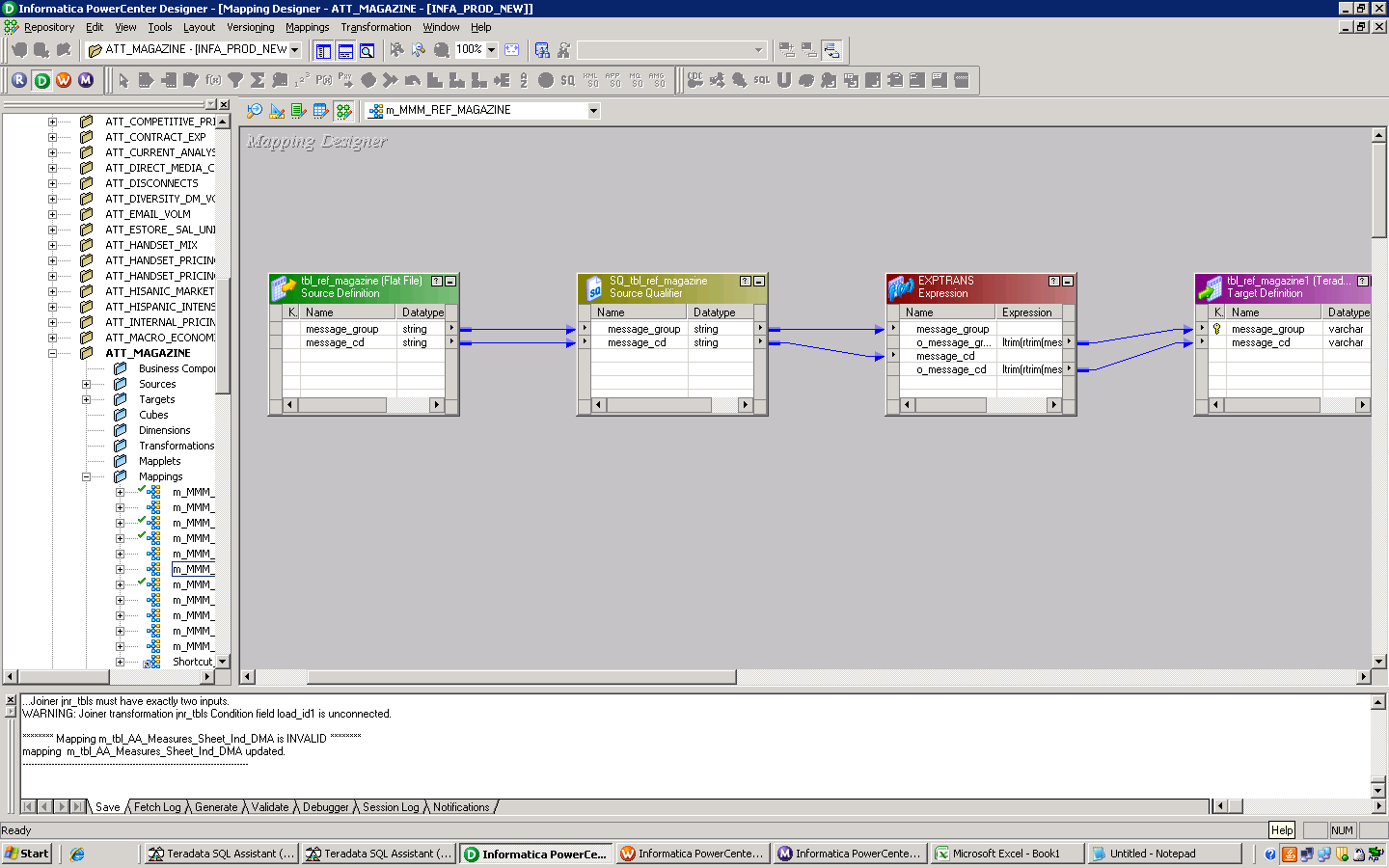
* The Update operator is a consumer-type operator that emulates the Teradata MultiLoad utility to load a large volume of data at high speed into up to five tables on the Teradata Database. Use this operator to maintain tables in the Teradata Warehouse.
* The Update operator uses multiple sessions to perform highly scalable and parallel inserts, updates, deletes, and upserts into up to five new or preexisting Teradata tables in a single pass. The data sources for the Update operator can come from anywhere, such as a flat file, a queue, an ODBC-compliant source, an access module provided by Teradata, a named pipe, or a customer access module created by an end user.
* Features:
  + Support upsert/delete/insert operation
  + Similar to Load, multi-session bulk load
  + Two staged operation: data acquisition and application phase.
  + More flexible than Load, but slower
  + Update is recommended for load thousands to millions records to non-empty tables
* Restrictions (common ones):
  + Join index/foreign key/unique secondary index/hash index, Drop USI and Create USI needs to be incorporate in scripts.
  + Row size: approximately 64k
  + PPI (partitioned primary index tables):
  + For DELETE/UPDATE operation, all values of primary index + partitioned columns needs to be specified.
  + NO update on the partitioned column set
  + NO primary index columns update

##### Stream (TPUMP)

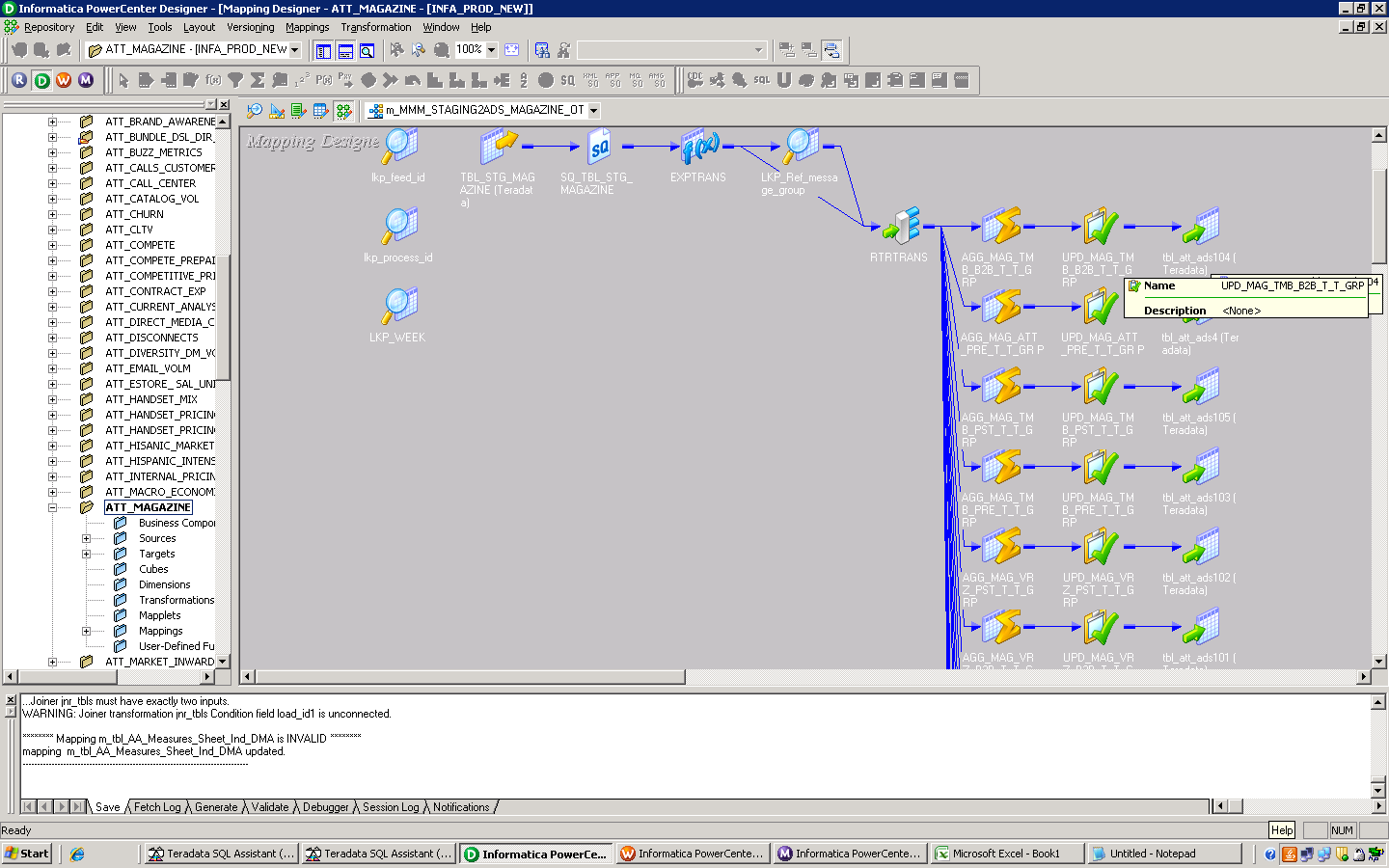
* The Stream operator is a consumer-type operator that emulates the Teradata TPump utility to perform high-speed DML transactions (SQL INSERT, UPDATE, DELETE, or UPSERT) in a near-real-time mode to a table (or tables) while queries are performed on the table (or tables).
* The Stream operator allows parallel inserts, updates, deletes, and upserts to empty or preexisting Teradata Database tables. The Stream operator uses Teradata SQL within the Teradata PT infrastructure in its communication with the Teradata Database. It uses multiple sessions to load data into one or more empty or preexisting tables.
* The Stream operator provides an alternative to the Update operator for the maintenance of large databases under control of a Teradata Database. The Stream operator can be used to maintain tables in the Teradata Warehouse, as can the Update operator. However, the Stream operator allows access to the database and does not lock the target tables that are updated, so that interactive read and write activities can be performed concurrently. The Stream operator supports many of the same features and facilities as the Teradata TPump standalone utility.
* Features:
  + Parallel data load activity
  + Macro based operation for multiple statements packaged together.
  + Serializable execution operation of statements
  + Handles streams of data with rich set of operation: insert/udpate/delete/upsert
  + Not quite fast, but better than insert/update/delete
  + All operations are primary index based
  + Need careful tuning (speed, primary index etc.)
  + Ideal for Mini-batch and small amounts of data to be loaded, recommended data volume is thousands of records to a few couple hundred thousand at a time.
* Restrictions:
  + Jobs have tunable parameters that need to be discussed and tested with Teradata DBAs to ensure optimal performance.

# Appendix

## Mapping Examples



Ex: Mapping of basic extract from flat file.



Ex: Mapping using one source and populating many different target tables. This occurs by using a Router Transformation.

## Recommended Fields

The following fields are recommended to be added onto the end of every table, regardless of whether these fields are found in source or not.

* Surrogate Key – A sequenced generated ID used to track records across tables. This is useful when working with data coming from different systems
* Time Stamp – Records when the mapping was last run. This is crucial when tracking data extracts and allows the developer to know if the data is up to date
* Source Name – This column is needed if numerous sources are used. This can also be used to list the name of the flat file which allows developers know if the correct file was used
* Mapping Stamp – This column is needed if more than one mapping is used to populate a table. If so it is recommended to list the mapping where the data came from
* Job\_ID - A distinct ID created to track the exact run of a mapping

### **Source Data Extract (Data Staging)**

Staging either takes the form of a series of flat file dumps or as tables in a database and usually does not mirror the table structures in the target data structures. The tables do usually follow some de-normalized design patterns. The staging area will be truncated on every run. It is advisable to have the structures of the staging area as a replica of source systems. Staging area data structures can be relational tables or flat files.

There are two types of data staging architecture, one layer and two layers. In the one layer architecture, if the source systems are online with small volume or offline systems (files) then there should be a single staging area. In the two layers architecture the source systems are online systems and a detail level of quality check needs to be performed at staging. This approach will help in source system contention, impact on resources and when there is smaller window to access the source system with large volume of data. The first staging area can be used to park the raw data (delta) from source system which will provide the source system snapshot extracted. The staging area can hold one/two days of data based on requirements. The second staging area will be used to store the cleansed and standardized data. This staging area can hold the delta or historical data based on business needs. Retaining historical data helps in data analysis for future use without dependency on source systems.

Based on the current planned data architecture the recommended approach is to use one layer data staging as the Operational Data Store (ODS) will take the place of the function of the second layer. The data staging area will be used to park the raw data from source system which will provide the source system snapshot extracted. The staging area will hold only the previous extract and will be truncated during each extract. The ODS will then be used for historical and quality purposes.

### **Debugging**

#### Informatica debugger

A developer can debug a valid mapping to gain troubleshooting information about data and error conditions. A developer can configure and run the debugger from within the Mapping Designer. For better debugging it is better to have many mappings than many paths in one mapping. Having many mappings instead of many paths allows the developer to pinpoint a bug more quickly. To debug a mapping, the developer will need to configure and run the Debugger from within the mapping designer.

#### Configure the Debugger

The Debugger wizard in the mapping designer is used to configure the debugger for the mapping. The mapping must be configured to a session prior to executing the mapping in debugger mode. Upon invoking the debugger wizard, the developer must choose the relevant session for a mapping. The developer can select the following different debugger session types:

* Existing non-reusable session: Uses existing source, target, and non-reusable session configuration properties. The Debugger does not suspend on error
* Existing reusable session: Uses existing source, target, and session configuration properties. When the developer runs the Debugger, it creates and runs a debug workflow for the reusable session
* Create a debug session instance: The developer can create source, target and session configuration properties on their own through Debugger Wizard. It creates and runs a debug workflow for the session

#### Create Breakpoint

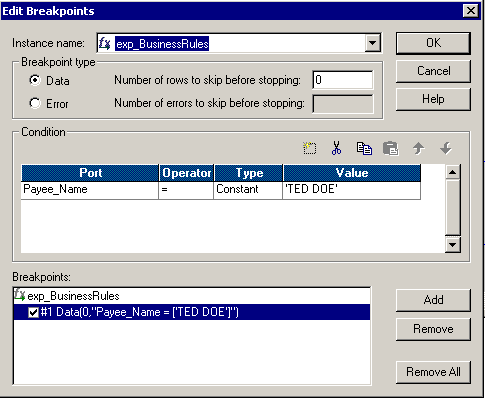
Before running the debugger, specify breakpoint conditions in Breakpoint Editor using the mapping designer. The breakpoint condition is where Integration service pauses if condition evaluates to true. Breakpoints are set using the Breakpoint Editor in the mapping designer. The debugger pauses when a breakpoint evaluates to true and the transformation data can be reviewed and or modified. To create the breakpoints, specify the breakpoint parameters in the following order:

1. Select the Instance Name
2. Select the breakpoint type
3. Enter the condition

Selecting Instance Name: While selecting the instance name, the breakpoint can be created for an individual transformation or for Global condition.

* Transformation Instance: Select a transformation from a mapping to configure the breakpoint. The Integration service evaluates the breakpoint condition when it processes the particular transformation
* Global Instance: Select the instance name as Global to configure a breakpoint condition that the Integration Service evaluates when it processes each transformation

The recommended approach is to set breakpoints wherever data is being sourced or altered (Ex: Source Qualifier, Lookup, and Router Transformations). The debugger will pause upon meeting the condition defined. In the example below, a breakpoint is defined for Payee\_name = “TED DOE”. The debugger will pause once the matching record is found. The developer can also edit a breakpoint by altering the condition.



Ex: Screenshot illustrates setting a breakpoint for a specific condition

### **Encryption**

Encryption should be based off of security policies put in place by the Security Team.  The policies would list how much and what data is encrypted (All, Specific Environments, Specific Tables, a Few Fields, or No encryption).  ETL developers would code for the policies but will not define the policies. If eencryption is elected to be enforced it can be handled in Informatica using the Source Qualifier, Lookup, and Expression Transformations. In both the Source Qualifier and lookup transformations the developer would use a SQL override to encrypt the data fields. The expression transformation can encrypt the data by editing the ports. Use the field that the developer wants to encrypt as the input port then override the value using the encryption rule as the output port. The developer will need to make sure that the database or staging table can handle the length of the encryption. This can be seen if the values sourced do not match the values returned after encryption/decryption. If this occurs the developer should check the field length of every port of the fields encrypted to see if it can handle the full length of encryption. Encrypting the data normally extends the length of the value and if the port or table is not ready the data will be truncated. Then when it is eventually decrypted the data will not match.

## Metadata

### **Informatica Technical Metadata**

Technical metadata is used by the IT support organization to ensure that the data is valid, timely, and accurately reflects what is being pulled from source systems. This metadata is also used for the following purposes: change control, ease of impact analysis and development effort for future modifications, and enhancement of data warehousing architecture. The following is a brief list of typical technical metadata:

* + Data warehouse field lengths and definitions
  + Field-to-field mappings between source and target
  + Query response times
  + Usage of queries and aggregation tables
  + Timings of loads, updates, archives into and out of the data warehouse
  + Timings and verifications of success for batch file transfers

Informatica PowerCenter captures various forms of metadata in its data repository which is a queryable, though somewhat cryptic, database. Examples of such metadata are table definitions, column mappings, business rules, data definitions, and execution statistics. Repository originates in two forms:

**Static**

* Source and Target Tables
* Workflow/Mapping Configuration
* Last Saved and other edit audit fields.

**Dynamic**

* Which jobs ran last night
* Were they successful
* What tables were affected
* How many rows were read & written
* How long did it take to run

Only use outside functions if it will provide a significant increase in throughput or decrease in processing time. Otherwise, metadata benefits decrease as no metadata will be stored in the repository. In short Informatica Metadata is strictly technical metadata.

### **Business Metadata**

Business metadata describes information available through a data warehouse in business terms. Business metadata starts with informative definitions of the data available to users, including business descriptions of the sources and of calculations or transformations that may be applied in the process of moving the data from the sources. It includes search capabilities that allow users to request a list of all data items with similar names, which ensures that users select the correct data item for their query. It includes context information to allow users to understand the context within which each data item was created. Business metadata also includes data on the timeliness of data – that is, exactly when the latest update occurred.  The following is a brief list of typical business metadata:

* Business Rules describing what is and is not included within the date warehouse
* Definitions of Business Hierarchies and KPIs
* Common Business Definitions and Calculations for data elements
* Transformation and Conversion Rules in Business context
* Source System Names/Locations
* User security profile
* Descriptions of warehouse attributes
* A description of warehouse data transformations over time