#### Sprint 1 **–** OP**’**s HANDOVER DOCUMENT

Last saved: 11/25/2014 8:14 PM



Table of Contents

Document Control 3

4.1 Business Overview 4

4.2 Technical Overview 4

5.1 Software Requisition 6

5.2 System Architecture 6

5.3 Modular Components 7

6.1 EXTRACTION Routines 7

6.1.1 Description 7

6.1.3 RUNTIME PARAMETERS 8

6.1.4 ENVIRONMENT CONFIGURATION VARIABLES 8

6.1.5 BEHAVIOR AND RUNTIME SCENARIOS 8

6.1.6 LOGGING 8

6.2 FILE WATCHER 8

6.2.1 Description 8

6.2.2 Command Line 9

6.2.3 RUNTIME PARAMETERS 10

6.2.4 ENVIRONMENT CONFIGURATION VARIABLES 10

(~/configs/ filewatcher.cfg) 10

6.2.5 BEHAVIOR AND RUNTIME SCENARIOS 11

6.2.6 LOGGING 11

6.3 LOAD MAPS 12

6.3.1 Description 12

6.3.3 RUNTIME PARAMETERS 12

6.3.4 ENVIRONMENT CONFIGURATION VARIABLES 13

6.3.5 BEHAVIOR AND RUNTIME SCENARIOS 13

6.3.6 LOGGING 14

6.4 WFM (Workflow Management) 15

# Document Control

|  |  |  |
| --- | --- | --- |
| **WHO** | **WHEN** | **WHAT** |
| Sreekanth Rama | 09/25/2014 | Initial Version |
| Rodger Karl | 11/25/2014 | Updated last release components. |
|  |  |  |

# Executive Overview

The main objective of this document is to demonstrate how to run the jobs on the server using the Infosense the DI/ETL Postgres meta models and unix command line utilities.

# Scope

The scope of this document is showing “how to run the jobs” and “setup etl metadata”.

# References

This table provides key contacts regarding any data extract/load issues that may arise.

|  |  |  |
| --- | --- | --- |
| Issue | Reference Organization | Contact Name |
|  |  |  |
|  |  |  |
|  |  |  |

# Application Description

The Application Description Section gives a detailed description of the application. It is organized into business and technical descriptions:

## Business Overview

The main objective of this project is to load a series of dbms extracts and external files into a target Postgres database with an integrated set of dimensions and fact groups tailored to Iora Health Care Services, Business Fulfillment (aka CLAIMS) and applicable business intelligence and health care analytics (aka Aggregates). This document is technical in nature and doesn’t cover the rationale for consumption of the data. This document is the ‘how’ and ‘what’ of managing data production from staging, integration, and aggregation tiers of the data warehouse.

## Technical Overview

In this section, we show you how the data flows from file to table.

Two major components have been assembled for Iora to process data into staging and integrate into the CORE or AGGREGATE data architecture. See System Architecture for depiction of these data tiers and the processing component modules for details on ‘how’ and ‘what’. Please note, only specific components were built given the time and materials of this engagement.

Upon completion of all ‘staging’, scheduled or event driven workflow management may be invoked. Please see ‘workflow management’ section for details on common and control of this module.

# System Architecture

## Software Requisition

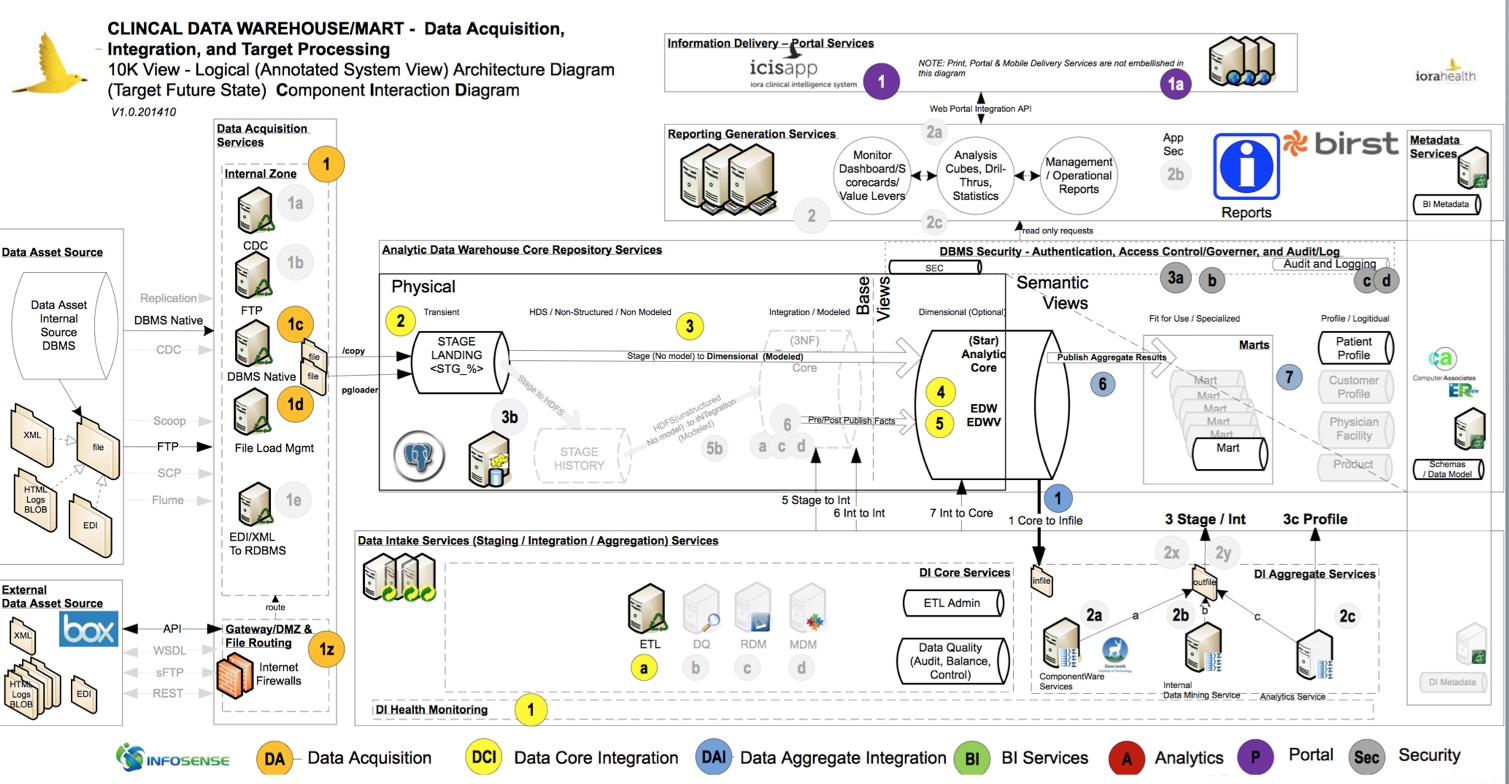
* Terminal Services (X.10) (e.g. PUTTY/SSH terminal)
* FTP CLIENT
* POSTGRESQL (Lisp Compiler and pgloader)
* Bash Command Shell
* Postgres 9.2.4 or higher Console / Admin
* Navicat 11.1.4 for Postgres (Optional)
* Clozure Common Lisp Version 1.9-r15757
* Postgres Add-Ons

CREATE EXTENSION pg\_trgm

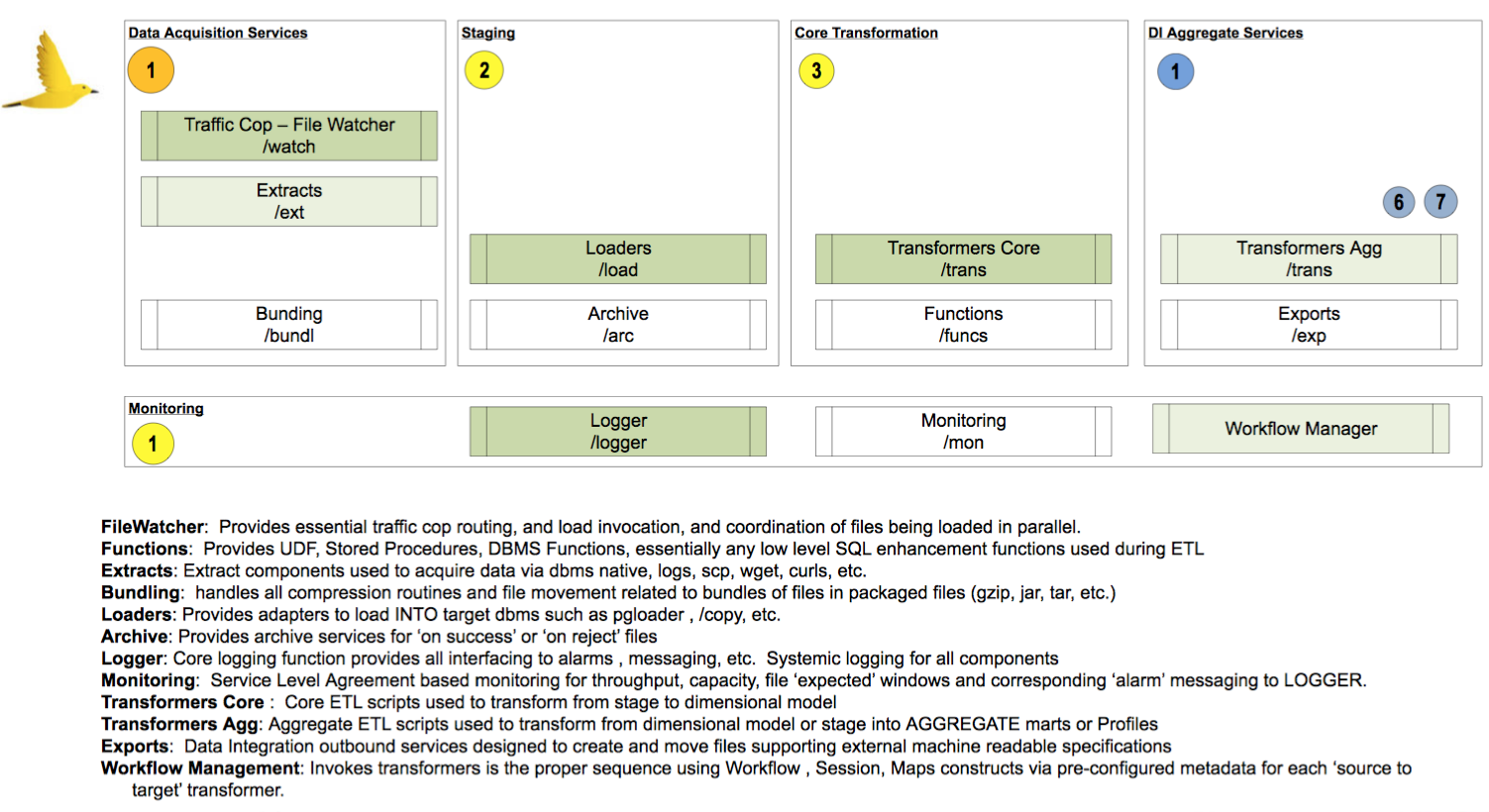
CREATE EXTENSION fuzzystrmatch

When Applicable: PostGIS - install and add ALL STATES (inito tiger schema) for (WHEN APPLYING HOUSEHOLDING and ADDRESS enrichment)
schemas are:
postgis,
postgis\_topology,
postgis\_tiger\_geocoder

## System Architecture



## Modular Components



# Component Overview

*Note: All references in this demo points to DEVELOPMENT users or databases or locations. It is only a matter of changing the environment variables across other environments.*

Logging into IORA machines

* SSH to IORA , speak to the sys administrator to get setup with ssh tunneling

Use the following command to login



## EXTRACTION Routines

## Description

The extraction routines were not built fully as time was not permitted and the ‘need’ was not identified as only one internal DBMS exists and all other input files were/are manually handled by the data management team. Instead, a routine script was built to ‘extract’ from the only internal Postgres dbms needed. This script is stored under ~/scripts/ext

* + 1. Command Line

**ALIAS ‘ex** is setup to move to the extract directory.

Runtime Command

[dw\_dev@analytics.production.icisapp.com:~/scripts/ext]$ 1run.ext.icis

## RUNTIME PARAMETERS

| Flag | Description | Example |
| --- | --- | --- |
| -f | Reference to the file that lists by line return each line as:  \* host, dbms, and tables to extract , separated by : | icis.table.snow\_icis\_only |

## ENVIRONMENT CONFIGURATION VARIABLES

## BEHAVIOR AND RUNTIME SCENARIOS

Just runs the extract using a Postgres /copy into a file. If filewatcher is running, it gets picked up immediately and staged into target dbms.

## LOGGING

APP\_ID = ‘EX’

Common Logging (Begin and End Only)

20141116\_loadit.log: 20141116 15:54:54 EX 00000000000000 000 EXBEG BEGIN EXTRACT FOR : 10.0.223.20 snowflake\_production dw\_dev users 0000001~

20141116\_loadit.log: 20141116 15:54:54 EX 00000000000000 000 EXEND END EXTRACT FOR : 10.0.223.20 snowflake\_production dw\_dev users 0000001~

Common Errors

None trapped.

## FILE WATCHER

## Description

The purpose of the the ‘filewatcher’ module is to load all inbound ‘files’ into the staging area of the Data Warehouse infrastructure. Filewatcher is geared to perform the loads in parallel upon discovery within the ~/data/inbound directory. These extract must be greater then X minutes / days before the watcher will pick them up to avoid ‘in process’ extracts (e.g. notably ftp processes).

Upon discovery of a legitimate file configured in ETL\_FILE\_CTRL, a JOBID is generated and the processing components are instantiated within a new ~/data/working/<JOBID> directory. The <JOBID> is generated using a 14 digit format FW<YYMMDDHHMISS>, then for each pattern match found, it will use the numeric part of jobid and append a 2 digit counter in the end (1 to 99). All files to be loaded nohupped in ordered to execute in parallel. Please refer to section 6.4.1 in this document to see how “load\_pg” works.

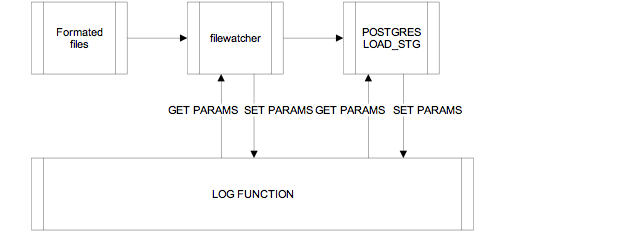
**Use Case**

Files are extracted into different formats (delimited, or fixed or Archived) files. These files land in <INBOUND> directory. These files are picked up by filewatcher process and are loaded into the database.

The filewatcher looks for files every 30 seconds, once, it identifies the files, it will loop through the files and will load into the database. In the process, a set of configuration files are automatically built with combination of functions and Unix scripts.

*NOTE: At each step, the LOG FUNCTION will be invoked multiple times to monitor the status of each step. In case of an error/failure (log level >= 3) , an email notification will be sent out.*

The following diagram displays the data flow between files and staging loads.

 Figure 1: Data Flow between files and Postgrisql database loads

## Command Line

**ALIAS ‘fw’** is setup to move to the filewatcher directory.

Runtime Command

[dw\_dev@analytics.production.icisapp.com:~/scripts/watch]$ filewatcher

Background Command[dw\_dev@analytics.production.icisapp.com:~/configs]$ fw

[dw\_dev@analytics.production.icisapp.com:~/scripts/watch]$ nohup filewatcher &

## RUNTIME PARAMETERS

| Flag | Description | Example |
| --- | --- | --- |
| -k | Kill Flag | -k Y |

## ENVIRONMENT CONFIGURATION VARIABLES

## (~/configs/ filewatcher.cfg)

| Configuration ATTRIBUTE NAME | DESCRIPTION | EXAMPLES |
| --- | --- | --- |
| FILESEARCH\_REGEXP | General file pattern to search | "\*.tstme|\*.tst|\*.txt|\*.zip|\*.csv" |
| RUN\_MODE | Variable to set filewatcher continuous or not (one loop only) | CONTINOUS |
| INTERVAL | Seconds delay between each filewatcher run, applicable for continuous mode | 30 |
| TIME\_UNIT | At what unit of time should we watch files. | MIN |
| MIN\_OLD | How many minutes to wait before picking up a file. When setting TIME\_UNIT is at MIN level. Minutes old | 1 |
| DAY\_OLD | How many DAYS to wait before picking up a file. When setting TIME\_UNIT is at DAY level. DAYS old | 30 |
| WORKSPACE | place to put temporary files | /tmp |
| SRCDIR | Source default directory to look for files (NOT USED) | ${ENVHOME}/data/inbound |
| DESTDIR | Place where to move the files to (NOT USED) | ${ENVHOME}/data/working |
| JOBSDIR | Place where to put job specific logs | ${ENVHOME}/jobs |
| ABORT\_LOAD\_ON\_ERROR | Weather or not to abort the loads on error (NOT USED) | N |
| DEBUGLVL | Debug setting at 0,1,2 , 0 is LOG\_LVL   * 0 only to logging. * 1 is ALL LEVELS sent to LOGGER * 2 is ALL LEVELS to logger plus full STDOUT LOGGING | 1 |
| WATCH\_MAX\_JOBS | Maximum number of load jobs that can be invoked in parallel before a queue begins | 15 |

## BEHAVIOR AND RUNTIME SCENARIOS

Filewatcher runs ‘silent’ when DEBUGLVL is set to 0 or 1. Filewatcher should be run in the ‘background’ and always be running. It is recommended the filewatcher be setup on a job schedule to ‘test’ heartbeat ongoing depending on your companies desired velocity, a runtime ‘check’ should be scheduled every “X” Units of time.

## LOGGING

APP\_ID = ‘FW’

Standard Loggings (0,1)

20141119\_loadit.log: 20141119 20:55:46 FW FW141119205546 000 000 START 0000045File Watcher started..... job FW141119205546~

20141119\_loadit.log: 20141119 20:57:17 FW FW141119205546 000 001 WHILE LOOP 4 0000001~

20141119\_loadit.log: 20141119 20:57:17 FW FW141119205546 000 001 FLIST 0000086Full File Processing >> /home/dw\_dev/data/inbound/hum/IORASG\_PE20141101\_RECLMDSP.TXT ~

20141119\_loadit.log: 20141119 20:57:17 FW FW141119205546 000 001 FILED 0000080File Directory Parsed >> hum >> File Parsed >> IORASG\_PE20141101\_RECLMDSP.TXT ~

20141119\_loadit.log: 20141119 20:57:17 FW FW141119205546 027 000 FOUND 0000063Matched File to Workflow> >> WF:2:SESSION:8:MAP:txt.reclmdsp ~

20141119\_loadit.log: 20141119 20:57:17 FW 14111920571711 027 001 JOBID 0000041JOBID Created for File >> 14111920571711~

20141119\_loadit.log: 20141119 20:57:17 FW 14111920571711 027 001 JBDIR 0000067Job Directory created >> /home/dw\_dev/data/working/14111920571711 ~

20141119\_loadit.log: 20141119 20:57:17 FW 14111920571711 027 001 JBMOV 0000101Moved File to Job Folder >> /home/dw\_dev/data/working/14111920571711/IORASG\_PE20141101\_RECLMDSP.TXT ~

20141119\_loadit.log: 20141119 20:57:17 FW 14111920571711 027 001 STRIP 0000108Stripping File for s/\x0//g >> /home/dw\_dev/data/working/14111920571711/IORASG\_PE20141101\_RECLMDSP.TXT ~

20141119\_loadit.log: 20141119 20:57:17 FW 14111920571711 027 001 JMETA 0000110Moved Metadata to Job Folder >> /home/dw\_dev/data/working/14111920571711/IORASG\_PE20141101\_RECLMDSP.TXT.meta ~

20141119\_loadit.log: 20141119 20:57:17 FW FW141119205546 027 001 LDPGL 0000119Invoke pgloader >> /home/dw\_dev/scripts/load/load\_pg -j 14111920571711 -f IORASG\_PE20141101\_RECLMDSP.TXT -d hum ~

Common Errors:

FILEWATCHER is already running. Or you have a session open with VIM/VI.

20141130.log: 20141130 22:01:48 FW 00000000000000 000 003 ALRDY 0000050 **ERROR: File Watcher process is already running. ~**

NO METADATA FOUND or IMPROPERLY CONFIGURED IN ETL schema, FILE IS THROWN TO ARCHIVE.

## LOAD MAPS

## Description

Loading falls into three categories for Iora. These are:

* LOAD\_PG: PGLOADER bound (open source tool built on Lisp)
* LOAD\_CPY: COPY inbound using Filewatcher based invocation
* LOAD\_MAP: PSQL inbound using Workflow Management invocation and etl\_ctrl.
  + 1. Command Line

**ALIAS ‘ld’** is setup to move to the load directory.

Runtime Commands

LOAD\_CPY

**load\_cpy** -h <this help> optional

DI Load Script for postgres COPY

-> -f <Filenameorpattern> semi-required

-> -d <File source drop directory> Semi optional

-> -c <INPUT CONFIGURATION FILE> Override if applicable for CONF located under /home/dw\_dev/dev/scripts/conf

-> -j <Job id> optional 14 digit numeric value in the format YYYYMMDDHHMISS (If not set, Jobid is created.

LOAD\_PG

**load\_pg** -h <this help> optional

DI Load Script for pgloader

-> -f <Filenameorpattern> semi-required

-> -d <File source drop directory> Semi optional

-> -c <INPUT CONFIGURATION FILE> Override if applicable for CONF located under /home/dw\_dev/dev/scripts/conf

-> -j <Job id> optional 14 digit numeric value in the format YYYYMMDDHHMISS (If not set, Jobid is created.

LOAD\_MAP

**load\_map** -h <this help> optional

DI Load Script for map specific (ETL\_CTRL\_KEY)

-> -w <workflow id> required

-> -s <session\_name> required

-> -m <map name> required

-> -j <Job id> optional 14 digit numeric value in the format YYYYMMDDHHMISS (If not set, Jobid is created.

## RUNTIME PARAMETERS

|  | Flag | Description | Example |
| --- | --- | --- | --- |
| LOAD\_CPY  LOAD\_PG only | -f | The filename to be loaded | Fileclaims.txt |
| LOAD\_CPY  LOAD\_PG only | -d | The directory under < ETL\_INBOUND\_DIR> e.g., ~/data/inbound to located the file | drw |
| LOAD\_CPY  LOAD\_PG only | -c | Override the automated generated of the configuration file and use this one  (NOT IMPLEMENTED) | Sample.conf |
| All | -j | JOBID overrides the generation and will refer to a specific directory under ~/data/working | 1234 |
| LOAD\_MAP  only | -w | Workflow Unit , see ETL\_CTRL.ETL\_WF\_ID | 10 |
| LOAD\_MAP  only | -s | Session Name , see ETL\_CTRL.ETL\_SESS\_NM | 1 |
| LOAD\_MAP  only | -m | Map Name, see ETL\_CTRL.ETL\_MAP\_NM | d\_age\_gend\_bnd |

## ENVIRONMENT CONFIGURATION VARIABLES

| Configuration ATTRIBUTE NAME | DESCRIPTION | EXAMPLES |
| --- | --- | --- |
| DEBUGLVL | Debug setting at 0,1,2 , 0 is LOG\_LVL   * 0 only to logging. * 1 is ALL LEVELS sent to LOGGER * 2 is ALL LEVELS to logger plus full STDOUT LOGGING | 1 |
| LPATH | Path to source LOAD scripts (NOT USED) | ${ENVHOME}/scripts/load |
| LOADLOG | Internal logging (NOT USED) | LOAD.log |

## BEHAVIOR AND RUNTIME SCENARIOS

Three scenarios exist.

1. Manual loading , testing a new file.
   1. Staging Loads: Call this ‘prototyping’ the configuration using the ETL\_CTRL, ETL\_FILE\_CTRL.
   2. Integration or Aggs Loads: Call this ‘prototyping’ the configuration using the ETL\_CTRL, ETL\_WF, ETL\_WF\_MSTR\_COMP
2. Filewatcher automated loads. LOAD\_CPY and LOAD\_PG are invoked by FILEWATCHER. No manual invocations.
3. Workflow management: LOAD\_MAPS are invoked by wfm. No manual invocations.

## LOGGING

APP\_ID = ‘LD’

Common Logging (Begin and End Only)

Load Logging is extensive, please refer to detailed logs online within the ~/logs/current.log or historical logs (<yyyymmdd\_loadit.log)

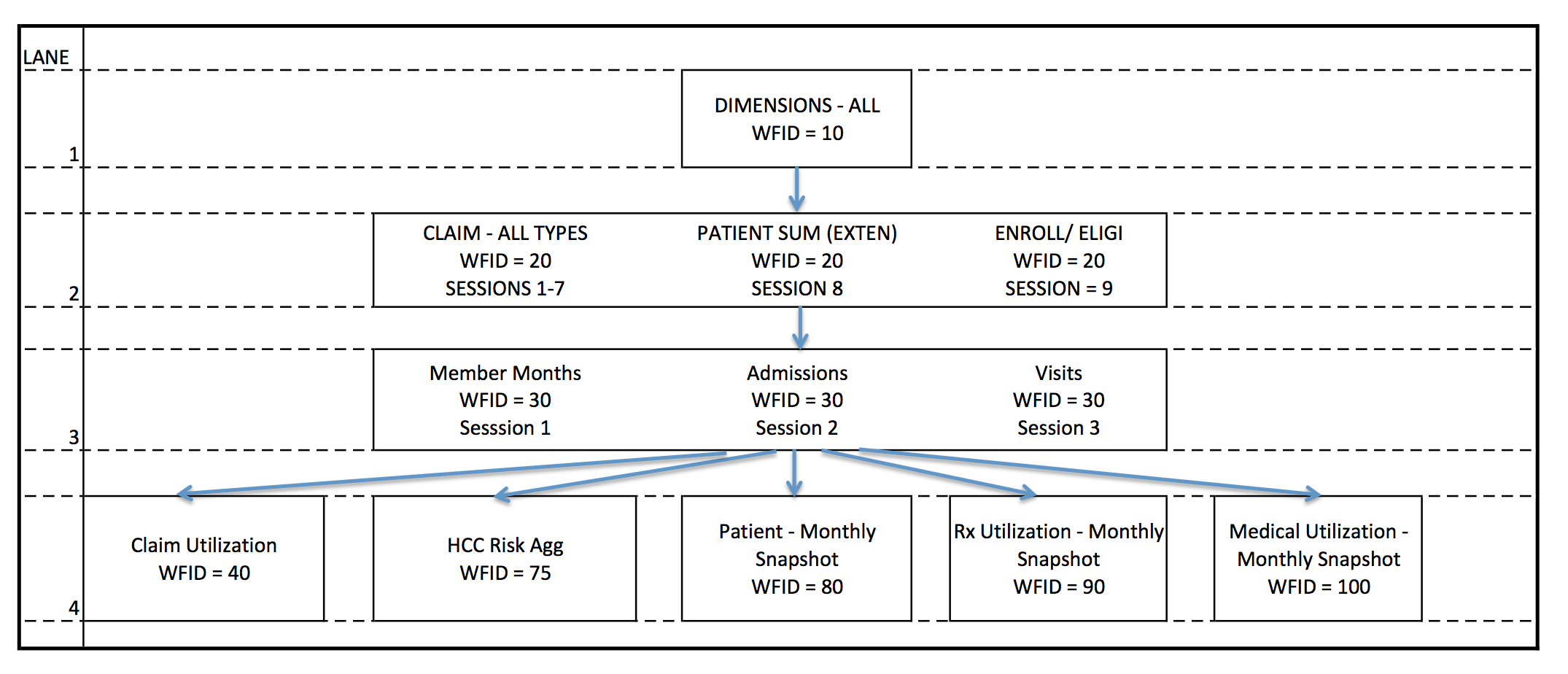
Common Errors

* METADATA CONFIGURATION ISSUES with ETL schema.
* FILE is parsing incorrectly or has bad ‘data’ control characters
* DBMS target structures not in place.

## WFM (Workflow Management)

Workflow management falls into several tiers comprising units of work to run either in parallel or serially. These components may be either run individually or run in ‘block’ units. A pre-defined master workflow has been assembled for Iora to handle the existing requirements as of 2014-12-01.

The following diagram outlines the master workflow unit , workflow components, and sessions.



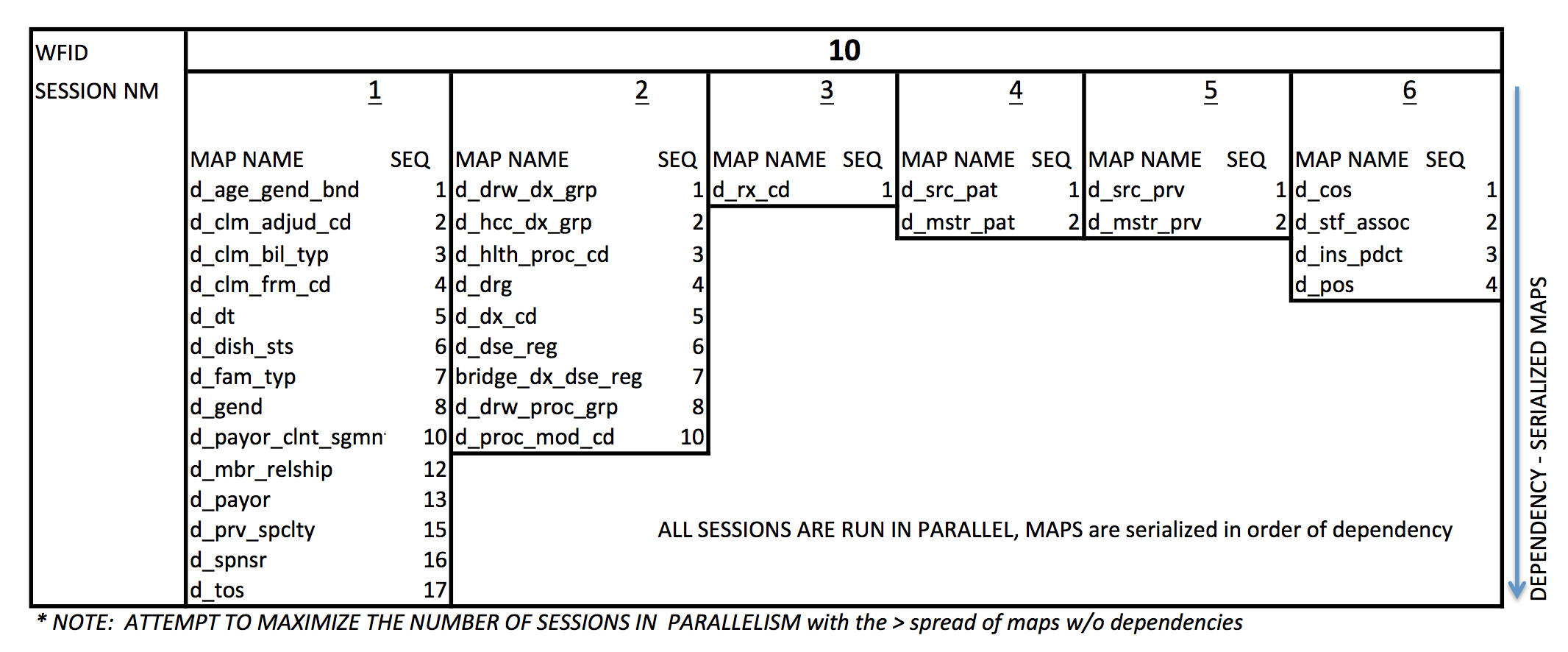
**A workflow master** is a special kind of workflow which comprises one to many workflow units. Workflow master is used to manage multiple workflows and should be considered the main runtime ‘utility’ for operations. The configuration of the workflow master is setup in the etl.etl\_wf\_mstr\_comp.

**A workflow** unit comprises one to many workflow sessions. Workflow units are used to manage multiple sessions. Each session is run in parallel.

**A session** unit comprises one to many source to target mappings. Each Session is used to manage a dependency chain of mappings. All mappings are run in sequential serialized form according to session order sequence identity in the etl control configuration.

**A map** comprises the source to target script to transform data to the desired target object. The map is the heart of the workflow management operations. See LOAD modules for reference on operation of specific logistics.

Below is a depiction of workflow unit 10, comprised of



### Command Line

**ALIAS ‘wf’** is setup to move to the workflow app directory.

->WORKFLOW MANAGEMENT mechanism

->./wfm -h? <this help> optional

-> -W < Master Workflow - multiple workflows , timed by dependency , serialized by lanes. See completion dependencies in etl\_wf\_mstr\_comp

-> -w < workflow level multiple stream(s) of sessions - see (ETL.etl\_wf > etl.etl\_ctrl.etl\_wf\_id

-> -s < Session level stream of maps - see (etl.etl\_ctrl.etl\_sess\_nm , must also include -w

-> -m < Mapping level singleton map - see (etl.etl\_crl.etl\_map\_nm - OPTIONAL

\*\*\*\* You must pass either a WORKFLOW, WORKFLOW + SESSION combo , or a specific ETL MAP NAME \*\*\*\*\*\*

### RUNTIME PARAMETERS

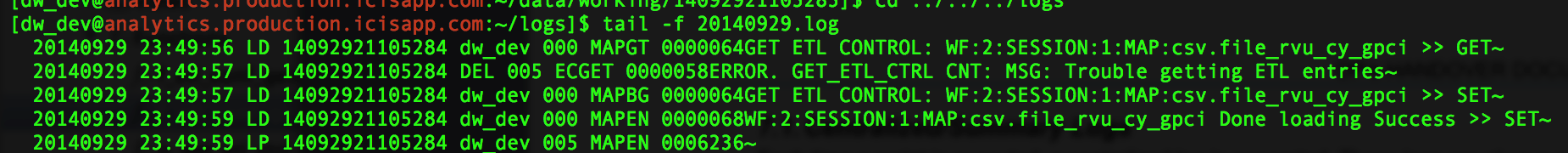
| Flag | Description | Example |
| --- | --- | --- |
| -W | A master workflow (if used, this is ‘forked’ and monitoring may be done using LOGS. No other parm is required when invoking a MWF | 101 |
| -w | Workflow Unit , see ETL\_CTRL.ETL\_WF\_ID  No other parm is required when invoking a WF | 10 |
| -s | Session Name , see ETL\_CTRL.ETL\_SESS\_NM  A –w is required when invoking a Session | 1 |
| -m | Map Name, see ETL\_CTRL.ETL\_MAP\_NM  A –w and –s is required when invoking a Map | d\_age\_gend\_bnd |

# ETL Metadata and Logging

The status and results of the scripts that are running can be monitored at different levels using different types of logs. The log types are

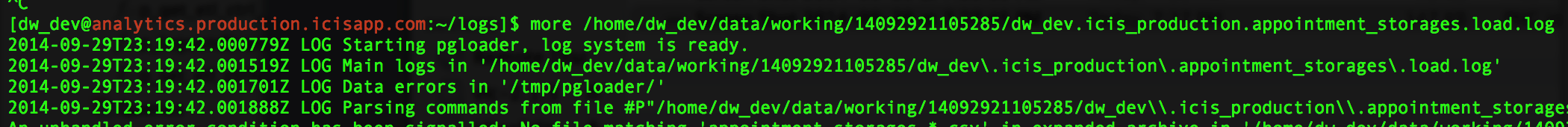
### Centralized/Summary Logs

Each time a script is executed, a centralized log is generated. These logs are always pushed into a file (except in the event of failure). This file is located in the directory *${ENVHOME}/logs* and the filename is *<CURRENT\_DATE>.log (Ex: 20120101.log). A*t the end of each day this file is loaded into Teradata into the table <ENV>ETL.LOGTBL.

sample log looks like

### Detailed/Standard Logs

These are the logs generated by the scripts at the run time and are located in the directory *${ENVHOME}/data/working/<JOBID>/* . These logs are used by log miner scripts to extract the statistics like row counts, processing times, failure messages and / or failure codes, return codes, emails etc,.

Sample log looks like:

### ETL\_CTRL and Logging

The ETL\_CTRL stores all the MAPS. MAPS can be either filewatcher bound or workflow management bound (i.e., ETL\_OWNER\_NM = ‘wfm’ or ‘filewatcher’ ). Each maps is a runtime source to target script that is either stored under ~/scripts/trans or is dynamically generated via staging metadata data within the ETL schema.

All transform scripts are currently stored as template scripts and are referenced within the ETL\_CTRL.ETL\_SCRPT\_PARM\_TXT (e.g., ‘edw.d\_spnsr.psql’ ) and must be physically stored under ~/scripts/trans

To setup a new ‘map’ , you must insert a unique ETL\_CTRL record using the following sample SETUP Insert DML below. If you setting up a filewatcher control, you must also setup the ETL\_FILE\_CTRL metadata. Lastly, bear in mind you must also have the DDLs (tables) deployed and accessible for the loads to successfully execute.

|  |  |  |
| --- | --- | --- |
| SCHEMA.TABLE | ETL.ETL\_CTRL |  |
| **COLUMN** | **EXAMPLE (LOAD\_MAP )** | **Description** |
| ETL\_CTRL\_KEY | 100101 | KEY: Unique Surrogate Key for the record. Artificial |
| ETL\_OWNER\_NM | ,‘wfm’ | APP OWNER ID: The application that owns this map. Right now, we have two applications: FW (filewatcher), and WFM (Worfklow Manageer) FW is for staging files , WFM is for scheduled staging to core or aggregate Transformations (in dbms processing) |
| ETL\_MAP\_NM | ,'f\_med\_agg' | -m MAP NAME – used to identify the logical map name, may be used in reference to command line executions. |
| ETL\_WF\_ID | ,100 | -w WORKFLOW ID – used to identify a logical workflow unit, may be used to identify command line executions |
| ETL\_SESS\_NM | '1' | -s SESSION NAME – used to identify a logical SESSOIN group , may be used to identify command line executions |
| ETL\_WF\_SESS\_ORDR\_SEQ | 1 | Sequence Order - the order in which the map must execute within a given session process |
| ETL\_SCRPT\_PARM\_TXT | ,'edw.f\_med\_agg.psql' | SCRIPT REF: When using WFM, The reference to the script located under ~/scripts/trans  When using FW, the reference to the file or configuration used to bind a staging file into the DBMS. |
| TGT\_TABLE\_NM | ,'f\_med\_agg' | TARGET TABLE NAME – the name of the physical table targeted by this map record. |
| -- REST TYPICALLY STATIC ON INIT |  |  |
| ETL\_ABOUT\_MAP\_NTS | ,’SQL’ | MAP TYPE – optionally describes the MIME type for this map control |
| TECH\_TYP\_CD | ,’NA’ | Not used at IORA |
| SRC\_SCHEM\_NM | ,’NA’ | When FW, the name of the directory under ~/data/inbound/ to search for files  When WFM, not used at IORA |
| TGT\_SCHEMA\_NM | ,’edw’ | TARGET SCHEMA – the target schema for this map |
| TGT\_CNCTION | ‘psql’ | CONNECTION STRING: Not used at IORA |
| ETL\_JOB\_STRT\_DT | ,'1900-01-01 00:00:00' | AUDIT MAP BEGIN DATE – defaulted on insert, set when running as begin date of process invocation |
| ETL\_JOB\_END\_DT | ,'9999-12-31 00:00:00' | AUDIT MAP END DATE – defaulted on insert, set when running as end date of process last updated |
| SRC\_ROW\_CNT | ,0 | TOTAL RECORDS or ROWS found on source, set to 0 default on insert |
| TGT\_ROW\_CNT | ,0 | TOTAL RECORDS UPDATED or ROWS found on source, set to 0 default on insert |
| TGT\_RJCT\_ROW\_CNT | ,0 | TOTAL RECORDS REJECTED or ROWS found on source, set to 0 default on insert |
| ETL\_STS | ,’INIT’ | MAP STATUS, set to ‘INIT’ initially.  **USE ‘SUSPEND’ if you want the processes to IGNORE THIS RECORD.** |
| ETL\_PARM\_BG\_DT | ,'1900-01-01 00:00:00' | REALTIME: DELTA PROCESSING BEGIN: Not used at IORA |
| ETL\_PARM\_END\_DT | ,'9999-12-31 00:00:00' | REALTIME: DELTA PROCESSING END: Not used at IORA |
| SRC\_SYS\_ID | 0 | AUDIT FOR ORIGINATING SOURCE SYSTEM. See ETL.DATA\_SRC.SRC\_SYS\_ID. 0 is the internal system ANALYTICS |
| SRC\_TBL\_NM | ‘NA’ | For FW: This is the name of the originating generic FILE name or TABLE name reference. This is NOT used to bind search or file controls. |
| CDW\_JOB\_CTRL\_ID | 9999 | AUDIT JOB CONTROL: Not used by IORA, set to 9999 |
| TGT\_ROW\_UPDT\_CNT | 0 | TOTAL RECORDS or ROWS updated on Target , set to 0 default on insert |
| USER\_NAME | ‘dw\_dev’ | SECURITY APP USER: Not used at IORA. |
| CDW\_INS\_DT | Now() | When this record was inserted |
| CDW\_UPD\_TS | Now() | When this record was last updated |
| CDW\_SRC\_SYS\_ID | 0 | Where this data originates, Set to 0 (INTERNAL EDW data management) |
|  |  |  |

### WORKFLOW SCHEMA

Describes the purpose and intent of the workflow schemas

|  |  |  |
| --- | --- | --- |
| **SCHEMA.TABLE** | ETL.ETL\_WF |  |
| **COLUMN** | **DESC** | **EXAMPLE** |
| ETL\_WF\_ID | The workflow master identifier | 10 |
| ETL\_WF\_NME | A short label for the wokflow unit. Contains descriptors for master workflows as well | Im a short name |

|  |  |  |
| --- | --- | --- |
| **SCHEMA.TABLE** | ETL.ETL\_WF\_MSTR\_COMP |  |
| **COLUMN** | **DESC** | **EXAMPLE** |
| ETL\_WF\_ID | The workflow master identifier | 101 |
| ETL\_WF\_LANE\_ID | The lane identifier (think swim lanes), All workflow must start with LANE 1 | 1 |
| ETL\_WF\_ID | A FK reference to the workflow units setup within etl\_ctrl | 10 |
| DEP\_WF\_ID | The dependency the workflow id has on predecessor workflows |  |
| ETL\_VELO | NOT USED AT IORA |  |
| ETL\_PARM\_TXT | NOT USED AT IORA |  |
| ETL\_WF\_TRCE\_ID | Security User , NOT USED at IORA currently | dw\_dev |

Please note: YOU MUST HAVE A WORKFLOW on each ETL\_CTRL for WFM/MAPS

|  |  |  |
| --- | --- | --- |
| **SCHEMA.TABLE** | ETL.ETL\_CTRL |  |
| **COLUMN** | **DESC** | **EXAMPLE** |
| ETL\_WF\_ID | The workflow UNIT identifier. FK to the ETL\_WF and ETL\_WF\_MSTR\_COMP | 10 |
|  |  |  |

### ETL FILE CONTROL

Used to configure filewatcher input files.

|  |  |  |
| --- | --- | --- |
| **SCHEMA.TABLE** | ETL.ETL\_FILE\_CTRL |  |
| **COLUMN** | **DESC** | **EXAMPLE** |
| FILE\_CTRL\_ID | A unique identifier for the file control metadata. Usually use the same as ETL\_CTRL, but it doesn’t have to be the same technically | 85 |
| FILE\_TYPE\_CD | The ‘mime’ of the file type. Either CSV or FXD is all that is supported in this release | CSV  FXD  ARC |
| FILE\_NM | The ‘search’ pattern to find the file . WILD CARDING is expected using UNIX style grep |  |
| FILE\_OWNERSHIP\_CD | Application that owns the control Filewatcher is the only control in this release | Filewatcher |
| FILE\_DIR\_NM | The directory under <ETL\_INBOUND\_DIR> that the files may be found. | snowflake\_production |
| FILE\_HOST\_NM | Host where the directory/file may be found. NOT USED at IORA in this release. If you were making a multi-hosted solution, this is applicable | Localhost |
| FILE\_INS\_DT | Date of Configuration | 2014-09-19 |
| SRC\_SYS\_ID | MUST match the ETL.SRC\_SYS\_ID. Also, must be setup in ETL.DATA\_SRC | 5 |
| SRC\_SCHEMA\_NM | MUST match the ETL\_CTRL.src\_schema\_Nm | Snowflake\_production |
| SRC\_TBL\_NM | MUST match the ETL.src\_tbl\_nm | Specialities |
| ROW\_CNT | NOT USED |  |

### NOTIFICATIONS

In the event of script failures, mail alerts are sent to a list of users specified in *${ENVHOME}/configs/email\_list.conf*

These users will be notified a brief message of the error code, error application and error message. A sample of this notification is shown below:

LOG SEVERITY: 5

WORKFLOW ID: 2

SESSION NAME: SS2

MAP NAME:CSV.BetCatCodes

APPLICATION ID: LB

JOB\_ID: 20120608143042

ORIGINATING SOURCE SYSTEM ID:

FUNCTION CODE: MAPEN

ERROR MESSAGE: LOAD FAIL. ERROR=8, WF:2,SESSION:SS2,MAP:BetCatCodes

# ADDITIONAL REFERENCES

ETL Metamodal needs to be configured with care. For easy of use, refer to existing etl\_ctrl entries and its dependancies.

### DATA\_SRC

This is a reference table for SRC\_SYS\_ID. DATA\_SRC table holds information about any new source that needs to be added. For easy of use, the 3 different types of file formats are added as 3 different sources (FXD, CSV and ARCHIVE). This SRC\_SYS\_ID will be used for our configurations.

Important fields:

SRC\_SYS\_ID: a Surrogate key, New id for new source

SRC\_SYS\_MNEM\_CD: a shortcut for the source

VNDR\_CD: A shortcut for any extension of the source, Default is NA

SRC\_PHY\_TYP\_CD: Type of source, File or Table etc,.

SRC\_SYS\_CNTCT\_ID: Point of Contact person/email of the source

### ETL\_WF

This is a reference table to ETL\_WF\_ID. If any new workflow is being added, then we need to add an entry here. This Workflow id is incremental and should be used for their respective configurations.

Important fields:

ETL\_WF\_ID: a Surrogate key, New id for new workflows

### ETL\_CTRL

ETL\_CTRL is heart of our metamodal. Listed below gives you some introduction of the fields that needs to be configured correctly. If it is an archive file load, then we should repeat the entries with different TGT\_TABLE\_NM but same SRC\_TABLE\_NM

|  |  |
| --- | --- |
| Filed | Description |
| ETL\_CTRL\_KEY | Incrementing integer value, Unique |
| ETL\_MAP\_NM | ETL\_MAP\_NM can be anything, but for a load that is sourcing from a zip file, this should be the matching file pattern |
| TECH\_TYP\_CD | This is where you setup the delimiter, If any. |
| TGT\_SCHEMA\_NM | Target schema name. ex: STG\_DRM |
| TGT\_TABLE\_NM | Target table name |
| \*\_CNT fields | Initially set to 0. |
| ETL\_STS | Initially set to “INIT”, then it changes while loading. |
| ETL\_PARM\_BG\_DT | Date to set to start the next load. |
| SRC\_SCHEM\_NM | Source schema name, must match the one that we mention in DATA\_SRC\_TBL |
| SRC\_TBL\_NM | Source table name, must match the one that we mention in DATA\_SRC\_TBL |
| ETL\_WF\_ID | The workflow id that we created in step 2 |
| ETL\_SCRPT\_PARM\_TXT | Name of the .conf file. |

### DATA\_SRC\_TBL

A reference table for Source table information. DATA\_SRC\_TBL is a reference to source table information.

Important fields:

SRC\_SYS\_ID: a Surrogate key, key that is created for the source

SRC\_SCHEM\_NM: Source schemaname from where the file/table is brought

SRC\_TBL\_NM: Source table name

### DATA\_SRC\_COL

DATA\_SRC\_COL is used to:

1. Mention source to target mapping
2. Build the conf file
3. Set the formats.

Care should be taken to

1. Not to add audit columns.
2. Setup with Correct ETL\_CTRL\_KEY
3. Setting up the tgt\_formats

### ETL\_FILE\_CTRL

Here, we mention the file pattern to look for and their directory.

FILE\_pATTERN could be like any of the following

1. File\_name.csv
2. File\*name.csv
3. File.name.csv
4. File\*
5. \*.txt

Important fields:

FILE\_CTRL\_ID: a Surrogate key

FILE\_NM: The file pattern that we need to look for

FILE\_DIR\_NM: The landing directory of the source file