

Baylor Information Services (BIS)

**Enterprise Data Warehouse**

ETL Standards & Guidelines

|  |  |
| --- | --- |
| Prepared by | <Author name> |
| Version | <version number> |
| Revision Date | Thursday, July 12, 2012 11:13 |

This document is a property of Baylor Health Care System and contains confidential information related to processes developed and managed by Baylor Information Services. Any circulation of this document outside BIS will require approvals from the project manager. Changes to this document will be made without any prior notice.

**Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Change description** | **Changed By** |
| Draft | 05/14/2012 | Created new | Sajjan Janardhanan |
| 1 | 06/29/2012 | Updates regarding control table | Sajjan Janardhanan |
| 1.1 | 07/12/2012 | Updated the visio diagram for the control table | Sajjan Janardhanan |

**Review History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Review Comments** | **Reviewed By** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Document Approvals**

|  |  |  |
| --- | --- | --- |
| **Approver Role** | **Approver Name** | **Approval Date (Electronic)** |
| Business Lead / Project Mgr |  |  |
| EDW Project Manager |  |  |
| Business SME/Tester |  |  |
| EDW Data Analyst/Tester |  |  |
| EDW Architect |  |  |
| EDW Developer Lead |  |  |

**Table of Contents**

[1 Purpose 3](#_Toc329818713)

[2 Target Audience 3](#_Toc329818714)

[3 Control Table Usage 3](#_Toc329818715)

[4 INFORMATICA ETL 6](#_Toc329818716)

[5 Database 6](#_Toc329818717)

[6 Scripting 7](#_Toc329818718)

[7 Share Point a.k.a eTeams 7](#_Toc329818719)

[8 Other Considerations 7](#_Toc329818720)

[8.1 Design Document 7](#_Toc329818721)

[8.2 Test Plan & Test Cases 8](#_Toc329818722)

[8.3 Trouble Shooting 8](#_Toc329818723)

[8.4 Email Alerts 8](#_Toc329818724)

[8.5 Transition to Support 8](#_Toc329818725)

[9 Glossary 9](#_Toc329818726)

[10 Appendix A – File Naming Standards 9](#_Toc329818727)

[11 Appendix B – Templates 9](#_Toc329818728)

**Table of Figures**

[Figure 1: New ETL Control Table 4](#_Toc329818729)

[Figure 2: UNIX scripts usage overview 5](#_Toc329818730)

# Purpose

This document provides the standards & guidelines to be followed by the ETL development team starting from the design phase until the implementation phase. This is an attempt to bring in consistency and standards within the ETL team under the BIS group. This document will be updated periodically in attempt to refine the standards over time.

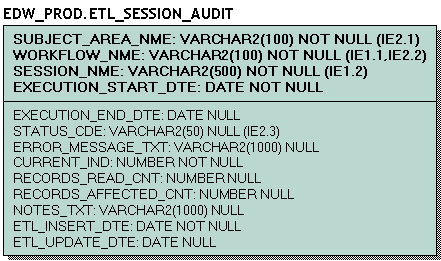
# Target Audience

This document is mainly intended for the members of the ETL team, but other members are welcome to go through the document to get a good understanding of the standards put in place by the ETL team.

# Control Table Usage

Many ETL programs maintained within the BIS stack make use of control tables that are local to each subject area, which serves specific needs. In such a situation, each control table would tend to have a different table structure and a different amount of detail. This can pose a challenge for consolidated reporting or dashboards and most certainly, for job monitoring and support. In an effort to standardize the control table usage and consumption, a single control table would be used by all existing and new processes within any existing or new subject areas. Legacy processes that make use of individual control tables cannot be converted to make use of the standardized control tables right away, but every effort will be made to make changes to use the new & standardized control table. The structure of the new standardized control table has been provided below. The control table design will be revisited at a later time to make it normalized and hold additional information related to SLA, table level load information etc.

Figure : New ETL Control Table



Entries will be made into this table via UNIX scripts, which will be generic and parameter driven. These scripts will reside in the folder [/dw001/app/edwetl/scripts/common]. Illustrated below is a Visio diagram that describes the usage of these scripts at a high level.

Figure : UNIX scripts usage overview



The 3 scripts “startCycleETL.sh”, “endCycleETL.sh” and “failCycleETL.sh” are wrapper scripts calling the core script “CycleETL.sh”. Each of the 3 wrapper scripts call the core script with different parameters depending on the action that needs to be taken. It is recommended to make use of the wrapper scripts, instead of the core script.

The script [startCycleETL.sh] makes a new entry within the control table where the status says “Running” and end date is NULL, if the prior entry for a process has been marked complete, else the script would end in failure. This is to ensure the support personnel take a second look to understand the reason for the absence of a completed entry in the prior run. Once it is has been confirmed that the process can be run without any impact on the data, the prior entry can be manually closed and the script [startCycleETL.sh] can be re-run.

The script [endCycleETL.sh] closes an existing open entry within the control table with a status of “Completed” and end date with SYSDATE. If an open entry does not already exist, this script would end in failure. In addition to closing the open entry, this script also copies ETL load information, which is sourced directly from the MX views within the INFA repository. As the MX views reside in a database different from the ETL control table, a DB link will be required within the EDW or the INFA database, depending on any restrictions laid out by DBAs.

The script [failCycleETL.sh] closes an existing open entry within the control table with a status of “Failed” and end date with SYSDATE. Every other logic within this script is similar to that of the script [endCycleETL.sh].

The script [gather\_etl\_stats.sh] is an optional step in any process. This is a generic script to send the load statistics of a particular workflow to the EDW ETL support group.

None of these scripts will be called from within an ETL workflow. These scripts will be defined as independent jobs within the same of different job streams in the Tivoli Workload Scheduler.

# INFORMATICA ETL

The preferred tool for ETL development within BIS is Informatica PowerCenter. Every effort will be made to create ETL solutions within Informatica PowerCenter, before considering PL/SQL or SQL Loader.

The following bullets outline the standards to be followed during the ETL development & implementation.

1. Ideally, every project would contain its own folder within the repository, in which case, the mapping name would contain the name of the table that gets loaded.
2. Every ETL mapping can have only one pipeline and load only a single target table, unless the logic makes use of “constraint based loading” or the pipelines perform supplemental tasks like truncating the target table before performing the load in the main pipeline.
3. Any reusable logic should be implemented using a mapplet, reusable transformation or a user defined function depending on the logic requirements.
4. All complex expression logic should have comments that outline the logic, so changes to the logic can be effortless.
5. Changes to expression logic will be maintained historically with comments.
6. The repositories will have the team-based feature turned on in the development environment at the minimum. It is desired to have this feature turned on in all the environments though.
7. No business values should be hardcoded within the ETL, if it makes sense to parameterize them.
8. Sessions & email tasks should be made reusable. There can be exceptions though.
9. Database connections, schema names, table names, file names, pre & post SQLs, pre & post commands should be parameterized within the sessions.
10. The option “fail parent if task fails” should be checked, Stop on errors should be set to “1”, log history should be set to 20 for all relevant tasks.
11. All jobs will be scheduled using Tivoli. All notification emails for successful or failure completion of the jobs will be sent via Tivoli. Every attempt will be made to send a single notification email regarding a status from every process.
12. All process dependencies will be set within Tivoli. Example: file watch or completion of another process etc.
13. The columns ETL\_CREATED\_DTE & ETL\_UPDATED\_DTE will be updated with the value of SESSSTARTTIME.
14. The logic within the ETL mapping for upserts should be “Data Driven”. The logic within the mapping should determine the action to be performed.
15. ETL development or code migration should not be performed using a generic userid.
16. Members of the development team are encouraged not to make use of the generic userid “etlsvc”, but their own login id.
17. All deployments will be performed using “Deployment Groups”. A static deployment group “BIS\_Common” has already been created for this purpose. The final goal would make use of dynamic deployment groups, labels, queries etc.
18. All code deployments should be performed by the administrator or any individual identified by the administrator.

# Database

# Scripting

Scripting can be performed in either UNIX shell or Perl. Every script will have sufficient information regarding the changes made to the script with the author and date of the change, stating its purpose. All scripts would have sufficient logging mechanism to create log files in a designated folder. Individual log files would be created for every execution of the script containing the date & time at which the log file was created within the name of the log file. The standard name for the log file is (SJ). Any exception around all critical tasks should be explicitly handled within the script. Any email sent out from within the script would follow the guidelines mentioned in the section “Email Alerts” in this document. All scripts would have the permission set to 754 or in some cases 755.

Every log file should contain a standard header as given below (SJ).

When scripting in UNIX, Korn shell scripting is preferred. The script files should have an extension of “ksh” or “sh”. Any reusable logic or global variables will be made available in the form of a function within the file (SJ) located in the folder (SJ). This file will be referenced within the scripts to make use of these reusable functions and global variables.

# Share Point a.k.a eTeams

All requirements, functional specification, design document, source to target mapping spreadsheets will be made available in the eTeams after an official version is published. Further changes to any ETL related published documents should be made to the version that’s in the eTeams by doing a check-out & check-in. Following this approach will ensure all versions of the document are available to view by every member of the team. A sample of the folder structure has been provided below (SJ).

# Other Considerations

This section discusses miscellaneous items that have not been discussed in the document yet or do not qualify to have an independent section.

## Design Document

Every project that has ETL effort would require an ETL design document. This document should contain information around the entire process at a high level, so every member of the team can understand the big picture. The section that involves ETL would need to be highlighted to understand the role that ETL plays in the entire process. If a high level overview can’t be provided for any reason, the process overview can be around the ETL process alone. The preferred tool for any illustration within the document is Microsoft Visio, because this is a licensed application within BIS and can be embedded within a Word document, but at the same time can be edited for a future release, if needed. All Visio files would need to be archived in the BIS eTeams SharePoint site. The name of the original Visio file should be mentioned under the diagram to help a different member of the team to make changes to the illustration or reuse if needed. Multiple illustrations would be created in different tabs within a single Visio file per project. The naming convention of the ETL design document & Visio files have been discussed in Appendix A. The template for the ETL design document will be attached in Appendix B soon (SJ).

## Test Plan & Test Cases

Every new project that involves ETL effort will require a test plan for unit & integrated testing that outlines the effort and resources involved during the unit testing & QA phase. The ETL test plan & test cases could be mentioned in a single document for maintainability. The naming convention has been discussed in Appendix A. The template will be attached in Appendix B soon (SJ).

## Trouble Shooting

Trouble shooting in the production environment in the event of an issue is not a recommendation approach; therefore every program should have sufficient logging mechanism in place to identify the cause of a job failure in the production environment to be able to recreate the scenario in a non-production environment for testing the fix. If the code in question is a script, then the code can be designed to operate either in a BAU or a debug mode. In the debug mode, the script would not perform any critical tasks or can take a totally different flow in the logic of the program with the intent to troubleshoot.

## Email Alerts

During any development, unit testing or QA phase, email alerts are defined in many areas of the code to confirm the logical flow of the program in every step. However, once the code is migrated to production, several email alerts for a single ETL process to individuals who do not own the process may be misconstrued as multiple runs of the ETL job that ended successfully or in failure. Therefore only single alerts will be sent once the jobs start running in production. All email alerts will be controlled by the enterprise scheduling application.

The subject of the email alert would use the following format: [<ENVIRONMENT>] <STATUS>: <JOB NAME>

The “environment” indicates if the email was sent from the development, test or the production environment. The valid values are DEV, TST & PRD. The “status” indicates if the alert corresponds to a success or a failure notification. The valid values are SUCCESS & FAILURE. The “job name” indicates the name of the ETL workflow, UNIX script or a name given to the entire process that has a combination of scripts, ETL programs & stored procedures etc.

The body of the email alert should have information in terms of a reference code, which will help the production support team to locate the process from which the email notification was sent. If these email notifications are sent to teams who are not part of the BIS EDW group, these reference numbers can be used to get the context of the email notification. The reference numbers should be unique to every process, script and ETL and should be documented within eTeams. These reference codes should be controlled by the ETL team lead or the administrator. Information regarding specific programs or actual cause of failure should not be mentioned within the email notifications, but should be clearly mentioned within the log files only.

## Transition to Support

The ETL development team will prepare a Run-Book document, which will contain all the information about the process required for supporting the process. The production support team will use this document as reference to resolve any issue in production. The ETL development team would support the new process implemented in production for at least 2 weeks or a month, before the support ownership is given to the production support team. The naming convention has been discussed in Appendix A and the template will be attached in Appendix B (SJ).

# Glossary

# Appendix A – File Naming Standards

|  |  |
| --- | --- |
| **File Type** | **Naming Standard** |
| ETL Design Document | BIS\_ETL-Design\_<project name>\_v<version #>.docx |
| Visio Diagrams | BIS\_ETL-Diagrams\_<project name>.vsd |
| ETL Run Book | BIS\_ETL\_Runbook\_<project name>\_v<version #>.docx |
| ETL Unit Test Cases | BIS\_ETL\_UnitTestCase\_<project name>.xlsx |
|  |  |

# Appendix B – Templates