

|  |  |
| --- | --- |
|  | DARS |
| **Run Book**  **Product/Application/System:**  **Project:** DARS  **Date:** July 2010  **Version:** 0.1 |

Table of Contents

[1 Purpose 5](#_Toc285399222)

[2 Target Audience 5](#_Toc285399223)

[3 DARS Process Overview 6](#_Toc285399224)

[4 DARS ETL Process Detail 7](#_Toc285399225)

[4.1 AD sync 7](#_Toc285399226)

[4.1.1 Workflow wf\_AD\_DARS\_PreLoad 8](#_Toc285399227)

[4.1.2 Workflow wf\_AD\_DARS 9](#_Toc285399228)

[4.2 DOC Field Update 10](#_Toc285399229)

[4.2.1 Workflow wf\_AD\_DOC\_FIELD\_UPDATE 10](#_Toc285399230)

[4.3 Non-AD 11](#_Toc285399231)

[4.3.1 Workflow wf\_NON\_AD\_DARS 11](#_Toc285399232)

[4.4 Staging 13](#_Toc285399233)

[4.4.1 Workflow wf\_STAGING\_DARS\_CHASE 13](#_Toc285399234)

[4.4.2 Workflow wf\_STAGING\_DARS\_EMC 15](#_Toc285399235)

[4.4.3 Workflow wf\_STAGING\_DARS\_VLS 16](#_Toc285399236)

[4.4.4 Workflow wf\_STAGING\_DARS\_WAMU 17](#_Toc285399237)

[4.5 ORG employee sync 19](#_Toc285399238)

[4.5.1 Workflow wf\_ORG\_EMPLOYEE 19](#_Toc285399239)

[4.6 ENT tables sync 20](#_Toc285399240)

[4.6.1 Workflow wf\_EMPLOYEE\_ENT 21](#_Toc285399241)

[4.7 ORG employee login audit 21](#_Toc285399242)

[4.7.1 Workflow wf\_ORG\_EMPLOYEE\_LOGIN\_AUDIT 22](#_Toc285399243)

[4.8 PRC wrapper jobs 23](#_Toc285399244)

[4.8.1 Workflow wf\_PRC\_AD\_CLEAR\_TRIAL\_REQ\_DATE 23](#_Toc285399245)

[4.8.2 Workflow wf\_PRC\_AD\_PLAN\_BREAK\_UPDATE 24](#_Toc285399246)

[4.8.3 Workflow wf\_PRC\_GRD\_SHIPDATE\_UPDATE 24](#_Toc285399247)

[4.8.4 Workflow wf\_PRC\_OA\_DATA 25](#_Toc285399248)

[4.8.5 Workflow wf\_AD\_NIGHTLY\_BATCH\_UPDT 26](#_Toc285399249)

[4.8.6 Workflow wf\_PRC\_AD\_OA\_MODIFICATION\_TERMS 27](#_Toc285399250)

[5 Supplemental Process Detail 28](#_Toc285399251)

[5.1 Dynamic Parameter Files 28](#_Toc285399252)

[5.2 Virtual Partition Switching 29](#_Toc285399253)

[5.3 Document Processing 30](#_Toc285399254)

[5.4 UNIX scripts & paths 31](#_Toc285399255)

[6 Schedule Information 31](#_Toc285399256)

[7 Troubleshooting 32](#_Toc285399257)

[7.1 PDITD0002 32](#_Toc285399258)

[7.2 PDITD0003 32](#_Toc285399259)

[7.3 PDITD0004 32](#_Toc285399260)

[7.4 PDITD0005C 32](#_Toc285399261)

[7.5 PDITD0005E 32](#_Toc285399262)

[7.6 PDITD0005V 32](#_Toc285399263)

[7.7 PDIT0005W 33](#_Toc285399264)

[7.8 PDITD0007 33](#_Toc285399265)

[7.9 PDITD0007\_ENT 33](#_Toc285399266)

[7.10 PDITD0007 \_OELA 33](#_Toc285399267)

[7.11 PDITD0008 33](#_Toc285399268)

[7.12 PDITD0009 33](#_Toc285399269)

[7.13 PDITD0010 33](#_Toc285399270)

[7.14 PDITD0012 33](#_Toc285399271)

[7.15 PDITD0013 33](#_Toc285399272)

[8 Service Level Agreement 34](#_Toc285399273)

[9 High Level Technical Architecture & Contact Information 35](#_Toc285399274)

[10 Task Overview 35](#_Toc285399275)

[11 Pre-Installation Checklist 35](#_Toc285399276)

[12 Configuration & Setup 35](#_Toc285399277)

[12.1 Operating System 35](#_Toc285399278)

[12.2 Database Server 35](#_Toc285399279)

[12.3 Data Migration 35](#_Toc285399280)

[12.4 Web Server 35](#_Toc285399281)

[12.5 Application Server 35](#_Toc285399282)

[12.5.1 Application Server Setup 35](#_Toc285399283)

[12.5.2 Database Connection Setup 35](#_Toc285399284)

[12.5.3 Front-end / GUI setup 36](#_Toc285399285)

[13 Security Issues 36](#_Toc285399286)

[14 Other Packages 36](#_Toc285399287)

[15 Application Login IDs 36](#_Toc285399288)

[16 Content and Application Directory Configuration 36](#_Toc285399289)

[17 Production Support 37](#_Toc285399290)

[17.1 Alerts generated by system 37](#_Toc285399291)

[17.2 Informatica job monitoring / Visual confirmation 37](#_Toc285399292)

[17.3 General Tier 2 Escalation Procedures 37](#_Toc285399293)

[17.3.1 Escalation guidelines 37](#_Toc285399294)

[17.4 Security Administration Support 37](#_Toc285399295)

[17.5 Support Contact Information 37](#_Toc285399296)

[17.6 Business Contact 37](#_Toc285399297)

[18 Appendix 38](#_Toc285399298)

Document Control

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer Name** | **Role** |
| 1 |  |  |  |
|  |  |  |  |

Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Change description** | **Sections changed** | **Changed by** |
| 0.1 | 01/15/2011 | Initial draft | All | Sanjoy Banerjee |
| 0.2 | 01/29/2011 | Refinements | All | Sajjan Janardhanan |
| 0.3 | 02/13/2011 | Refinements | All | Sajjan Janardhanan |

# Purpose

This document provides overall details about the DARS process and acts as a reference document for the Tier 2 team (Production Support activities).

# Target Audience

The target audience for the run book are Tier 2 & Tier 3

Refer [Production Support](#_Production_Support) for contact details.

# DARS Process Overview

Initially all data required by the Agent Desktop (AD) application was housed in the CHAPSAPP schema in the BUSPROD database at Information One. Due to heavy usage of this database by various teams & individuals, the AD users were experiencing many outages, which led to need of a more reliable environment to house the AD application and its data. Therefore, the DARS environment was created to house mission critical applications, starting with AD.

Illustrated below is the process in DARS phase 1.



# DARS ETL Process Detail

The processes discussed in this section are the ETLs that belong to the DARS subject area. They are listed below for quick reference.

|  |  |
| --- | --- |
| Process Name | Control-M job name |
| AD sync | PDITD0002 |
| DOC field update | PDITD0003 |
| Non AD | PDITD0004 |
| Staging CHASE | PDITD0005C |
| Staging EMC | PDITD0005E |
| Staging VLS | PDITD0005V |
| Staging WAMU | PDITD0005W |
| Org Employee | PDITD0007 |
| ENT table sync | PDITD0007\_ENT |
| AD login audit | PDITD0007\_OELA |
| Clear trial request date | PDITD0008 |
| Plan break update | PDITD0009 |
| Guardian shipments update | PDITD0010 |
| Option Arm data | PDITD0011 |
| AD nightly batch update | PDITD0012 |
| Option Arm modification terms | PDITD0013 |

Every workflow connected to a process has a corresponding wrapper UNIX script, which is used by the Control-M scheduling suite to kick off the ETL workflow. Sometimes a process may contain 2 workflows, where one is called a pre-load and the other has no suffix. For example, wf\_AD\_DARS\_Preload & wf\_AD\_DARS. In this case, the Control-M job kicks off the wrapper UNIX script, which kicks off the former and the latter is kicked off from within the former.

## AD sync

Illustrated below is a high level overview of the AD sync processes.



Most of the ETL processes run on a daily schedule and some even run multiple times in a day. All the scheduling has been handled using the Control-M scheduling suite. Details of schedules of various processes in DARS have been provided in the scheduling section.

The loading strategy in all of the ETLs are either full CDC(change data capture) or delta insert or delta delete & insert. Given below is a presentation that contains various coding methods, out of which “CDC-DDF” & “3Di” are widely used. The list of mappings and their corresponding load strategy from the presentation below has been discussed in the spreadsheet below.

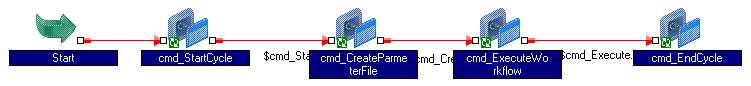


A screenshot of the ETL workflows have been provided below. The pre-load workflow is required for creation of the dynamic parameter file for the AD workflow. The parameter file is required to be dynamic in connection with the delta mechanism implemented in the “3Di” ETLs. The value of the parameter “$$ETL\_Delta” constantly changes after every run and is used during the next run for capturing only the delta records. The process & tables related to creating dynamic parameter files is a supplemental process, which is used by many workflows within DARS and also other subject areas and will be discussed in a different document.

### Workflow wf\_AD\_DARS\_PreLoad

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0002 |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_ad\_dars\_preload.ksh |
| **Schedule Information** | Daily 0800 EST to 2300 EST, every 3 hours |

Below is a screenshot of the ETL workflow.



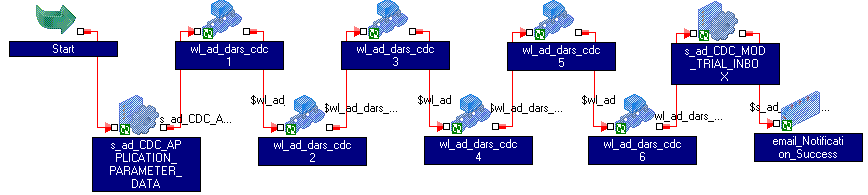
The table below details the function of each task in the table below.

|  |  |
| --- | --- |
| Task Name | Task function |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| cmd\_CreateParameterFile | Creates a dynamic parameter file for the CYCLE passed as parameter. This initiates a workflow wkf\_CmnCreateParameterFile, which creates the required parameter file in a pre-defined location. This process will be discussed in a separate document as mentioned earlier in this document. Also, this task appends a static parameter file “wf\_AD\_DARS\_global.parm” with the dynamically generated parameter file “wf\_AD\_DARS.parm”. |
| cmd\_ExecuteWorkflow | Runs the main workflow wf\_AD\_DARS |
| cmd\_EncCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

This workflow has “Suspend On Error” option checked, so the workflow gets into a “Suspended” state, when the master workflow “wf\_AD\_DARS” gets into a suspended state.

### Workflow wf\_AD\_DARS

Below is a screenshot of the ETL workflow.



This workflow loads 105 tables in the CHAPSAPP schema at BUSPROD from ADMGR @ DART01D. The sessions are contained in 6 worklets connected in sequence to limit the usage of the database server during the load process, but at the same time, improving efficiency by connecting the sessions in parallel within the worklet.

The tables corresponding to these sessions have FK reference constraints defined, which are disabled, before the load and enabled back after the load. Disabling the constraints is performed in the session “s\_ad\_CDC\_APPLICATION\_PARAMETER\_DATA” in the PreSQL property. Enabling the constraints is performed in the session “s\_ad\_CDC\_MOD\_TRIAL\_INBOX” in the PostSQL property.

To enable selective execution of tasks within the worklets in the workflow, worklet variables corresponding to each session have been created in each of the worklets and mentioned in the link to the sessions. The values to these worklet variables are set in the parameter file “wf\_AD\_DARS\_global.parm”, which gets appended to the dynamic parameter file “wf\_AD\_DARS.parm”. Setting the value of a worklet variable to zero would not run the session when the workflow runs. The default value given to these worklet variables is 1.

The file below lists the sessions contained in each of the above worklets; sorted by Session name.



## DOC Field Update

Illustrated below is the high level overview of the DOC field update process.

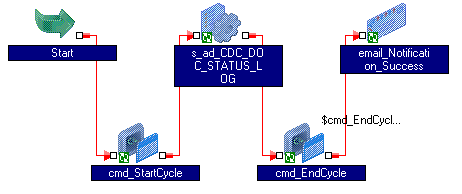


This process copies data in the table DOC\_STATUS\_LOG in the DITMGR schema at BUSPROD into a similar table in the ETLMGR schema at DART01D. This table in BUSPROD is loaded by the DOC generation process that deals with sending DOC related information to external vendors like Guardian & Lender Live. The data in this table is the result of processing a return file from Guardian, which is required for further processing in the DARS environment.

### Workflow wf\_AD\_DOC\_FIELD\_UPDATE

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0003 |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_ad\_doc\_field\_update.ksh |
| **Schedule Information** | Daily 0300 EST to 1100 EST, every 1 hour |

Below is a screenshot of the ETL workflow.



The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| s\_ad\_CDC\_DOC\_STATUS\_LOG | Loads the table DOC\_STATUS\_LOG in ETLMGR@DART01P, sourcing from DITMGR@BUSPROD and updates the flag in the source table to “Y”, so the same rows are not picked up again. |
| cmd\_EncCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

This workflow has “Suspend On Error” option checked. There is no dynamic parameter requirement in this process; therefore this process makes use of a static parameter file.

## Non-AD

Illustrated below is the high level overview of the non-AD load process.



This process copies data from various tables in CHAPSAPP & PORTMGR schema at BUSPROD into partitioned tables in ETLMGR schema at DART01P. These tables are read-only to the AD application and with the help of the partitioned table; data is made available constantly, even though data is being loaded in the background. This will be discussed in a separate section later in this document.

### Workflow wf\_NON\_AD\_DARS

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0004 |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_non\_ad\_dars.ksh |
| **Schedule Information** | Daily @ midnight EST |

Illustrated below is the manner in which the ETL workflow is laid out. Each of the sessions in this workflow perform a truncate and load of an inactive partition within the target table and finally performing a virtual partition switch. This has been discussed in detail later in this document.

The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| D\_Start | This decision task acts like an anchor to the links to the sessions in the workflow. |
| S\_stg\_HPIMSA | Loads table ETLMGR.HPIMSA @DART01P |
| S\_stg\_HPIST | Loads table ETLMGR.HPIST @DART01P |
| S\_stg\_HPO\_FORM | Loads table ETLMGR.HPO\_FORM @DART01P |
| S\_stg\_HPO\_FORM\_STATUS\_JOIN | Loads table ETLMGR.HPO\_FORM\_STATUS\_JOIN @DART01P |
| S\_stg\_MOD\_QUAL\_INVESTOR | Loads table ETLMGR.MOD\_QUAL\_INVESTOR @DART01P |
| S\_stg\_RMV\_BPO | Loads table ETLMGR.RMV\_BPO @DART01P |
| S\_stg\_RMV\_FORM | Loads table ETLMGR.RMV\_FORM @DART01P |
| S\_stg\_RMV\_INBOX\_CHANNEL\_JOIN | Loads table ETLMGR.RMV\_INBOX\_CHANNEL\_JOIN @DART01P |
| S\_stg\_TRCHK\_MOD\_FORM\_INBOX | Loads table ETLMGR.TRCHK\_MOD\_FORM\_INBOX @DART01P |
| S\_stg\_ZIP\_CNTY\_MSA\_ST | Loads table ETLMGR.ZIP\_CNTY\_MSA\_ST @DART01P |
| Wl\_nonad\_AVM\_VALUES | This worklet helps the workflow in reaching successful completion, even if all the sessions don’t run within the worklet. |
| * s\_stg\_AVM\_VALUES\_PreLoad | Checks if the target table in DART01P has a sequence # less than that of the max sequence # in BUSPROD and proceeds to load only, if the prior condition satisfies. |
| * s\_stg\_AVM\_VALUES | Loads table ETLMGR.AVM\_VALUES @DART01P for the highest sequence value |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

Below is a screenshot of the ETL workflow.



This workflow has “Suspend On Error” option checked. There is no dynamic parameter requirement in this process; therefore this process makes use of a static parameter file.

## Staging

Illustrated below is the high level overview of the staging process for all 4 clients.

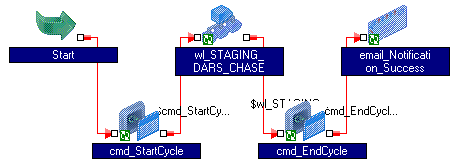


This process copies data from INFOPROD for all 4 clients and stages the data in the DARS environment. These tables that stage INFOPROD data in DART01D reside in the ETLMGR schema. Similar to non-AD tables, the ETLs that load the staging tables also make use of the virtual-switch of table partition. However, the staging tables go one step further, because they are sub-partitioned. With the help of this design, data is made available to the AD application constantly, even though a load is being performed in the background. This special logic will be discussed in a separate section in the document.

### Workflow wf\_STAGING\_DARS\_CHASE

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0005C |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_staging\_dars\_chase.ksh |
| **Schedule Information** | STWTFS @ 0500 EST |

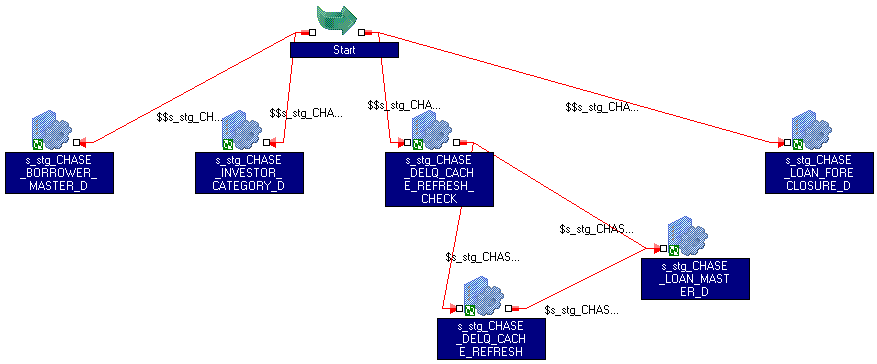
Below is a screenshot of the ETL workflow.



The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| Wl\_STAGING\_DARS\_CHASE | This worklet contains the ETL to extract data from various tables in MORTGMGR schema in INFOPROD & load the staging tables in ETLMGR schema in DART01P |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

The worklet “wl\_STAGING\_DARS\_CHASE” has been illustrated below.



The table below gives a summary of the functions of tasks contained in the worklet.

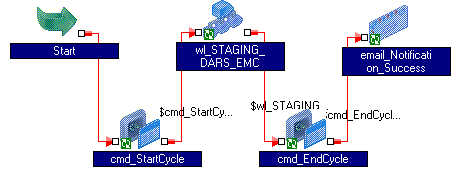
| Task Name | Task function |
| --- | --- |
| S\_stg\_CHASE\_BORROWER\_MASTER\_D | Loads data in the table BORROWER\_MASTER\_D in ETLMGR schema in DART01P from the tables in MORTGMGR schema at INFOPROD. |
| S\_stg\_CHASE\_INVESTORY\_CATEGORY\_D | Loads data in the table BORROWER\_MASTER\_D in ETLMGR schema in DART01P from the tables in MORTGMGR schema at INFOPROD. |
| S\_stg\_CHASE\_DELQ\_CACHE\_REFRESH\_CHECK | Checks if the view MSP\_DELQ\_GROUPINGS\_M has a greater RUN\_YR\_MO the value from the last ETL run |
| S\_stg\_CHASE\_DELQ\_CACHE\_REFRESH | Refreshes the cache on the table MSP\_DELQ\_GROUPINGS\_M, |
| S\_stg\_CHASE\_LOAN\_MASTER\_D | Loads data into the table LOAN\_MASTER\_D in ETLMGR schema in DART01P from the tables in MORTGMGR. The ETL would make use of the newly created cache, if the earlier cache was stale, else the ETL would make use of the existing cache. |
| S\_stg\_CHASE\_LOAN\_FORECLOSURE\_D | Loads data into the table LOAN\_MSP\_FORECLOSURE\_D in ETLMGR schema in DART01P, from the tables in MORTGMGR schema at INFOPROD. |

This workflow has “Suspend On Error” option checked. There is no dynamic parameter requirement in this process; therefore this process makes use of a static parameter file. This workflow makes use of a mapping variable to retain the value of the previous max RUN\_YR\_MO for creation of data cache for the table MSP\_DELQ\_GROUPINGS\_M.

### Workflow wf\_STAGING\_DARS\_EMC

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0005E |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_staging\_dars\_emc.ksh |
| **Schedule Information** | TWTFS @ 0500 EST |

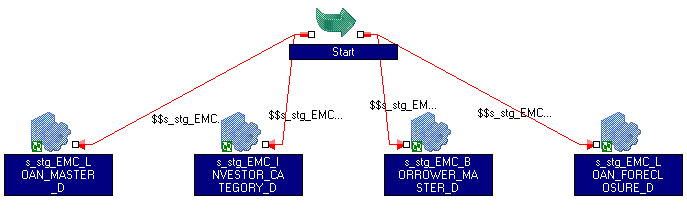
Below is a screenshot of the ETL workflow.



The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| Wl\_STAGING\_DARS\_EMC | This worklet contains the ETL to extract data from various tables in MORTGMGR schema in INFOPROD & load the staging tables in ETLMGR schema in DART01P |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

The worklet “wl\_STAGING\_DARS\_EMC” has been illustrated below.



The table below gives a summary of the functions of tasks contained in the worklet.

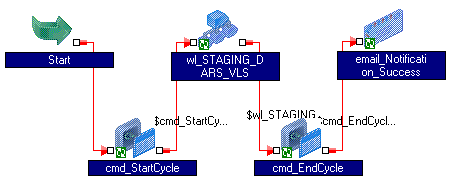
| Task Name | Task function |
| --- | --- |
| S\_stg\_EMC\_LOAN\_MASTER\_D | Loads data into the table LOAN\_MASTER\_D in ETLMGR schema in DART01P from the tables in ME589MGR. |
| S\_stg\_EMC\_INVESTORY\_CATEGORY\_D | Loads data in the table BORROWER\_MASTER\_D in ETLMGR schema in DART01P from the tables in ME589MGR schema at INFOPROD. |
| S\_stg\_EMC\_BORROWER\_MASTER\_D | Loads data in the table BORROWER\_MASTER\_D in ETLMGR schema in DART01P from the tables in ME589MGR schema at INFOPROD. |
| S\_stg\_EMC\_LOAN\_FORECLOSURE\_D | Loads data into the table LOAN\_MSP\_FORECLOSURE\_D in ETLMGR schema in DART01P, from the tables in ME589MGR schema at INFOPROD. |

This workflow has “Suspend On Error” option checked. There is no dynamic parameter requirement in this process; therefore this process makes use of a static parameter file.

### Workflow wf\_STAGING\_DARS\_VLS

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0005V |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_staging\_dars\_vls.ksh |
| **Schedule Information** | TWTFS @ 0500 EST |

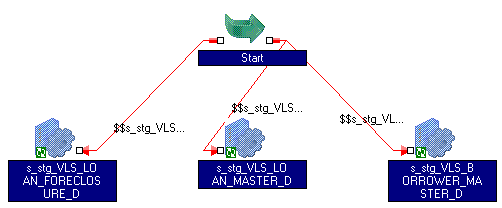
Below is a screenshot of the ETL workflow.



The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| Wl\_STAGING\_DARS\_EMC | This worklet contains the ETL to extract data from various tables in LOANMGR schema in INFOPROD & load the staging tables in ETLMGR schema in DART01P |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

The worklet “wl\_STAGING\_DARS\_VLS” has been illustrated below.



The table below gives a summary of the functions of tasks contained in the worklet.

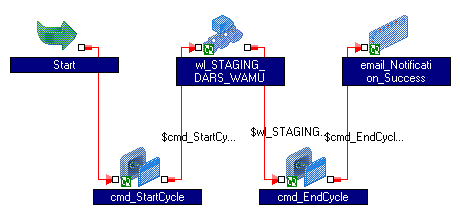
| Task Name | Task function |
| --- | --- |
| S\_stg\_VLS\_LOAN\_FORECLOSURE\_D | Loads data into the table LOAN\_VLS\_FORECLOSURE\_D in ETLMGR schema in DART01P, from the tables in LOANMGR schema at INFOPROD. |
| S\_stg\_VLS\_LOAN\_MASTER\_D | Loads data into the table LOAN\_MASTER\_D in ETLMGR schema in DART01P from the tables in LOANMGR. |
| S\_stg\_VLS\_BORROWER\_MASTER\_D | Loads data in the table BORROWER\_MASTER\_D in ETLMGR at DART01P from the tables in LOANMGR schema at INFOPROD. |

This workflow has “Suspend On Error” option checked. There is no dynamic parameter requirement in this process; therefore this process makes use of a static parameter file.

### Workflow wf\_STAGING\_DARS\_WAMU

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0005W |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_staging\_dars\_wamu.ksh |
| **Schedule Information** | STWTFS @ 0500 EST |

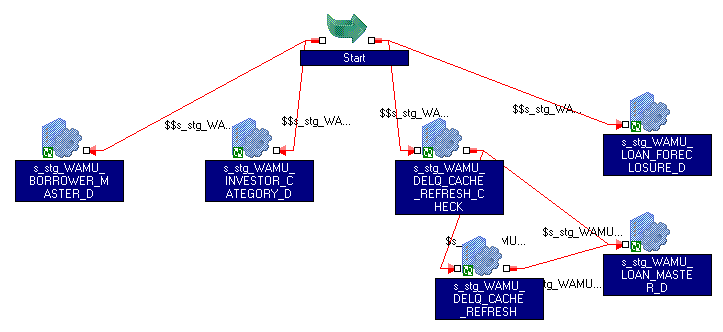
Below is a screenshot of the ETL workflow.



The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| Wl\_STAGING\_DARS\_WAMU | This worklet contains the ETL to extract data from various tables in MW156MGR schema in INFOPROD & load the staging tables in ETLMGR schema in DART01P |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

The worklet “wl\_STAGING\_DARS\_WAMU” has been illustrated below.



The table below gives a summary of the functions of the tasks contained in the worklet.

| Task Name | Task function |
| --- | --- |
| S\_stg\_WAMU\_BORROWER\_MASTER\_D | Loads data in the table BORROWER\_MASTER\_D in ETLMGR schema in DART01P from the tables in MW156MGR schema at INFOPROD. |
| S\_stg\_WAMU\_INVESTORY\_CATEGORY\_D | Loads data in the table BORROWER\_MASTER\_D in ETLMGR schema in DART01P from the tables in MW156MGR schema at INFOPROD. |
| S\_stg\_WAMU\_DELQ\_CACHE\_REFRESH\_CHECK | Checks if the view MSP\_DELQ\_GROUPINGS\_M has a greater RUN\_YR\_MO the value from the last ETL run |
| S\_stg\_WAMU\_DELQ\_CACHE\_REFRESH | Refreshes the cache on the table MSP\_DELQ\_GROUPINGS\_M, |
| S\_stg\_WAMU\_LOAN\_MASTER\_D | Loads data into the table LOAN\_MASTER\_D in ETLMGR schema in DART01P from the tables in MW156MGR. The ETL would make use of the newly created cache, if the earlier cache was stale, else the ETL would make use of the existing cache. |
| S\_stg\_WAMU\_LOAN\_FORECLOSURE\_D | Loads data into the table LOAN\_MSP\_FORECLOSURE\_D in ETLMGR schema in DART01P, from the tables in MW156MGR schema at INFOPROD. |

This workflow has “Suspend On Error” option checked. There is no dynamic parameter requirement in this process; therefore this process makes use of a static parameter file. This workflow makes use of a mapping variable to retain the value of the previous max RUN\_YR\_MO for creation of data cache for the table MSP\_DELQ\_GROUPINGS\_M.

## ORG employee sync

Illustrated below is the high level overview of the org-employee process.

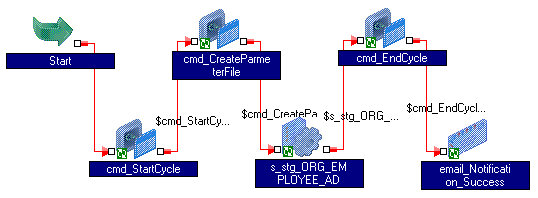


The table ORG\_EMPLOYEE at CHAPSAPP in BUSPROD is the table in which AD had it’s user information, until AD was moved to the new DARS environment. Currently the AD system makes use of a more normalized set of tables. However, there are still some reporting teams, who pull data from the ORG\_EMPLOYEE. This workflow copies only the basic information of an employee from DART01P into BUSPROD, which is common to all applications. Also, no updates are performed to existing records of employees in ORG\_EMPLOYEE, because the table is shared by multiple applications. The workflow works in an append-only mode.

### Workflow wf\_ORG\_EMPLOYEE

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0007 |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_org\_employee.ksh |
| **Schedule Information** | Daily @ 0100 EST |

Below is a screenshot of the ETL workflow.



The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| Cmd\_CreateParameterFile | Creates a dynamic parameter file for the CYCLE passed as parameter. This initiates a workflow wkf\_CmnCreateParameterFile, which creates the required parameter file in a pre-defined location. This process will be discussed in a separate document as mentioned earlier in this document. Also, this task appends a static parameter file “wf\_ORG\_EMPLOYEE\_global.parm” with the dynamically generated parameter file “wf\_ORG\_EMPLOYEE.parm”. |
| S\_stg\_ORG\_EMPLOYEE\_AD | Loads the table ORG\_EMPLOYEE in CHAPSAPP @ BUSPROD on an append-only mode, while sourcing from APPSHAREDMGR @ DART01P |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

## ENT tables sync

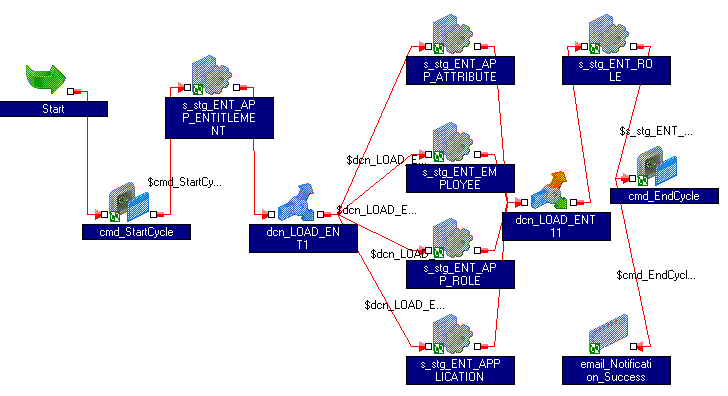
Illustrated below is the high level overview of the ENT table sync process.



### Workflow wf\_EMPLOYEE\_ENT

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0007\_ENT |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_employee\_ent.ksh |
| **Schedule Information** | Daily @ 0100 EST |

Below is a screenshot of the ETL workflow.



The sessions in this workflow load the ENT tables in CHAPSAPP @ BUSPROD, sourcing from the corresponding ENT tables in APPSHAREDMGR @ DART01P. The referential constraints are first disabled, before the load is performed and enabled back in the last session. The command tasks cmd\_StartCycle & cmd\_EndCycle create & close the entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter.

## ORG employee login audit

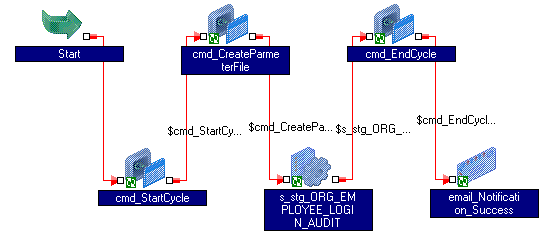
Illustrated below is the high level overview of this process.



### Workflow wf\_ORG\_EMPLOYEE\_LOGIN\_AUDIT

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0007\_OELA |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_org\_employee\_login\_audit.ksh |
| **Schedule Information** | Daily @ 0100 EST |

Below is a screenshot of the ETL workflow.



The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| Cmd\_CreateParameterFile | Creates a dynamic parameter file for the CYCLE passed as parameter. This initiates a workflow wkf\_CmnCreateParameterFile, which creates the required parameter file in a pre-defined location. Also, this task appends a static parameter file “wf\_ORG\_EMPLOYEE\_LOGIN\_AUDIT\_global.parm” with the dynamically generated parameter file “wf\_ORG\_EMPLOYEE\_LOGIN\_AUDIT.parm”. |
| S\_stg\_ORG\_EMPLOYEE\_LOGIN\_AUDIT | Loads the table ORG\_EMPLOYEE\_LOGIN\_AUDIT in CHAPSAPP @ BUSPROD on an append-only mode, while sourcing from APPSHAREDMGR @ DART01P. This ETL makes use of the delta-logic, therefore the need for a dynamic parameter file. |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

## PRC wrapper jobs

Some ETL processes in DARS have been created just as a wrapper to the stored procedure in the DARS environment. The overall structure of the workflows is identical, except for a few exceptions. Each of these jobs has its own Control-M job # and schedule. Illustrated below is a high level overview of the processes that fall under this category.



### Workflow wf\_PRC\_AD\_CLEAR\_TRIAL\_REQ\_DATE

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0008 |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_prc\_ad\_clear\_trial\_req\_date.ksh |
| **Schedule Information** | Daily @ 0850 EST |

Below is a screenshot of the ETL workflow.



The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| s\_ad\_PRC\_AD\_CLEAR\_TRIAL\_REQ\_DATE | Runs PRC\_AD\_CLEAR\_TRIAL\_REQ\_DATE in the ADMGR schema @ DART01P |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

### Workflow wf\_PRC\_AD\_PLAN\_BREAK\_UPDATE

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0009 |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_prc\_ad\_plan\_break\_update.ksh |
| **Schedule Information** | Daily @ 0845 EST |

Below is a screenshot of the ETL workflow.



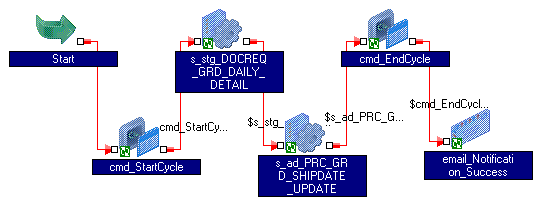
The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| s\_ad\_PRC\_AD\_PLAN\_BREAK\_UPDATE | Runs PRC\_AD\_PLAN\_BREAK\_UPDATEin the ADMGR schema @ DART01P |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

### Workflow wf\_PRC\_GRD\_SHIPDATE\_UPDATE

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0010 |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_prc\_grd\_shipdate\_update.ksh |
| **Schedule Information** | Daily @ 0855 EST |

Below is a screenshot of the ETL workflow.



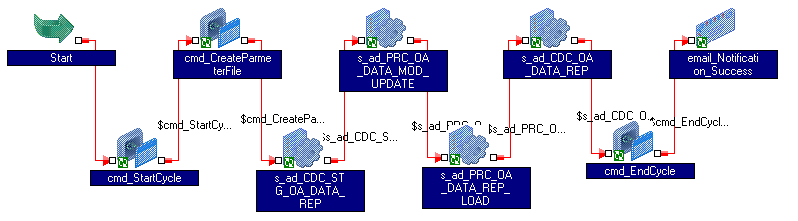
The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| s\_stg\_DOCREQ\_GRD\_DAILY\_DETAIL | Loads the table DOCREQ\_GRD\_DAILY\_DETAIL from DITMGR @ BUSPROD into ETLMGR @ DART01P. The target table is partitioned and the ETL makes use of the partition virtual switch. |
| s\_ad\_PRC\_GRD\_SHIPDATE\_UPDATE | Runs PRC\_GRD\_SHIPDATE\_UPDATE in the ADMGR schema @ DART01P |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

### Workflow wf\_PRC\_OA\_DATA

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0011 – Decommissioned |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_prc\_oa\_data.ksh |
| **Schedule Information** | Daily 0800 EST to 2300 EST, every 3 hours |

Below is a screenshot of the ETL workflow.

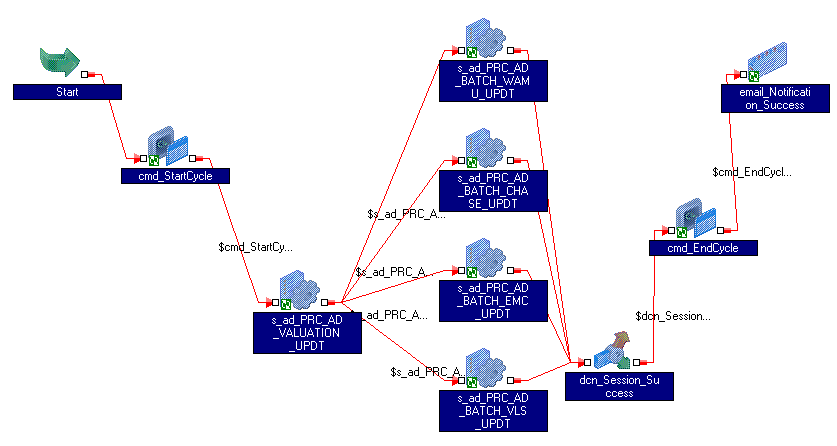


The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| cmd\_CreateParmeterFile` | Creates a dynamic parameter file for the CYCLE passed as parameter. This initiates a workflow wkf\_CmnCreateParameterFile, which creates the required parameter file in a pre-defined location. Also, this task appends a static parameter file “wf\_PRC\_OA\_DATA\_global.parm” with the dynamically generated parameter file “wf\_PRC\_OA\_DATA.parm”. |
| s\_ad\_CDC\_STG\_OA\_DATA\_REP | Performs a delta insert on the table STG\_OA\_DATA\_REP in the schema ADMGR @ DART01P, sourcing from CHAPSAPP @ BUSPROD |
| s\_ad\_PRC\_OA\_DATA\_MOD\_UPDATE | Runs PRC\_OA\_DATA\_MOD\_UPDATE in the ADMGR schema @ DART01P |
| s\_ad\_PRC\_OA\_DATA\_REP\_LOAD | Runs PRC\_OA\_DATA\_REP\_LOAD in the ADMGR schema @ DART01P |
| s\_ad\_CDC\_OA\_DATA\_REP | Performs a full-CDC loaded on the table OA\_DATA\_REP in the schema ADMGR @ DART01P, sourcing from CHAPSAPP @ BUSPROD. |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

### Workflow wf\_AD\_NIGHTLY\_BATCH\_UPDT

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0012 |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_ad\_nightly\_batch\_updt.ksh |
| **Schedule Information** | * TWTFS after all staging jobs * SUN after 5C & 5W |



The table below gives a summary of the functions of tasks contained in the workflow.

| Task Name | Task function |
| --- | --- |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| s\_ad\_PRC\_AD\_VALUATION\_UPDT | Runs PRC\_AD\_VALUATION\_UPDT in the package PKG\_AD\_NIGHTLY\_BATCH\_UPDT in the schema ADMGR @ DART01P |
| s\_ad\_PRC\_AD\_BATCH\_CHASE\_UPDT | Runs PRC\_AD\_BATCH\_CHASE\_UPDT in the package PKG\_AD\_NIGHTLY\_BATCH\_UPDT in the schema ADMGR @ DART01P |
| s\_ad\_PRC\_AD\_BATCH\_EMC\_UPDT | Runs PRC\_AD\_BATCH\_EMC\_UPDT in the package PKG\_AD\_NIGHTLY\_BATCH\_UPDT in the schema ADMGR @ DART01P |
| s\_ad\_PRC\_AD\_BATCH\_VLS\_UPDT | Runs PRC\_AD\_BATCH\_VLS\_UPDT in the package PKG\_AD\_NIGHTLY\_BATCH\_UPDT in the schema ADMGR @ DART01P |
| s\_ad\_PRC\_AD\_BATCH\_WAMU\_UPDT | Runs PRC\_AD\_BATCH\_WAMU\_UPDT in the package PKG\_AD\_NIGHTLY\_BATCH\_UPDT in the schema ADMGR @ DART01P |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

### Workflow wf\_PRC\_AD\_OA\_MODIFICATION\_TERMS

|  |  |
| --- | --- |
| **Control-M job name** | PDITD0013 |
| **UNIX wrapper script** | /etlapps/prod/code/dars/Scripts/wf\_prc\_ad\_oa\_modification\_terms.ksh |
| **Schedule Information** | Daily @ 0715 EST |

Below is a screenshot of the ETL workflow.



The table below gives a summary of the functions of tasks contained in the workflow.

|  |  |
| --- | --- |
| Task Name | Task function |
| cmd\_StartCycle | Creates an entry in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |
| s\_ad\_PRC\_AD\_OA\_MODIFICATION\_TERMS | Runs PRC\_AD\_OA\_MODIFICATION\_TERMS in the schema ADMGR @ DART01P. |
| Cmd\_EndCycle | Closes the entry for the CYCLE in the table METACYCLERUNSTTS in the DITMGR schema for the CYCLE passed as parameter. |

# Supplemental Process Detail

## Dynamic Parameter Files

This section discusses about the dynamic parameter file generation process, which has been used for workflows that contain sessions that contain a delta-capture load mechanism. The set of tables that contain the parameter entries for a workflow have been illustrated below.



Given below is a brief description of each of these tables that exist in the DITMGR schema in the BUSxx database.

|  |  |  |
| --- | --- | --- |
| Table Name | Description | Maintained by? |
| METAAPPERROR | Not used | NA |
| METACYCLE | Contains an entry for the workflow | Manual |
| METACYCLERUNSTTS | Contains an entry for every run of the workflow | Unix Script |
| METADYNMCPARAMGRP | Not used | NA |
| METADYNMCPARAMGRPXREF | Not used | NA |
| METADYNMCPROCESSPARAM | Not used | NA |
| METAETLLASTRUNDTTM | Intermediate table that stores the start time of a session. This is related to the delta capture logic. | ETL programs |
| METAETLLASTRUNDTTMHIST | Future use | ETL program |
| METAPARAMFILE | Contains the parameter file information for a workflow. | Manual |
| METAPROCACTN | Not used | NA |
| METAPROCESS | Contains the sessions in a workflow | Manual |
| METAPROCESSCYCLEXREF | Acts like a junction table, which helps in defining the sessions that come under a workflow in the parameter file | Manual |
| METAPROCESSERROR | Not used | NA |
| METAPROCESSLOG | Not used | NA |
| METAPROCESSPARAMGRPXREF | Acts like a junction table, which helps in mentioning the worklet & workflow under which a session comes under. | Manual |
| METAPROCESSTYPE | Helps in categorizing the process as an Informatica ETL process or as a UNIX script | Manual, but one time setup. |
| METASTATICPROCESSPARAM | Contains all the parameters associated with a session. | Manual & ETL programs |

The parameter files are created from these META tables with the help of the workflow “wkf\_CmnCreateParameterFile” in the folder “map\_DIT\_DARS”. This workflow is called in the command task “cmd\_CreateParmeterFile” in the workflows that require dynamic parameters. One of the workflows that require dynamic parameter file generation is the “AD sync” process. The UNIX script that is called from within the command task “cmd\_CreateParmeterFile” to invoke the workflow that creates the parameter file is given below.

$PMRootDir/code/common/Scripts/createParamFile.ksh wf\_AD\_DARS.parm dars

The UNIX scripts related this process have been listed in the table below.

These scripts reside in the folder “/etlapps/prod/code/common/Scripts” @ ret4a137v .

|  |  |
| --- | --- |
| Script Name | Description |
| startCycle.ksh | Creates or opens an entry in the table METACYCLERUNSTTS for a cycle/workflow run |
| endCycle.ksh | Closes an open entry in the table METACYCLERUNSTTS for a cycle/workflow run |
| createParamFile.ksh | Calls the workflow that creates the parameter file for the specified cycle/workflow |
| db\_info\_DIT | Contains encrypted database passwords to be used by the UNIX scripts, while connecting to the database |
| dit\_unix\_variables | Contains variables/constants that are used across the DIT workspace |
| functions.ksh | Contains functions that are used across the DIT workspace |
| ENV.env | Environment file, which primarily has the information about the database that needs to be accessed to fetch the parameter entries |
| start\_infa\_workflow.ksh | This is a generic script that kicks off a workflow. This makes of the PMCMD command to start a workflow in asynchronous or in a non-wait mode, due to Control-M limitations. This script also recovers a workflow, if it is already in a “Suspended” state. |

## Virtual Partition Switching

This load mechanism strategy has been used in Non-AD & Staging ETL load programs. The primary objective behind using such a load strategy is to make the data available to the users, even when a truncate & load is being performed in the background. This is achieved by partitioning the table, so the application looks at one of the partitions, while the ETL loads the other. This is achieved by having a control table, which indicates the most recently loaded partition in the table and a view on the partitioned table, which presents only the most recently loaded partition to the application. The following diagram illustrates this process.

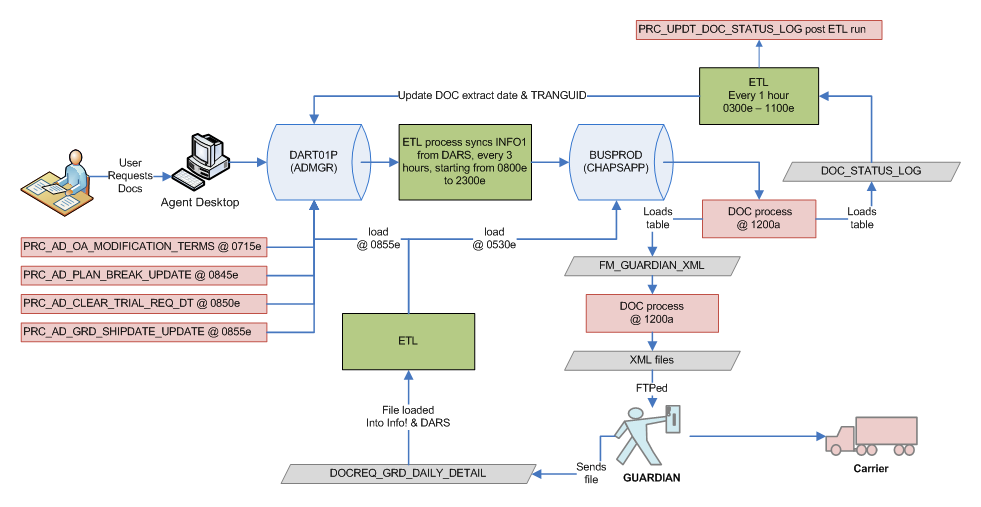


## Document Processing

The following steps detail the complete document processing cycle.

* 1. User requests for documents to be processed for a loan through the AD application
  2. The request is recorded in the DARS environment
  3. The AD sync ETL carries over this request over to the Info-One environment
  4. The request is picked up by the doc-process at night and the following is performed
     1. Loads the table FM\_GUARDIAN\_XML with the TRANGUID for the request
     2. Loads the table DOC\_STATUS\_LOG with the document extract date & corresponding TRANGUID. This is required to sync back this information into DARS using the procedure PRC\_UPDT\_DOC\_STATUS\_LOG.
     3. Create XML files sourcing from FM\_GUARDIAN\_XML using the procedures listed below.
     4. XML files are FTPed to GUARDIAN
  5. GUARDIAN sends back a file, which is loaded into the table DOCREQ\_GRD\_DAILY\_DETAIL in Info-One & DARS
  6. The following procedures are run on schedule to complete the doc request cycle.
     1. PRC\_AD\_OA\_MODIFICATION\_TERMS
     2. PRC\_AD\_PLAN\_BREAK\_UPDATE
     3. PRC\_AD\_CLEAR\_TRIAL\_REQ\_DT – Clears the request date for trial docs where required criteria is not met.
     4. PRC\_AD\_GRD\_SHIPDATE\_UPDATE – Updates the actual shipment date information of the docs into DARS. This updates tables MOD\_FORM, MOD\_FORM\_STATUS\_JOIN etc.

The document processing cycle has been illustrated at a high level below.



## UNIX scripts & paths

The table below lists the paths in which scripts related to DARS are located.

|  |  |
| --- | --- |
| Description | Path |
| DARS parameter files | /etlapps/prod/code/dars/ParmFiles |
| DARS wrapper scripts to kick-off the workflows | /etlapps/prod/code/dars/Scripts |
| Common scripts across all projects | /etlapps/prod/code/common/Scripts |
| UNIX scripts log file location | /etlapps/prod/data/common/scriptlogs |
| Infa folder tree structure for DARS | /etlapps/prod/data/dars |

# Schedule Information

|  |  |  |  |
| --- | --- | --- | --- |
| Control-M Job # | Job Description | UNIX script name | Schedule Info |
| PDITD0002 | AD Sync | wf\_ad\_dars\_preload.ksh | Daily, every 3 hours  0800est ~ 2300est |
| PDITD0003 | DOC Field Update | wf\_ad\_doc\_field\_update.ksh | Daily, every 1 hour  0300est ~ 1100est |
| PDITD0004 | Non AD | wf\_non\_ad\_dars.ksh | Daily  0000est |
| PDITD0005C | Stage (CHASE) | wf\_staging\_dars\_chase.ksh | STWTFS  0500est |
| PDITD0005E | Stage (EMC) | wf\_staging\_dars\_emc.ksh | TWTFS  0500est |
| PDITD0005V | Stage (VLS) | wf\_staging\_dars\_vls.ksh | TWTFS  0500est |
| PDITD0005W | Stage (WAMU) | wf\_staging\_dars\_wamu.ksh | STWTFS  0500est |
| PDITD0007 | ORG Employee Sync | wf\_org\_employee.ksh | Daily  0100est |
| PDITD0007\_ENT | ENT Tables Sync | wf\_employee\_ent.ksh | Daily  0100est |
| PDITD0007\_OELA | ORG Employee  Login Audit | wf\_org\_employee\_login\_audit.ksh | Daily  0100est |
| PDITD0008 | PRC – Clear Trial Request Date | wf\_prc\_ad\_clear\_trial\_req\_date.ksh | Daily  0850est |
| PDITD0009 | PRC – Plan Break Update | wf\_prc\_ad\_plan\_break\_update.ksh | Daily  0845est |
| PDITD0010 | PRC – Guardian Shipdate Update | wf\_prc\_grd\_shipdate\_update.ksh | Daily  0855est |
| PDITD0011 | PRC – Option Arm Data | wf\_prc\_oa\_data.ksh | Decommissioned |
| PDITD0012 | PRC – AD Nightly Batch Update | wf\_ad\_nightly\_batch\_updt.ksh | Decommissioned |
| PDITD0013 | PRC – OA Modification Terms | wf\_prc\_ad\_oa\_modification\_terms.ksh | Daily  0715est |

# Troubleshooting

The following sections discuss the actions to be taken, when the ETL jobs end in failure. The corrective action will vary depending on the cause of ETL job failure, but the sections below will discuss some common failures that could occur and the steps to take before the jobs are recovered or even restarted.

## PDITD0002

If the cause of failure is a “unique constraint violation”, this could be related to an existing issue in the delta-insert logic (3Di) in the ETLs. The underlying cause for this issue is that the ETLs make use of the Infa server time, but the AD application makes use of the application server’s time and these two times are not in synch. If such a failure occurs in the production environment, then the rows causing the constraint violation would need to be deleted from the target. These rows can be identified by running the SQL from SQ transformation on BUSPROD.

Other failures in this job would need to be dealt with, depending on the nature of the failure. The sessions that are other than SNPSHT, AUDIT, COMMENT & LOGGER follow a full-CDC load mechanism, but the others in general have a delta-insert load mechanism. Such jobs cannot be restarted; because it would lead to duplicate rows in the target, if the target table does not have a PK defined or would lead to a unique constraint violation, if PK is defined. If the safest option was to perform a truncate & load into the target, the $$LOAD\_TYPE mapping parameter would need to be set to 1 for that session in the parameter file. This can be set by updating the table METASTATICPROCESSPARAM and recreating the parameter file.

## PDITD0003

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure.

## PDITD0004

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure.

## PDITD0005C

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure. This job has a dependency on BDE, which could get delayed at times. Under such circumstances, this job may have to be manually restarted.

## PDITD0005E

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure. This job has a dependency on BDE, which could get delayed at times. Under such circumstances, this job may have to be manually restarted.

## PDITD0005V

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure. This job has a dependency on BDE, which could get delayed at times. Under such circumstances, this job may have to be manually restarted.

## PDIT0005W

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure. This job has a dependency on BDE, which could get delayed at times. Under such circumstances, this job may have to be manually restarted.

## PDITD0007

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure.

## PDITD0007\_ENT

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure.

## PDITD0007 \_OELA

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure.

## PDITD0008

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure. The core logic of this job resides in the PL/SQL procedure, which is called by the wrapper ETL job.

## PDITD0009

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure. The core logic of this job resides in the PL/SQL procedure, which is called by the wrapper ETL job.

## PDITD0010

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure. The core logic of this job resides in the PL/SQL procedure, which is called by the wrapper ETL job.

## PDITD0012

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure. The core logic of this job resides in the PL/SQL procedure, which is called by the wrapper ETL job.

This job has been decommissioned, because this feature has been incorporated within the AD application.

## PDITD0013

This process has been designed to be restartable, therefore the workflow can be recovered, if the job ends in failure. However, prior analysis would need to be performed to understand the cause of failure. The core logic of this job resides in the PL/SQL procedure, which is called by the wrapper ETL job.

# Service Level Agreement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control-M Job # | Frequency | Expected Start Time | Expended End Time | SLA |
| PDITD0002 | Daily, every 3 hours | 0700 est | 2300 est | 1900 est |
| PDITD0003 | Daily, every hour | 0300 est | 1100 est | 1500 est |
| PDITD0004 | Daily | 0000 est | NA | 0500 est |
| PDIT0005C | Daily | 0500 est | NA | 0900 est |
| PDIT0005E | Daily | 0500 est | NA | 0900 est |
| PDITD0005V | Daily | 0500 est | NA | 1000 est |
| PDITD0005W | Daily | 0500 est | NA | 0900 est |
| PDITD0007 | Daily | 0100 est | NA | 0800 est |
| PDITD0007\_ENT | Daily | 0100 est | NA | 0800 est |
| PDITD0007\_OELA | Daily | 0100 est | NA | 0800 est |
| PDITD0008 | Daily | 0850 est | NA | 1000 est |
| PDITD0009 | Daily | 0845 est | NA | 1000 est |
| PDITD0010 | Daily | 0855 est | NA | 1100 est |
| PDITD0011 | Daily, every 3 hours  (decommissioned) | 0800 est | 2300 est | 0500 est |
| PDITD0012 | Daily after staging jobs  (decommissioned | NA | NA | 1000 est |
| PDITD0013 | Daily | 0715 est | NA | 1000 est |

# High Level Technical Architecture & Contact Information

The HE CMS process runs using the following software tools/architecture:

* Informatica Power Center 11.6.1
* IBM AIX OS
* Oracle 10g

The following table depicts the management staff responsible for the various aspects of the project.

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Name | Role | Areas of responsibility |
| 110030 | Krishna Nutakki | Vice President | Project Management & Support |

# Task Overview

NA

# Pre-Installation Checklist

NA

# Configuration & Setup

NA

## Operating System

NA

## Database Server

NA

## Data Migration

NA

## Web Server

NA

## Application Server

NA

### Application Server Setup

NA

### Database Connection Setup

NA

### Front-end / GUI setup

NA

# Security Issues

NA

# Other Packages

NA

# Application Login IDs

This section states the list of all login ids and privileged accounts used in the application.

|  |  |  |
| --- | --- | --- |
| **Application** | **Systems** | **Login-ID** |
| FTP | NA | NA |
| Oracle | BUSPROD | DITAPP |
| Oracle | DART01P | DITAPP |
| UNIX | ETL server | DITMGR |
| Informatica | DARS | DITMGR |

# Content and Application Directory Configuration

This section states the list of directories for the application.

|  |  |  |
| --- | --- | --- |
| **Components of Application** | **Path** | **Owner** |
| Informatica repository details | Repository: DIT\_PROD  Integration Service:  Domain:  Gateway Host:  Folder |  |
| All scripts location to run the process | Refer to section “UNIX scripts & paths” |  |
| Session log file path |  |  |
| Source file path |  |  |
| Target file path |  |  |
| Target FTP location |  |  |
| Archive file location |  |  |
| Script log file location |  |  |

# Production Support

## Alerts generated by system

A failure alert indicating that there is an issue with the execution of the process and needs immediate attention.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Failure Category | Probable Reason | Action |
| 1 | Failure | Any job failure | Contact level 2 support and re-run the process when Informatica server is up |
| 2 | Threshold | The BDE jobs haven’t finished | If the staging jobs do not finish by 1000 EST, check if there is a delay with BDE process |

## Informatica job monitoring / Visual confirmation

Informatica Jobs will be monitored using Workflow Monitor and any failure while processing or any deviation from scheduled run timings/SLA will be captured and reported. The On-call support person will also be contacted in such scenarios thru a pager. This kind of monitoring will ensure that issue like Informatica Server is down are captured and reported.

## General Tier 2 Escalation Procedures

This section indicates the escalation procedure pertaining to production support calls based on type of problem.

|  |  |  |  |
| --- | --- | --- | --- |
| Tier 2 and steady state contact list | | | |
| Escalation level | Role | Primary Contact # | Comments |
| 1 | Level 2 support | 469-236-8821 | Sajjan Janardhanan (cell) |
| 2 | Manager – Default IT |  | Yugandhar Mudedla |
| 3 | Manager – 2nd point escalation |  | Krishna Nutakki |

### Escalation guidelines

Please contact Sajjan Janardhanan. The ongoing escalation guidelines are TBD.

## Security Administration Support

NA

## Support Contact Information

|  |  |  |  |
| --- | --- | --- | --- |
| Server & Database Contact List | | | |
| Escalation level | Role | Primary Contact # | Comments |
| 3 | Unix SA | 614 244 0688 | Ramana Malakalapalli |
| 3 | Oracle DBA | ? | ? |
| 3 | Informatica Administrator | ? | Kishor Sonawane |

## Business Contact

|  |  |  |  |
| --- | --- | --- | --- |
| Group | Contact Person | Primary Contact # | Role |
| Agent Desktop | Chris Hostetter |  | Business Analyst |
| AD documents | Swathy Pasham |  | Business Analyst |
| AD documents | Adnan Jaffrey |  | Business Analyst |
| AD documents | Thomas Archer |  | Business Analyst |

# Appendix

Following is the list of the additional or external teams that need to be contacted in case of support/recovery issues.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name of the team | Primary contact | Secondary contact |
| 1 | Application team |  |  |
| 2 | Connectivity services |  |  |
| 3 | Info-One |  |  |