

## Table of Contents

<b>1</b>	End of Day Info Processes/Scripts and Activities.....	4
<b>1.1</b>	Perform back-up .....	13
<b>1.1.1</b>	ShutaimXX.bat.....	13
<b>1.1.2</b>	AimXXX_backup.bat.....	13
<b>1.2</b>	EA1.BAT.....	14
<b>1.2.1</b>	EA1_SOFTWARE_UPD.BAT.....	15
<b>1.2.2</b>	EA1.SQL.....	15
<b>1.3</b>	Automatic termination of clients no longer eligible.....	16
<b>1.3.1</b>	EA1_TERM_EOD.SQL.....	16
<b>1.3.2</b>	EA1_TERM_WL_CLIENTS.SQL.....	22
<b>1.4</b>	Perform automatic category changes.....	26
<b>1.4.1</b>	EA1_CAT_EOD.SQL.....	26
<b>1.4.2</b>	EA1_CAT_SYNC_EOD.SQL.....	29
<b>1.5</b>	Mark appointments as Missed or Kept.....	31
<b>1.5.1</b>	EA1_APPT_EOD.SQL.....	31
<b>1.6</b>	Delete information no longer used by the system .....	34
<b>1.6.1</b>	EA1_OTHER_PURG.SQL.....	34
<b>1.7</b>	Set exclusively breastfeeding clients to active status .....	40
<b>1.7.1</b>	EA1_IEN_ACTIVE.SQL.....	40
<b>1.8</b>	Process any pending transfer requests .....	41
<b>1.8.1</b>	EA1_TRANSFER.SQL .....	41
<b>1.9</b>	Generate notices, letters, and reports.....	42
<b>1.9.1</b>	CS_LETTER_OF_TERM.FMX.....	42
<b>1.10</b>	Gather updated client information, new issuance information, void information, and requests for transfer.....	46
<b>1.10.1</b>	EA1_TAB_UP.SQL.....	46
<b>1.10.2</b>	EA1_EXPORT_TAB.TXT.....	53
<b>1.11</b>	Initiate Polling Process .....	54
<b>1.11.1</b>	EC1.BAT.....	54
<b>1.12</b>	Delete information no longer used by the system.....	55
<b>1.12.1</b>	EC1.SQL .....	55
<b>1.12.2</b>	EC1_OTHER_PURG.SQL.....	56
<b>1.12.3</b>	EC1_STALE_DATE_FI.SQL.....	58
<b>1.13</b>	Compile direct payment FI information and send to bank.....	60
<b>1.13.1</b>	EC1_CTRL_CHK.SQL .....	60
<b>1.14</b>	Compile vendor information to send to bank .....	62
<b>1.14.1</b>	EC1_VENDOR_FSMC.BAT.....	62
<b>1.14.2</b>	EC1_VENDOR_FSMC.SQL .....	63
<b>1.14.3</b>	EC1_PGA_FSMC.BAT .....	65
<b>1.14.4</b>	EC1_PGA_FSMC.SQL.....	66
<b>1.15</b>	EC2.BAT .....	68
<b>1.16</b>	Consolidate participant information .....	70
<b>1.16.1</b>	EC_TRGOFF.SQL .....	70
<b>1.16.2</b>	EC2_SP_TRNC.SQL .....	70
<b>1.16.3</b>	EC2_TRNC.SQL.....	72
<b>1.16.4</b>	EC2_AGENCY.SQL.....	78
<b>1.16.5</b>	EC2_DELTRIG.SQL .....	80

1.16.6	EC2_DELTRIG_AU.SQL.....	81
1.16.7	EC2_IN_TAB.SQL .....	83
1.16.8	EC2_TAB_UP.SQL .....	89
1.16.9	EC2_ST_CLIENTS_SUM.SQL.....	95
1.16.10	EC2B.SQL.....	96
1.16.11	EC2_DEL_CL_FUT_RECS.SQL .....	97
1.16.12	EC_TRGON.SQL.....	98
1.17	Financial - process totals and store in F_CASELOAD table. Totals are based on priority, language, caseload type, ethnic group and poverty level.....	99
1.17.1	EC2_CL_REC_PROCESS.SQL .....	99
1.17.2	PROCEDURE EODP_CL_REC_PROC_P1.....	99
1.17.3	PROCEDURE EODP_I_U_CASELOAD_P1 .....	100
1.17.4	EC2.SQL .....	102
1.18	Update caseload assignment information from the clinics .....	103
1.18.1	EC2_SYNC_CLINIC_ASSIGNMENT.SQL.....	103
1.19	Consolidate and update all food instrument data to calculate FI obligation value .....	105
1.19.1	EC2_FOOD_UP.SQL.....	105
1.19.2	EC2_FI_UPD.SQL.....	107
1.20	Search for potential dual enrollment clients .....	111
1.20.1	EC2_DUAL_ENROLL.SQL.....	111
1.21	Compile new issuance and void information to send to bank.....	113
1.21.1	EC2_ISSUE_FSMC.SQL.....	113
1.22	EC4.BAT .....	115
1.23	Create consolidated record of all FI issuance and redemption information and create vendor payment records.....	117
1.23.1	EC4_BANK.SQL .....	117
1.23.2	EC4_BNK_POST.SQL .....	118
1.23.3	Inputs .....	118
1.23.4	EC4.SQL .....	122
1.24	Create date of current food funds available from all sources as well as current food obligations and expenditures to establish current cash position.....	123
1.24.1	EC4_CASHFLOW_UPD.SQL.....	123
1.25	Financial (end of month) - Populate F_INCOME to POVERTY table. These values are used in populating of the caseload table.....	129
1.25.1	EC4_MON_END.SQL .....	129
1.26	Vendor (end of month) calculate and update peer group averages.....	131
1.26.1	EC4_VEN_PEER_FI.SQL .....	131
1.26.2	EC4_VEN_LOC_AGCY_FI.....	134
1.27	Vendor (end of month) run and print analysis factor reports.....	137
1.27.1	EC4_VEN_PRICE.SQL .....	137
1.27.2	EC4_VEN_RISK_HISTORY.SQL .....	141
1.28	Gather new/updated data to be sent to the agencies.....	142
1.28.1	EC4_TRNC.SQL.....	142
1.28.2	EC4_TAB_DOWN.SQL .....	143
1.28.3	EC4_AGCY_BEING_PROCESSED.SQL .....	147
1.29	Prepare archival retrieval data .....	149
1.29.1	EC1_PRG_ARCHV.SQL.....	149
1.29.2	EC4_RETRIEVE.SQL .....	155
1.29.3	EC4_INS_RETR_CLI.SQL .....	156
1.29.4	EC4_INS_RETR_VEN.SQL.....	158
1.30	Print out status logs.....	161

1.30.1	EC5.BAT .....	161
1.30.2	EC_PRINT.BAT .....	161
1.31	Consolidate information from central .....	163
1.31.1	EA3.BAT .....	163
1.31.2	EA_TRGOFF.SQL .....	164
1.31.3	EA3_TRNC.SQL .....	164
1.31.4	EA3_PRE.SQL .....	170
1.31.5	EA3_DELTRIG.SQL .....	170
1.31.6	EA3_IN_TAB.SQL .....	172
1.31.7	EA3_TAB_UP.SQL .....	183
1.31.8	EA_TRGON.SQL .....	195
1.32	Clinic serial number replenish .....	196
1.32.1	EC4_CL_SER_REP.SQL .....	196
1.33	Update caseload assignment information from the clinics .....	198
1.33.1	EA3_SYNC_LA_ASSIGNMENT.SQL .....	198
1.33.2	EA3.SQL .....	198
1.34	Send redemption/rejection information to agencies .....	200
1.34.1	EA3_PROC_FI.SQL .....	200
1.35	Transmit archival clients and update records at agencies .....	204
1.35.1	EA3_PRG_ARCHV.SQL .....	204
1.35.2	EA3_RETRIEVE.SQL .....	207
1.35.3	EA3_INS_RETR_CLI.SQL .....	208
1.35.4	EA3_INS_RETR_VEN.SQL .....	210
1.36	Print out status logs of the process .....	212
1.36.1	EA_PRINT.BAT .....	212
1.37	RUN_CASELD.CMD .....	214
1.37.1	Processing .....	214
1.37.2	INSERT_MONTHLY_CASELOADS.SQL .....	214
1.37.3	Outputs .....	215

## 1 End of Day Info Processes/Scripts and Activities

### *Overview*

The End of Day Window in the System Administration module permits the user to initiate the end of day processing procedures required at different host computer sites. Since the AIM system architecture is a distributed client/server system that will utilize 21 Local Agency/Clinic servers and one Central Operations server, this processing runs on a nightly basis (every night of the week, excluding Sunday) to:

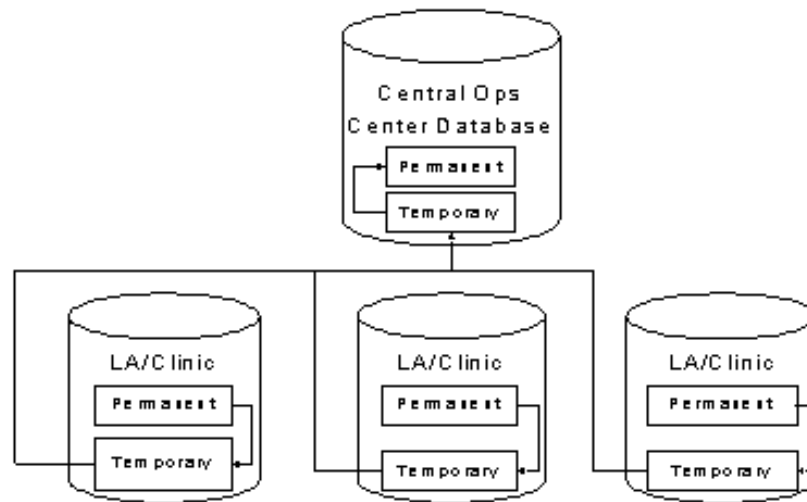
- \*• synchronize data at the Local Agency/Clinic Servers to the data at the Central Operations Server
- \*• perform automated participant data updates such as category changes and appointment status
- \*• delete unnecessary information to improve system performance
- \*• generate log files
- \*• send/receive food instrument issuance and reconciliation information with the banking intermediary FSMC
- \*• archive and retrieve archived participant records
- \*• update caseload information at the local and central database levels
- \*• update vendor peer group averages data

This screen allows the user to initiate the process manually, though each Local Agency/Clinic's AIM database can have a pre-programmed time that the End of Day process initiates automatically.

### **Flow of Data through End of Day**

The End of Day processing is shown in a table format at the end of this narrative. It provides the reader with a logical representation of the processing and scripts involved in the End of Day Process, separated as much as possible by the location at which they occur (Local Agency/Clinic, Central or Bank sites). In the physical running of the End of Day processing, the scripts don't run in this order. Some scripts run concurrently in the effort to accommodate time constraints. To view a detail of the order of the scripts, please see Appendix D - End of Day Script Run Order.

As discussed above, the End of Day processing runs in order to perform several automated updates, including synchronizing the data on the AIM distributed WIC system, so that the Local Agency/Clinic servers, Central server and Banking Intermediary (FSMC) all have updated information.



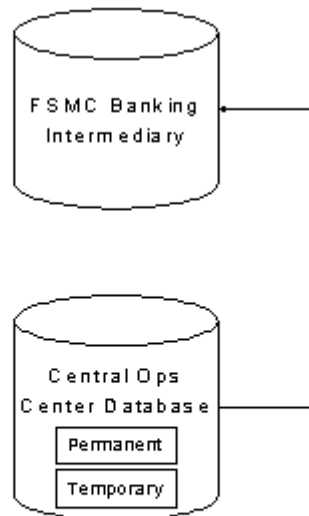
**Step 1** - Data is transferred from permanent tables to temporary tables at the Local Agency/Clinic level. From there files are zipped and sent up to a temporary table at the Central Server level. Here the files are moved into a permanent table at the Central Server.

The processing begins at the Local Agency/Clinic Servers with the system performing updates to the Local Agency/Clinic data, such as: performing automatic category changes and marking participant appointments as Missed or Kept. These updates are done within the LA/Clinic servers to the permanent system tables that reside on each server database. When the updates are done, the system gathers participant and food instrument information for transmission to the Central Server. The information is gathered together by copying the updated permanent food instrument and participant table information in the LA/Clinic servers, then copying the tables into temporary output tables. The tables are listed in a parameter file and are exported to a dmp file with that particular Local Agency's number as the filename and copied to the central server to be imported into the Central Operations Server database.

Coinciding with the Local Agency/Clinic servers performing updates, the Central server begins to delete unnecessary information on the Central server through the deletion of records from permanent tables stored on the Central Operations Center database. The system then begins to prepare archive retrieval data at the Central database by retrieving the archived vendor and participant records from tape, storing them in permanent tables on the Central Server, and then copying them into temporary tables in preparation for transfer to the LA/Clinic Servers. Also, the system compiles food instrument data (for Compliance Buys) and vendor information for transmission to the Banking Intermediary. The system compiles the food instrument data and the vendor information into separate temporary text files in preparation for transmission. New temporary files are created each time End of Day runs.

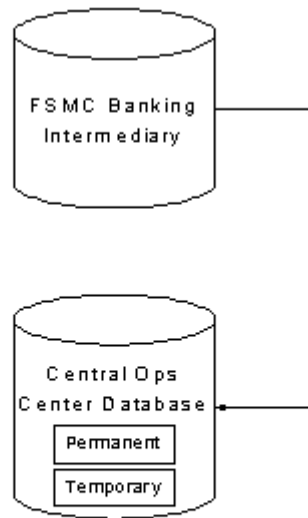
Both the updates at the LA/Clinic Databases and the compilation of information at the Central Operations Server Database haven't caused any information to move across the Arizona WIC WAN. At this point, the information is simply being formatted and prepared for transmission.

At a predetermined time (i.e. 7:30pm the Central Server searches for the dmp file by looking for the particular Local Agency number prefix and then verifying its existence by returning a value of 'True' to the search procedure. The Central server then performs consolidation and update processing of caseload, financial, food instrument and participant data by comparing the updated information to the Central Operations Server database permanent files and copying the more recent data from the temporary tables to the permanent tables .



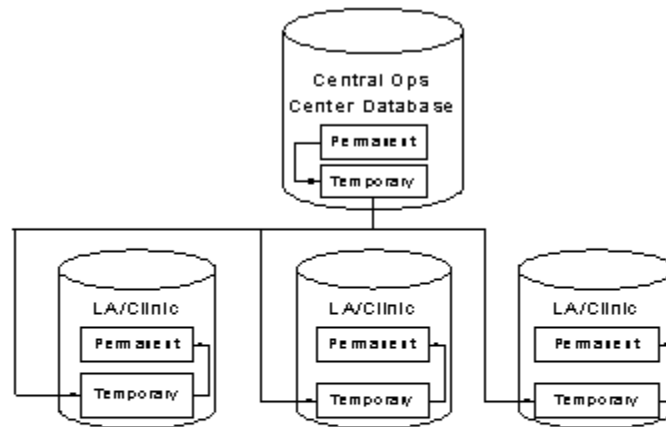
**Step 2** - Files are sent from the Central Server up to the FSMC

When the updating and processing of data at the Central Server is completed in the previous step the Central Operations Server Database compiles the new issuance and void data received from the LA/Clinics into an output temporary file. This food instrument information, along with the vendor information gathered earlier just after End of Day initiation is ready to be transmitted to the bank (this time without the benefit of a zip format). The system polls the banking intermediary and sends the updated information.



**Step 3** - Files are sent back down from the  
FSMC to the Central Server

While performing updates at FSMC, the system automatically picks up information at the bank (see Appendix B) and then consolidates and updates payment record and vendor information at the Central server. The updated temporary files received from FSMC are also in an unzipped format and the system compares the updated temporary information against the permanent tables and performs updates accordingly.



Step 4 - Data is transferred from a permanent table to a temporary table at the Central Server.  
 From there files are zipped and sent back down to temporary tables at the Local Agency/Clinic level.  
 Here the files are moved back into permanent tables.

After finalizing consolidation of the payment and vendor information at the Central site, the system prepares a dmp file containing updated caseload and food instrument redemption and rejection information. As well, archived client information that was requested is included in the dmp file. The dmp file is appropriately named based upon the LA/Clinic numbers. When the LA/Clinic receives its dmp file, the LA/Clinic database, compares the temporary tables within it to the permanent tables on the LA/Clinic database and performs updates accordingly.

1. The Oracle export (hot backup) is done at 2:30am from Tuesday until Saturday. It backs up the agency database, and shuts down the database for cold backup.
2. The NT Backup runs at 3:00am from Tuesday until Saturday. It backs up the c:\ and d:\ drives of the Local Agency server so that all the data is available on the backup tape. Tapes are then stored on and offsite according to Arizona policy and procedures.
3. The "at" job for rebuilding indexes is done on every Sunday at 7:00 am.
4. The "at" job for analyzing AIM schema is run on every Monday.
5. Database startup job runs at 5:30am from Tuesday until Saturday.

The End of Day processing will perform the following tasks utilizing the scripts listed each time it runs:

Processes	Script	LA	Central	Bank
Perform backup	aimXX_backup.cmd shutaimXX.cmd	x		
Automatic termination of participants no longer eligible (omit if done during clinic day)	EA1_TERM_EOD.SQL, EA1_TERM_WL_CLIENTS.S QL	x		
Perform automatic category changes	EA1_CAT_EOD.SQL	x		
Mark appointments as Missed or Kept	EA1_APPT_EOD.SQL EA1_NEW_APPT_EOD.SQL EA1_NEW_CLASS_EOD.SQL	x		
Delete information no longer used by the system (Includes participants with no Certification Record after 60 days on system)	EA1_OTHER_PURG.SQL,	x		
Set exclusively breastfeeding participants to active status	EA1_IEN_ACTIVE.SQL	x		
Void FIs not deposited after the stale date	EC1_STALE_FI.SQL		x	
Gather updated participant information, new issuance information, void information, requests for transfer, and outreach org. information	EA1_TAB_UP.SQL EA2_EXPORT_TAB.TXT	x		
Delete information no longer used by the system	EC1_OTHER_PURG.SQL	x	x	
Compile direct payment and revalidated FI information and send to bank	EC1_CTRL_CHK.SQL		x	x
Compile vendor information to be sent to the bank	EC1_VENDOR_FSMC.BAT, EC1_PGA_FSMC.SQL, EC1_VENDOR_FSMC.SQL,		x	
Consolidate participant information	EC2_SP_TRNC.SQL, EC2_TRNC.SQL, EC2_DELTRIG.SQL, EC2_DELTRIG_AU.SQL, EC2_IN_TAB.SQL, EC2_TAB_UP.SQL EC2_DEL_CL_FUT_RECS.SQ L		x	
Financial - process totals and store in F_CASELOAD table. Totals are based on priority, language, caseload type, ethnic group, and poverty level	EC2_CL_REC_PROCESS.SQL		x	
Update caseload assignment information from the clinics	EC2_SYNC_CLINIC_ASSIGN MENT.SQL		x	
Consolidate and update all food instrument data to calculate FI obligation value	EC2_FOOD_UP.SQL, EC2_FI_UPD.SQL		x	
Search for potential dual enrollment participants	EC2_DUAL_ENROLL.SQL		x	
Compile new issuance and void information to send to the bank	EC2_ISSUE_FSMC.SQL		x	x
Create consolidated record of all FI issuance and redemption information and create vendor payment	EC4_BANK.SQL, EC4_BNK_POST.SQL		x	

records				
Create date of current food funds available from all sources as well as current food obligations and expenditures to establish current cash position	EC4_CASHFLOW_UPD.SQL		x	
Financial (end of month) - Populate F_INCOME to POVERTY table. These values are used in populating the caseload table	EC4_MON_END.SQL		x	
Vendor (end of month) calculate and update peer group averages	EC4_VEN_PEER_FI.SQL EC4_VEN_LOC_AGCY.SQL		x	
Vendor run and print analysis factor reports	EC4_VEN_PRICE.SQL EC4_VEN_RISK_HISTORY.S QL		x	
Generate dual enrollment report to be sent to agencies	EC4_DUAL_RPT.BAT EC4_DUAL.RPT		x	
Gather new/updated data to be sent to the agencies	EC4_TRNC.SQL EC4_TAB_DOWN.SQL		x	
Prepare archival retrieval data	EC1_PRG_ARCHV.SQL, EC4_RETRIEVE.SQL, EC4_INS_RETR_CLI.SQL EC4_INS_RETR_VEN.SQL		x	
Print out status logs	EC_PRINT.BAT		x	
Consolidate information from central	EA3_TRNC.SQL, EA3_PRE.SQL, EA3_DELTRIG.SQL, EA3_IN_TAB.SQL, EA3_TAB_UP.SQL		x	
*Clinic serial number replenishing	EC4_CL_SER_REP.SQL		x	
Update caseload assignment information from the clinics	EA3_SYNC_LA_ASSIGNME NT.SQL	x		
Send redemption/rejection information to agencies	EA3_PROC_FI.SQL	x		
Transmit archival parts. and vendors and update records at agencies	EA3_PRG_ARCHV.SQL, EA3_RETRIEVE.SQL EA3_INS_RETR_CLI.SQL EA3_INS_RETR_VEN.SQL	x	x	
*Vendor monthly reports			x	
*Vendor quarterly reports			x	
Print out status logs of the process	EA_PRINT.BAT	x		

\*New End of Day processes

**Oracle notation for schema and table names:**

A table name preceded by “AIM.” indicates that it is in the schema that holds permanent tables. The schema name is the same for all agencies and the central server.

A table name preceded by “EODADM.” indicates that it is in the schema that holds temporary tables, starting with A\_, E\_, and R\_, as well as the DEL\_STATEMENTS and EOD\_CONTROLS tables.

The tables whose names begin with **A\_** (not the appointment scheduler tables) are those that are populated at the Central Server, to be sent to the Agency Servers.

The tables whose names begin with **E\_** are those that are populated at the Agency Servers to be sent to the Central Server.

The tables whose names begin with **R\_** are used for Participant data for newly-archived participants. The **DEL\_STATEMENTS** table contains the actual SQL delete statements that are issued at the Central Database, which must be executed again at the Agencies, and vice versa.

The **EOD\_CONTROLS** table contain information about the dates that the EOD runs have taken place: to\_date (the date/time that the EOD process started), from\_date (the date/time that the EOD process ended), good\_proc\_date (the last date on which a successful EOD run was performed).

A processing description with the **Count (\*)** command in it will provide a count of the number of records meeting a select statement’s WHERE clause.

In the Selection and Processing sections in the EOD portion of the DTSD, occasionally a table “**alias**” is used. This is a short name that is defined so that it can be used instead of the long table name. In the following example, the alias for the I\_BANK\_REPORTS table is **IBR**:

```
select ibr.serial_number,
       ifi.ifit_fi_type_code
from   i_bank_reports ibr,
       i_food_instruments ifi
where  ibr.serial_number = ifi.serial_number
```

## 1.1 Perform back-up

### *Overview*

The system performs hot and cold backups of the application and database from Tuesday until Saturday at 2:30am and 3:00am respectively.

### **Processes:**

#### 1.1.1 ShutaimXX.bat

The system initiates the following NT commands on each Local Agency server to perform the hot backup and shutdown of the database.

```
Exp system/manager parfile=d:\oracle\backup\expaim01.par  
C:\arizona\817\bin\sqlplus internal/oracle @d:\oracle\backup\aim_shut.sql
```

#### 1.1.2 AimXXX\_backup.bat

The system initiates the following NT commands on each Local Agency server to perform a cold backup. The Cold backup includes incremental as well as a full backup.

```
C:\winnt\system32\ntbackup backup c:\arizona d:\oracle c:\orant d:\arizona /b /d "Complete backup of all oracle directories" /e /hc:on /l "d:\aim\eod\agency\logs\full_bkp.log" /t normal
```

```
C:\winnt\sysem32\ntbackup backup c:\ d:\ /a /d "Incremental backup of all drives" /hc:on /t incremental /l "d:\aim\eod\agency\logs\inc_bkp.log"
```

## 1.2 EA1.BAT

### Overview

This batch file begins the End of Day processing at the Agency level.

### Processing

On every agency server this script is run as part of the end of day process.

This script is also called from the Systems Administration End of Day form when the proceed button is pressed if the AGCY\_SEQ is not equal to '00'.

The system sets the environmental variables for the particular Local Agency reading them from the EA1.INI file and places them in a temporary file.

The following SQL scripts are then executed:

EA1.SQL which updates the EOD\_CONTROLS.TO\_DATE to sysdate and sets the environmental variables. It then runs the following EA1 scripts:

- EA1\_TERM\_EOD.SQL
- EA1\_CAT\_EOD.SQL
- EA1\_CAT\_SYNC\_EOD.SQL
- EA1\_APPT\_EOD.SQL
- EA1\_NEW\_CLASS\_EOD.SQL
- EA1\_NEW\_APPT\_EOD.SQL
- EA1\_PURGE\_OLD\_APPT.SQL
- EA1\_PURGE\_NEW\_APPT.SQL
- EA1\_OTHER\_PURG.SQL
- EA1\_TERM\_WL\_CLIENTS.SQL
- EA1\_FOOD\_CHNG.SQL
- EA1\_IEN\_ACTIVE.SQL
- EA1\_TRANSFER.SQL
- EA1\_STATE\_CLIENTS\_UPD.SQL
- EA1\_CSF\_OU\_FIX.SQL

EA1\_TAB\_UP.SQL which extracts client information that has been changed or created and exports these tables.

The tables are then dumped into a .dmp file with format % . dmp where the wildcard represents the Local Agency ID, and a completion flag is created used to indicate the completion of the copying process with format % .don where the wildcard represents the Agency ID.

The script then transfers the dump files to central ftp\_server using FTP protocol.

The script finishes by updating the EOD controls by first searching for the completion flag and then running EA2.SQL which updates the EOD\_CONTROLS FROM\_DATE and GOOD\_PROC\_DATE to the sysdate.

### 1.2.1 EA1\_SOFTWARE\_UPD.BAT

This script updates new software at agencies from the Central Server.

### 1.2.2 EA1.SQL

#### Overview

Executes SQL scripts for EA1.BAT.

#### 1.2.2.1 Inputs

eod\_controls  
env\_variables

#### 1.2.2.2 Selection

env\_variables  
where code = 'EOD\_BY'

#### 1.2.2.3 Processing

update eod\_controls, setting TO\_DATE to the system date  
update env\_variables, setting env\_value to 'EA1' where code = 'EOD\_BY'

```
run SQL script EA1_TERM_EOD
run SQL script EA1_CAT_EOD
run SQL script EA1_CAT_SYNC_EOD
run SQL script EA1_TRANSFER
run SQL script EA1_APPT_EOD
run SQL script EA1_NEW_CLASS_EOD
run SQL script EA1_NEW_APPT_EOD.SQL
run SQL script EA1_PURGE_OLD_APPT.SQL
run SQL script EA1_PURGE_NEW_APPT.SQL
run SQL script EA1_OTHER_PURG
run SQL script EA1_TERM_WL_CLIENTS
run SQL script EA1_FOOD_CHNG
run SQL script EA1_IEN_ACTIVE
run SQL script EA1_TRANSFER.SQL
run SQL script EA1_STATE_CLIENTS_UPD.SQL
run SQL script EA1_CSF_OU_FIX.SQL
```

#### 1.2.2.4 Outputs

eod\_controls  
env\_variables

Log file: EA1\_SQL.LOG

### 1.3 Automatic termination of clients no longer eligible

#### 1.3.1 EA1\_TERM\_EOD.SQL

During the End of Day process automatic terminations\* are performed as follows:

- \*• The WIC participant is postpartum and today's date is 6 months after the actual delivery date
- \*• The CSF participant is postpartum and today's date is the last day of the calendar month of the Actual Delivery date + 12 calendar months
- \*• The WIC participant is breastfeeding and today's date is 12 calendar months after the Actual Delivery date
- \*• The CSF participant is breastfeeding and today's date is the last day of the calendar month of the Actual Delivery date + 12 calendar months
- \*• The WIC participant is 5 years of age and today's date is the last day of the calendar month in which the participant turned 5 years of age
- \*• The CSF participant is 6 years of age and today's date is the last day of the calendar month in which the participant turned 6 years of age
- \*• The WIC or CSF participant's current certification period has expired and there is not a new certification period

Participants that have missed three consecutive appointments are no longer automatically terminated during the End of Day process.

\*For all Participant records terminated all future appointments, except for certification and re-certification appointments, for the participant are deleted during the End of Day process.

The following table outlines the term\_code's and descriptions:

Term_code	Description
A	NO NUTRITIONAL RISK FOUND
B	NOT INCOME ELIGIBLE
C	BREASTFEEDING WOMAN NO LONGER BREASTFEEDING
D	RISK FACTORS RESOLVED-GRADUATED- SEEN FOR RECERTIFICATION
E	VOLUNTARY WITHDRAWAL
F	DUPLICATE RECORD
G	MOVED OUT OF STATE
H	LOST TO FOLLOW-UP
I	DEATH
J	MOVED OUT OF CURRENT LOCAL AGENCY
K	MISSED 2 FOOD BOX PICK-UPS
L	XX-DO NOT USE
M	ERROR IN TERMINATION DATE
N	NOT PRIORITY NUTRITIONAL RISK-SEEN FOR RECERTIFICATION
O	NOT CURRENTLY SERVING PRIORITY IDENTIFIED/REFUSED WAIT LIST
P	ABUSE OR THREAT TO ANYONE CONNECTED TO WIC/CSF PROGRAM
Q	MISUSE OF FOOD INSTRUMENTS
R	SALE OR EXCHANGE OF FOOD OR FOOD INSTRUMENTS
S	INTENTIONAL MISREPRESENT OR WITHHOLD FACTS TO OBTAIN BENEFIT
T	RECEIPT OF CASH OR CREDIT TO PURCHASE UNAUTHORIZED FOODS
U	NOT SEEN FOR RECERTIFICATION
V	NOT RECERTIFIED. AUTO TERM FOR INFANT, CHILD, OR WOMAN
W	WOMAN AT 6 WEEKS POSTPARTUM
X	PREGNANT WOMAN AT 3 MONTHS PAST EXPECTED DELIVERY DATE
Y	NON-BREASTFEEDING WOMAN AT 6 MONTHS PAST ACTUAL DELIVERY

Z	BREASTFEEDING WOMAN AT ONE YEAR PAST ACTUAL DELIVERY DATE
1	CHILD FIVE YEARS OLD
2	TRANSFERRED TO ANOTHER AZ WIC LOCAL AGENCY
3	MISSED 4 MONTHS OF FOOD PACKAGES
4	CHILD SIX YEARS OLD
5	CURRENTLY PARTICIPATING IN CSFP
6	CURRENTLY PARTICIPATING IN WIC
8	CATEGORICALLY INELIGIBLE
9	CATEGORY CHANGE
0	CERTIFICATION PERIOD ENDED

**1.3.1.1 Inputs**

eod\_controls  
 aas\_appt\_clients  
 aas\_appt\_items  
 aas\_appointments  
 aas\_class\_families  
 c\_cert\_term\_reasons  
 c\_certifications  
 c\_client\_groups  
 c\_client\_ne\_topics  
 c\_client\_services  
 c\_client\_svc\_ne\_materials  
 c\_clients  
 c\_family\_economic\_units  
 f\_wait\_lists

**1.3.1.2 Selection**

Select c\_certification records where termination\_date is null and check\_out\_flag = 'N'

**1.3.1.3 Processing**

Main Routine

For each C\_CERTIFICATIONS record

```

cat_code   := C_CERTIFICATIONS.CAT_CATEGORY_CODE
c_e_date   := C_CERTIFICATIONS.CERT_END_DATE
r_stat     := C_CLIENTS.REC_STATUS
term_code := null;
  
```

- Determine the Termination Reason

```

if cat_code = 'P' (postpartum) and C_CERTIFICATIONS.ACTUAL_DELIVERY_DATE is more
  than six months ago (WIC) or (C_CERTIFICATIONS.ACTUAL_DELIVERY_DATE is twelve
  months ago(CSF) and today is the last day of the month) then
  term_code := 'Y'
  
```

```

if cat_code in ('EN', 'PN') (nursing woman) and
  C_CERTIFICATIONS.ACTUAL_DELIVERY_DATE is more than 12 months ago and today is
  the last day of the month for CSF participants then
  term_code := 'Z'
  
```

```

if term_code is null and cat_code in ('IEN', 'IFF', 'IPN', 'C1', 'C2', 'C3', 'C4') (child)
  
```

(The child over 5 years should take place on the last day of the month following the end of the birthdate month.)

```
wic_max_date := LAST_DAY(ADD_MONTHS(C_CLIENTS.BIRTH_DATE,60))
```

```
if (C_CERTIFICATIONS.PROGRAM = 'WIC' and wic_max_date is today or earlier)
  term_code := '1'
```

```
if term_code is null and cat_code in ('C5')
```

(The child over 6 years should take place on the the last day of the month following the end of the birthdate month.)

```
csf.max_date:= Last_day (add_months (C_clients.birth_date,72))
```

```
if (C_certifications.program= 'CSF' & csf.max_date is today or earlier)
```

```
  term_code = '4'
```

```
if c_e_date is today or earlier (the certification end date is today or earlier)
```

```
  term_code := 'V'
```

```
if cat_code in ('P', 'EN', 'PN') and program is CSF:
```

```
  csf.max_date:= last_day (add.months (actual_delivery _date,12))
```

```
if csf.max_date is today or earlier
```

```
  term code = '8'
```

```
-- Terminate the certification period, and if the client does not have a future
```

```
-- certification period then terminate the client
```

```
If term_code is not null
```

```
  If C_CERTIFICATIONS.CERT_START_DATE > today and TERMINATION_DATE is null
    Active_client := 'Y' (The client has a future certification period)
```

```
  Else
```

```
    update C_CLIENTS set TR_TERMINATION_CODE = term_code, REC_STATUS = 'I',
```

```
      PREV_REC_STATUS = r_stat
```

```
    Active_client := 'N'
```

```
  End if;
```

```
  update c_certifications set termination_date = current date, wait_list_flag = 'N'
```

```
  update f_wait_lists set active_flag = 'N'
```

```
  insert term_code into c_cert_term_reasons
```

```
- Remove all future appointments except Certification and Re-Certification appointments
```

```
  If active_client = 'N'
```

```
    For each future AAS_APPOINTMENTS record that is not for a Certification or Re-
      Certification Service loop
```

```
      Delete the A_AAS_APPT_ITEMS records for the AAS_APPOINTMENTS record
```

```
      Delete AAS_APPT_CLIENTS for the AAS_APPOINTMENTS record.
```

```
Update the AAS_APPOINTMENTS record set CFEU_FAMILY_ID to null and
AAS_AS_STATUS_CODE = 'P'
Delete AAS_CLASS_FAMILIES
```

```
For each C_CLIENT_SERVICES record where
  APPT_FLAG = 'Y' and
  SERVICE_DATE > Today's date and
  SRV_SERVICE_CODE not in ('CERT','RECERT') loop
```

```
  Delete the C_CLIENT_SVC_NE_MATERIALS records for each of the
  C_CLIENT_NE_TOPICS records for the C_CLIENT_SERVICES record
  Delete the C_CLIENT_NE_TOPICS records for the C_CLIENT_SERVICES record
  Delete the C_CLIENT_SERVICES record
```

```
  End loop
End loop
```

```
-- Remove the Client from Class appointments,
```

```
  For each AAS_APPOINTMENTS record where FAMILY_ID = the family id loop
```

```
  For each AAS-CLASS_FAMILIES record where CFEU_FAMILY_ID = the family id
    Delete AAS_CLASS_FAMILIES record
    Update AAS_APPOINTMENTS to set appointment status as 'Pending'
```

```
  End loop
End loop
```

```
  Insert into ENV_VARIABLES (code, env_value, date_created, created_by, date_modified,
  modified_by, note)
  values (C_CLIENTS.CLIENT_ID, term_code, sysdate, USERID, null, null,'termscript');
```

```
  end if active_client = 'N'
```

```
End loop
```

```
End
```

#### 1.3.1.4 Outputs

```
aas_appointment
aas_appt_items
aas_appt_clients
aas_class_families
c_certifications
c_client_groups
```

c\_client\_ne\_topics  
c\_client\_services  
c\_client\_svc\_ne\_materials  
c\_clients  
f\_wait\_lists  
c\_cert\_term\_reasons

### 1.3.2 EA1\_TERM\_WL\_CLIENTS.SQL

#### *Overview*

This script updates client records to indicate that they are terminated if the clients are on the waiting list for more than the allotted time requirements defined in the F\_CONTROLS table for the waiting list. (This does not apply to clients classified in check out clinic status.)

#### 1.3.2.1 Inputs

eod\_controls  
aas\_appointment  
aas\_appt\_items  
aas\_appt\_clients  
aas\_class\_families  
a\_appointments  
a\_appt\_appt\_items  
a\_appt\_topic\_materials  
a\_appt\_topics  
a\_clinic\_days  
a\_groups  
c\_cert\_term\_reasons  
c\_certifications  
c\_client\_groups  
c\_client\_ne\_topics  
c\_client\_services  
c\_client\_svc\_ne\_materials  
c\_clients  
c\_family\_economic\_units  
f\_wait\_lists

#### 1.3.2.2 Selection

Select client ID (ccl\_cc\_client\_id) from f\_wait\_lists where date added to wait list (date\_on\_wait\_list) <= (effective date - F\_CONTROLS.WAIT\_LIST\_PERIOD)

#### 1.3.2.3 Processing

Main Routine

For each F\_WAIT\_LISTS record

```
r_stat      := C_CLIENTS.REC_STATUS
term_code  := 'V'
```

```
-- Terminate the certification period, and if the client does not have a future
-- certification period then terminate the client
```

If term\_code is not null

```
If C_CERTIFICATIONS.CERT_START_DATE > today and TERMINATION_DATE is null
  Active_client := 'Y' (The client has a future certification period)
```

Else

```
  update C_CLIENTS set TR_TERMINATION_CODE = term_code, REC_STATUS = 'I',
    PREV_REC_STATUS = r_stat
  Active_client := 'N'
```

End if;

```
update c_certifications set termination_date = current date, wait_list_flag = 'N'
update f_wait_lists set active_flag = 'N'
insert term_code into c_cert_term_reasons
```

- Remove all future appointments except Certification and Re-Certification appointments

If active\_client = 'N'

For each future A\_APPOINTMENTS record that is not for a Certification or Re-Certification Service loop

```
Delete the A_APPT_APPT_ITEMS records for the A_APPOINTMENTS record
Delete the A_APPT_TOPIC_MATERIALS records for each of the A_APPT_TOPICS
  for the A_APPOINTMENTS record
Delete the A_APPT_TOPICS records for the A_APPOINTMENTS record
update the A_APPOINTMENTS record set CCS_CC_CLIENT_ID to null and
  ATS_ATTEND_STATUS_CODE = 'P' and GRO_GROUP_ID = null and
  WE_WORK_EVENT_ID = null and CCS_ID = null
```

For each future AAS\_APPOINTMENTS record

```
Delete corresponding AAS_APPT_CLIENTS
Delete AAS_APPT_ITEMS
Mark AAS_APPOINTMENTS as Pending if this client is the only member for that
family appointment.
```

For each C\_CLIENT\_SERVICES record where

```
  APPT_FLAG = 'Y' and
  SERVICE_DATE > Today's date and
  SRV_SERVICE_CODE not in ('CERT','RECERT') loop
```

```
Delete the C_CLIENT_SVC_NE_MATERIALS records for each of the
  C_CLIENT_NE_TOPICS records for the C_CLIENT_SERVICES record
```

```

Delete the C_CLIENT_NE_TOPICS records for the C_CLIENT_SERVICES record
Delete the C_CLIENT_SERVICES record

```

```

End loop
End loop

```

```

-- Remove the Client from group appointments, and remove the group if there are no more -- clients in the
group

```

```

For each C_CLIENT_GROUPS record where CC_CLIENT_ID = the client id loop
  If there is a future A_APPOINTMENTS record where GRO_GROUP_ID =
  C_CLIENT_GROUPS.GROUP_ID
    Delete the C_CLIENT_GROUPS record

    If there are no more clients in the C_CLIENT_GROUPS table for this group

      For each future A_APPOINTMENTS record where GRO_GROUP_ID =
      C_CLIENT_GROUPS.GROUP_ID loop
        Delete the A_APPT_APPT_ITEMS records for the A_APPOINTMENTS
        record
        Delete the A_APPT_TOPIC_MATERIALS records for each of the
        A_APPT_TOPICS for the A_APPOINTMENTS record
        Delete the A_APPT_TOPICS records for the A_APPOINTMENTS record
        update the A_APPOINTMENTS record set CCS_CC_CLIENT_ID to null
        and ATS_ATTEND_STATUS_CODE = 'P' and GRO_GROUP_ID =
        null and WE_WORK_EVENT_ID = null and CCS_ID = null
        Delete A_GROUPS record
      End loop
    End loop
  End loop

```

```

End loop
End loop

```

```

Insert into ENV_VARIABLES (code, env_value, date_created, created_by, date_modified,
modified_by, note)
values (C_CLIENTS.CLIENT_ID, term_code, sysdate, USERID, null, null,'termscript');

```

```

end if active_client = 'N'

```

```

End loop

```

```

End

```

#### 1.3.2.4 Outputs

```

aas_appointments
aas_appt_items
aas_appt_clients
a_appointments

```

a\_appt\_appt\_items  
a\_appt\_topic\_materials  
a\_appt\_topics  
a\_groups  
c\_cert\_term\_reasons  
c\_certifications  
c\_client\_groups  
c\_client\_ne\_topics  
c\_client\_services  
c\_client\_svc\_ne\_materials  
c\_clients  
f\_wait\_lists

## 1.4 Perform automatic category changes

### 1.4.1 EA1\_CAT\_EOD.SQL

#### *Overview*

This script performs automatic category changes for clients based upon their certification category codes, current date and birth date. The following table outlines updates that take place on the client certification record. (Does not apply to clients classified in check out clinic status, i.e. o\_organizational\_units.check\_out\_flag = 'N' ):

<b>Old code</b>	<b>Months between current month and birth date</b>	<b>New code</b>
IEN, IPN, IFF	12	C1
C1	24	C2
C2	36	C3
C3	48	C4
C4	60	C5
PG1	216	PG2

#### 1.4.1.1 Inputs

c\_cat\_goals  
 c\_cat\_referrals  
 c\_cat\_nutr\_eds  
 c\_certifications  
 c\_client\_goals  
 c\_client\_ne\_topics  
 c\_client\_services  
 c\_clients  
 c\_family\_economic\_units  
 c\_health\_risk\_factors  
 o\_organizational\_units  
 c\_cert\_term\_reasons

### 1.4.1.2 Selection

Select all clients where category code in ('IEN', 'IPN', 'IFF', 'C1', 'C2', 'C3', 'C4', 'PG1') and the clinic the client is assigned to is not checked out and the current date is between the start and end certification dates, and the termination date is null. (This picks up the client's current certification record.)

### 1.4.1.3 Processing

For the C\_CLIENTS / C\_CERTIFICATIONS records:

If C\_CERTIFICATIONS.PROGRAM = 'WIC'

```
cmp_date := MONTHS_BETWEEN(SYSDATE, C_CLIENTS.BIRTH_DATE)
```

```
if category code starts with 'I' and cmp_date >=12
```

```
  new_cat_code := 'C1'
```

```
  Run procedure update_cert
```

```
elseif category code = 'C1' and cmp_date >=24
```

```
  new_cat_code := 'C2'
```

```
  Run procedure update_cert
```

```
elseif category code = 'C2' and cmp_date >= 36
```

```
  new_cat_code := 'C3'
```

```
  Run procedure update_cert
```

```
elseif category code = 'C3' and cmp_date >= 48
```

```
  new_cat_code := 'C4'
```

```
  Run procedure update_cert
```

```
elseif category code = 'PG1' and cmp_date >= 216
```

```
  new_cat_code := 'PG2'
```

```
  Run procedure update_cert
```

```
end if
```

```
end if
```

If C\_CERTIFICATIONS.PROGRAM = 'CSF'

```
cmp_date := MONTHS_BETWEEN(SYSDATE, LAST_DAY(C_CLIENTS.BIRTH_DATE))
```

```
if category code starts with 'I' and cmp_date >=12
```

```
  new_cat_code := 'C1'
```

```
  Run procedure update_cert
```

```
elseif category code = 'C1' and cmp_date >=24
```

```
  new_cat_code := 'C2'
```

```
  Run procedure update_cert
```

```
elseif category code = 'C2' and cmp_date >= 36
```

```
  new_cat_code := 'C3'
```

```
  Run procedure update_cert
```

```
elseif category code = 'C3' and cmp_date >= 48
```

```
  new_cat_code := 'C4'
```

```

Run procedure update_cert
elseif category code = 'C4' and cmp_date >= 60
    new_cat_code := 'C5'
    Run procedure update_cert
elseif category code = 'PG1' and cmp_date >= 216
    new_cat_code := 'PG2'
    Run procedure update_cert
end if
end if
end if

```

Procedure **update\_cert**:

*For infants and children*

Update c\_certifications set cert\_end\_date = ( birthday-1 ) , termination date = birthday

*For pregnant women*

Update c\_certifications set cert\_end\_date = ( birthday - 1), termination date = birthday-1,

Update c\_food\_package\_prescriptions set end\_date = (birthday-1)

If new\_cat\_code = 'C1' and priority = 2, then new priority := 3

Set start/end dates for calculation of duration of certification:

calc\_start\_date := current\_date + 1 day

calc\_end\_date := certification end date (c\_end\_date)

certification duration := (end date - start date + 1 day) / 7 days

set certification duration to 1 if it is <= 0

Set the new certification end date from the current date plus the new cert period:

Current date + 1 day + (calc\_end\_date - calc\_start\_date)

Insert into c\_certifications using the calc\_start\_date, calc\_end\_date, priority code, new\_cat\_code, certification duration.

Insert into c\_cert\_term\_reasons for the old certification record, set tr\_termination\_code = 'Q'

Update c\_food\_package\_prescriptions set cc1\_cert\_start\_date := calc\_start\_date for all food package prescriptions whose effective\_date is after the current date.

Select c\_cat\_referrals (category referrals) for the client's category code

Duplicate the c\_cat\_referrals for the client's category into c\_client\_referrals by inserting into c\_client\_referrals with c\_cat\_referrals.prg\_program\_id and c\_cat\_referrals.to\_from\_flag

Select crf\_risk\_factor\_id, crf\_cat\_category\_code, cc1\_cc\_client\_id, cc1\_cert\_start\_date  
From c\_health\_risk\_factors for the new certification period.

Insert duplicate records into c\_health\_risk\_factors until no more records are found.

Select c\_cat\_goals (category goals) for the client's category code

If the client has no goals for that category, duplicate the c\_cat\_goals for the client's category into c\_client\_goals with c\_cat\_goals.g1\_goal\_code and standard\_flag = 'Y'

Select c\_client\_services for the client with appt\_flag = 'N'

If no records were found, then

Find next available id value from c\_client\_services: next\_id

Insert into c\_client\_services setting key\_id = next\_id, ats\_attend\_status\_code = 'P', service\_date = current date, appt\_flag = 'N', cat\_category\_code to the client's category

Else

Delete c\_client\_ne\_topics for the client and most recent and c\_client\_services.id

End if

Get all net\_nutr\_ed\_topic\_code records for the client's category

Check if the nutrition education topics already exist for the client

If not, insert into c\_client\_ne\_topics, setting standard\_flag = 'Y', received\_flag = 'N'

Else, update the c\_client\_ne\_topics set standard\_flag = 'Y'

End Procedure **update\_cert**:

#### 1.4.1.4 Outputs

c\_certifications  
 c\_client\_goals  
 c\_client\_ne\_topics  
 c\_client\_referrals  
 c\_client\_services  
 c\_cert\_term\_reasons  
 c\_food\_package\_prescriptions  
 c\_health\_risk\_factors

#### 1.4.2 EA1\_CAT\_SYNC\_EOD.SQL

##### *Overview*

Client record's category code is kept in sync with the certification record's category code. In the case of a future certification, the c\_clients.cat\_category\_code maintains the current certification category until the current cert period ends. When the current cert period ends, this eod process will change the c\_clients.cat\_category\_code to equal the new c\_certifications.cat\_category\_code.

##### 1.4.2.1 Inputs

c\_clients  
 c\_certifications

#### 1.4.2.2 Selection

Select all client records where category code does not equal the category code of the current untermiated cert record

#### 1.4.2.3 Processing

Main routine

For all client records with a category mismatch, update the c\_clients.cat\_category\_code to the current c\_certifications.cat\_category\_code.

```
update c_clients
set cat_category_code = cat_code,
  modified_by = user,
  date_modified = sysdate
where client_id = cli_id;
```

#### 1.4.2.4 Outputs

c\_clients

## 1.5 Mark appointments as Missed or Kept

### 1.5.1 EA1\_APPT\_EOD.SQL

#### *Overview*

Client appointment status is updated from pending (P) to kept (K) for client's who's appointment attend status code is 'P' and the calendar date is today or earlier, depending on the following conditions (does not apply to clients classified in check out clinic status):

- \* If food instruments have been issued for the appointment month
- \* AND the disposition of any food instrument created on the date of the appointment
- \* AND/OR medical or bloodwork records create date is greater than or equal to the appointment date

Client appointment status is updated from pending (P) to missed (M) if the above conditions are not met.

If the participant is a mother of an exclusively breastfeeding infant, and she meets the criterion in 1-3 to have her appointment updated to kept (K) then the infant's appointment record will be updated to kept (K) also.

#### 1.5.1.1 Inputs

a\_appointments  
c\_bloodwork\_data  
c\_client\_groups  
c\_infant\_child\_medicals  
c\_woman\_medicals  
i\_food\_instruments

#### 1.5.1.2 Selection

Client appointments:

Select all a\_appointments records where ats\_attend\_status\_code = 'P' and cd\_calendar\_date is today or earlier.

Group appointments:

Select all a\_appointments records where ats\_attend\_status\_code = 'P' and cd\_calendar\_date is today or earlier.

**1.5.1.3 Processing**

## Main routine

1. for all pending client appointments today or earlier, if attend\_code = 'P',  
Run procedure **check\_for\_x**:

Search for all food instruments that has issue\_date in the current month, and  
 idis\_disposition\_code = '2'; if found, set found\_it := 'Y'  
 Search for c\_bloodwork\_data that has the date\_created equal to the appointment  
 calendar date; if found, set found\_it := 'Y'  
 Search for c\_infant\_child\_medicals that has date\_created equal to the  
 appointment calendar date; if found, set found\_it := 'Y'  
 Search for c\_woman\_medicals that has date\_created equal to the appointment  
 calendar date; if found, set found\_it := 'Y'

If found\_it = 'N',  
     ats\_code := 'M'  
 Else  
     ats\_code := 'K'

Run procedure **update\_appointments**:

Update a\_appointments set ats\_attend\_status\_code := ats\_code

Run procedure **update\_client\_services**:

Update c\_client\_services set ats\_atten\_status\_code := ats\_code for all pending  
client appointments today or earlier, if attend\_code = 'P',

2. For all pending group appointments today or earlier if attend\_code = 'P' for clients in the  
 group, set client services and groups to kept/missed.

For all clients in the group,

Run procedure **check\_for\_x**:

Search for all food instruments that has issue\_date in the current month,  
 and idis\_disposition\_code = '2'; if found, set found\_it := 'Y'  
 Search for c\_bloodwork\_data that has the date\_created equal to the  
 appointment calendar date; if found, set found\_it := 'Y'  
 Search for c\_infant\_child\_medicals that has date\_created equal to the  
 appointment calendar date; if found, set found\_it := 'Y'  
 Search for c\_woman\_medicals that has date\_created equal to the  
 appointment calendar date; if found, set found\_it := 'Y'

If found\_it = 'N',

```
        ats_code := 'M'  
Else  
        ats_code := 'K'
```

```
Update c_client_services set ats_atten_status_code := ats_code  
Update c_client_groups set ats_attend_status_code := ats_code
```

- 2a. And for the same group appointments: update appointments to missed if all clients in the group are missed.

```
Set temp_ats_code to 'A'  
if ats_attend_status_code = 'K' on the c_client_groups table for the group,  
    Set temp_ats_code to 'Z'
```

```
If temp_ats_code = 'Z',  
    Update a_appointments (the group appointment) set ats_attend_status_code = 'K'  
Else  
    Update a_appointments (the group appointment) set ats_attend_status_code =  
        'M'
```

If the previous processing updates the participants attend\_status\_code to 'K' and the participant is the mother of an exclusively breastfeeding infant, c\_clients.cat\_category\_code = 'IEN' and c\_clients.cc\_client\_id = c\_clients.client\_id (the mother's client id) and the mother's a\_appointments.acd\_calendar\_day and a\_appointments.start\_time equal the infant's a\_appointments.acd\_calendar\_day and a\_appointments.start\_time the infant's appointment record will be updated to kept (K) also.

#### 1.5.1.4 Outputs

a\_appoinments  
c\_client\_services  
c\_client\_groups

## 1.6 Delete information no longer used by the system

### 1.6.1 EA1\_OTHER\_PURG.SQL

#### *Overview*

This script deletes records that are no longer used by the system. The following outlines records that are deleted and any conditions relating to the deletion of these records (does not apply to clients classified in check out clinic status):

- \* Client records will be deleted, as well as their child records, if the client has no certification record or the client record has no appointment date and have not been waitlisted .
- \* Client records will be deleted, as well as their child records, with no application date and date created/date modified more than 60 days ago who have not been waitlisted
- \* The family records will be deleted, as well as the child records, if the family records have no related client records.
- \* The food instrument records will be deleted if the issue date is greater than twenty four months ago.
- \* The del statements records whose date created is earlier than the first day of two months ago.
- \* Transfer requests that are more than 1 month old.
- \* Income history records that do not have any dependent income records.
- \* Leftover records in the temporary food instrument table.
- \* Archived client records that were created more than 15 days in the past.

#### 1.6.1.1 Inputs

a\_appointments  
a\_appt\_appt\_items  
a\_appt\_topic\_materials  
a\_appt\_topics  
c\_b\_n\_hh\_reasons\_bfends  
c\_bloodwork\_data  
c\_cert\_peer\_counsels  
c\_cert\_term\_reasons  
c\_certifications  
c\_client\_communications  
c\_client\_goals  
c\_client\_groups  
c\_client\_ne\_topics  
c\_client\_notes  
c\_client\_progs  
c\_client\_referrals  
c\_client\_services  
c\_client\_svc\_ne\_materials  
c\_clients

c\_contacts  
c\_dietary\_assessments  
c\_economic\_unit\_members  
c\_family\_economic\_units  
c\_family\_economic\_units cfeu,  
c\_family\_phones  
c\_feu\_communications  
c\_food\_package\_prescriptions  
c\_health\_risk\_factors  
c\_i\_c\_healths  
c\_i\_c\_hh\_reasons\_bfends  
c\_income\_histories  
c\_incomes  
c\_infant\_child\_medicals  
c\_infant\_data  
c\_infant\_data  
c\_p\_couns\_notes  
c\_peer\_rbfends  
c\_resolutions  
c\_transfer\_comms  
c\_transfer\_histories  
c\_transfer\_phones  
c\_transfers\_info  
c\_woman\_medicals  
del\_statements  
f\_wait\_list\_contacts  
f\_wait\_lists  
i\_bf\_promo\_vchrs  
i\_fi\_reject\_reasons  
i\_food\_instruments  
i\_temp\_fis  
o\_organizational\_units  
r\_archived\_clients  
v\_compliance\_case\_clients

### 1.6.1.2 Selection

1. clients without a certification record, created more than 120 days ago
2. clients with no application date and date created/modified more than 120 days ago
3. c\_family\_economic\_units with no client records
4. i\_food\_instruments records whose issue date is more than 24 months ago
5. c\_transfers\_info records with date\_created more than ten days 1 month in the past
6. del\_statements records whose date\_created is earlier than the first day of two months ago

7. c\_income\_histories records without a dependent c\_income record
8. Any i\_temp\_fis records remaining
9. r\_archived\_clients records with date\_created more than 15 days in the past.

### 1.6.1.3 Processing

1. For all clients without a certification record, created more than 120 days ago, and not checked out:

Run procedure **delete\_client\_children**:

Procedure **delete\_client\_children**

Clid := C\_CLIENTS.CLIENT\_ID

Delete the following records for the client:

For all appointments for the client, delete:

a\_appt\_topic\_materials

a\_appt\_topics

a\_appt\_appt\_items

Update a\_appointments set ats\_attend\_status\_code = 'P', css\_cc\_client\_id = null,  
css\_id = null

c\_client\_svc\_ne\_materials

c\_prev\_names

c\_more\_ethnic\_groups

c\_prev\_families

c\_bf\_promo\_issuances

cw\_reasons\_bfends

i\_food\_instruments

f\_wait\_list\_contacts

c\_diet\_nutrients

c\_i\_c\_hh\_reasons\_bfends

c\_client\_ne\_topics

c\_woman\_medicals

c\_bloodwork\_data

c\_cert\_term\_reasons

c\_b\_n\_healths

c\_food\_package\_prescriptions

f\_wait\_lists

c\_p\_healths

c\_infant\_child\_medicals

c\_dietary\_assessments

c\_health\_risk\_factors

c\_i\_c\_healths

c\_peer\_rbfends

c\_contacts  
 c\_p\_couns\_notes  
 c\_resolved\_clients  
 i\_bf\_promo\_vchrs  
 v\_compliance\_case\_clients  
 c\_client\_services  
 c\_certifications  
 c\_transfer\_histories  
 c\_immunizations  
 c\_client\_referrals  
 c\_client\_communications  
 c\_resolutions  
 c\_client\_notes  
 c\_client\_progs  
 c\_client\_groups  
 c\_infant\_data  
 c\_client\_goals  
 c\_cert\_peer\_counsels

Delete from C\_INFANT\_DATA where cc\_client\_id\_mother = clid;

Update c\_clients set cc\_client\_id = null

Loop through all c\_income\_histories records for the client and  
 Delete all related c\_incomes, c\_economic\_unit\_members,  
 c\_income\_histories

Delete from C\_CLIENTS where client\_id=clid;

End Procedure **delete\_client\_children**

2. For all clients with no application date and date created/modified more than 120 days ago  
Run procedure **delete\_client\_children**: (see above)
3. For all c\_family\_economic\_units with no client records:

Delete all c\_income records where family has a c\_income\_histories record.  
 Delete all c\_economic\_unit\_members where family has a c\_income\_histories record.  
 Delete the following records for the family:

c\_income\_histories  
 c\_feu\_communications  
 c\_family\_phones  
 c\_family\_economic\_units  
 c\_fam\_referrals  
 c\_smoking\_histories  
 c\_economic\_unit\_members  
 c\_incomes

4. For all i\_food\_instruments records whose issue date is more than 24 months ago,

Delete the following record:  
 i\_fi\_reject\_reasons

### i\_food\_instruments

5. For all c\_transfers\_info records with date\_created more than ten days 1 month in the past,

Delete the following records for the transfer:

- c\_transfer\_comms
- c\_transfer\_phones
- c\_transfers\_info

6. Delete all del\_statements whose date\_created is earlier than the first day of two months ago.
7. Delete all c\_income\_histories\_records that do not have dependent c\_income records. This is to purge extra income history records created by the income calculator.
8. Delete any records remaining in the i\_temp\_fis table. If food instrument printing was not successful during the business day, unwanted records may be left in this table. So they need to be purged.
9. Delete any r\_archived\_clients records with a date\_created more than 15 days in the past.

#### 1.6.1.4 Outputs

- a\_appointments
- a\_appt\_appt\_items
- a\_appt\_topic\_materials
- a\_appt\_topics
- c\_b\_n\_healths
- c\_b\_n\_hh\_reasons\_bfends
- c\_bloodwork\_data
- c\_cert\_peer\_counsels
- c\_cert\_term\_reasons
- c\_certifications
- c\_client\_communications
- c\_client\_goals
- c\_client\_groups
- c\_client\_ne\_topics
- c\_client\_notes
- c\_client\_progs
- c\_client\_referrals
- c\_client\_services
- c\_client\_svc\_ne\_materials
- c\_clients
- c\_contacts
- c\_diet\_nutrients
- c\_dietary\_assessments
- c\_economic\_unit\_members
- c\_family\_economic\_units
- c\_family\_economic\_units
- c\_family\_phones
- c\_feu\_communications

c\_food\_package\_prescriptions  
c\_health\_risk\_factors  
c\_i\_c\_healths  
c\_i\_c\_hh\_reasons\_bfends  
c\_immunizations  
c\_income\_histories  
c\_income\_histories  
c\_incomes  
c\_infant\_child\_medicals  
c\_infant\_data  
c\_p\_couns\_notes  
c\_p\_healths  
c\_peer\_rbfends  
c\_resolutions  
c\_transfer\_comms  
c\_transfer\_histories  
c\_transfer\_phones  
c\_transfers\_info  
c\_woman\_medicals  
del\_statements  
f\_wait\_list\_contacts  
f\_wait\_lists  
i\_bf\_promo\_vchrs  
i\_fi\_reject\_reasons  
i\_food\_instruments  
i\_temp\_fis  
o\_organizational\_units  
r\_archived\_clients  
v\_compliance\_case\_clients  
c\_prev\_names  
c\_prev\_families  
c\_more\_ethnic\_groups  
c\_bf\_promo\_issuances  
c\_smoking\_histories  
c\_economic\_unit\_members  
c\_incomes

## 1.7 Set exclusively breastfeeding clients to active status

### 1.7.1 EA1\_IEN\_ACTIVE.SQL

#### *Overview*

**Note: Currently, Arizona WIC is not using this form.**

Clients which are exclusively breastfeeding have their records updated to a status of active (A) from a record status of pending (P). The category code that makes a client exclusively breastfeeding is a category of 'IEN' (does not apply to inactive infants in check out clinic status).

#### 1.7.1.1 Inputs

c\_bloodwork\_data  
 c\_certifications  
 c\_clients  
 c\_family\_economic\_units  
 c\_food\_package\_prescriptions  
 c\_i\_c\_healths  
 c\_infant\_child\_medicals  
 eod\_controls  
 o\_organizational\_units

#### 1.7.1.2 Selection

For all c\_clients with cat\_category\_code = 'IEN' (breastfeeding infant) and rec\_status = 'P' (pending)

#### 1.7.1.3 Processing

For all c\_clients with cat\_category\_code = 'IEN' (breastfeeding infant) and rec\_status = 'P' (pending)

Get the current certification record: cert\_start\_date. For the infant's current certification:

Get c\_i\_c\_healths records: v\_found := count(\*)

If v\_found > 0 then

Get c\_infant\_child\_medicals record: v\_found := count(\*)

If v\_found > 0 and birth date is more than six months ago

Get c\_bloodwork\_data record: v\_found := count(\*)

If a current certification record found and v\_found > 0

Get c\_food\_package\_prescriptions for the infant's current certification: v\_found := count(\*)

If v\_found = 0 (no food package prescriptions were found for the infant's current certification period: Update c\_clients set rec\_status = 'A', date\_modified = eod date

**1.7.1.4** Outputs

c\_clients

**1.8** Process any pending transfer requests**1.8.1** EA1\_TRANSFER.SQL***Overview***

Any transfer requests that could not be completed during the day will be performed at this time.

**1.8.1.1** Inputs

none

**1.8.1.2** Selection

none

**1.8.1.3** Processing

Runs the transfer\_pkg.initiate\_transfer database procedure which loops through the pending transfer requests in c\_transfers\_info while processing each one.

**1.8.1.4** Outputs

Prints message (hold\_message) to agcy\_sql.log indicating transfer\_ids that were processed including the number of successful and unsuccessful transfers.

Ex.  
Transfer ID: 10013  
Transfers Successful: 1 Transfers Failed: 0

## 1.9 Generate notices, letters, and reports

### 1.9.1 CS\_LETTER\_OF\_TERM.FMX

#### *Overview*

This form prints out the ineligibility notices for participants who have been terminated by `ea1_term_eod` and `ea1_term_wl_clients`. This form also prints appointment notices 14 days before the participant's appointment. A notice of Ineligibility is printed six weeks prior to the end of the certification period for Participants that will no longer be categorically eligible for CSF. The system also prints the CSFP Certifications Due report and a Notice to Reapply for CSF Participants six weeks prior to the end of their certification period if they remain potentially categorically eligible. This form also prints the Pending Food Package Approval report listing all participants with food packages that require approval.

The system prints the reports and notices for participants at their assigned clinic. This will be facilitated through the identification of the clinic printers for each Local Agency. Each printer will have a unique "name" that the Local Agency/Clinic server will use to initiate print commands over the Arizona WIC WAN. As of this submission, the new Server/Printers have not been implemented, so a list of the Clinic printer names cannot be provided.

#### 1.9.1.1 Inputs

a\_attend\_status  
a\_appointments  
c\_clients  
c\_certifications  
c\_client\_services

#### 1.9.1.2 Selection

#### Ineligibility Notice

C\_CERTIFICATIONS where termination\_date is null and sysdate equals  $\max(c\_certifications.cert\_end\_date)$  minus six weeks and a C\_CLIENT\_COMMUNICATIONS record does not exist where (date\_sent\_called is  $\geq c\_certifications.cert\_end\_date - 6$  weeks and `ct_comm_type_code = 'IN'`) and any of the following conditions are true:

1. `c_certifications.cat_category_code` in 'PG1','PG2'(Pregnant) and the `cert_end_date` is 42 days after the last day of the month for the `c_certifications.expected_delivery_date`
2. `c_certifications.cat_category_code = 'C5'`(child) and the `cert_end_date` is the last day of the 72nd month after `c_clients.birth_date`
3. `c_certifications.cat_category_code` in 'P', 'EN', 'PN'(Postpartum or Breastfeeding) and the `cert_end_date` is the last day of the 12th month after `c_certifications.actual_delivery_date`.

### Notice to Reapply for CSF Participants

C\_CERTIFICATIONS where termination\_date is null and sysdate equals max(c\_certifications.cert\_end\_date) minus six weeks and a C\_CLIENT\_COMMUNICATIONS record does not exist where (date\_sent\_called is >= c\_certifications.cert\_end\_date - 6 weeks and ct\_comm\_type\_code = 'RN') and any of the following conditions are true:

1. c\_certifications.cat\_category\_code in 'E1', 'E2', 'E3', 'E4' (Elderly).
2. c\_certifications.cat\_category\_code = 'C5'(child) and the cert\_end\_date is less than the last day of the 72nd month after c\_clients.birth\_date
3. c\_certifications.cat\_category\_code in 'P', 'EN', 'PN'(Postpartum or Breastfeeding) and the cert\_end\_date is less than the last day of the 12th month after c\_certifications.actual\_delivery\_date.

### CSF Recerts Due Report

Same criteria as the Notice to Reapply selection.

### Appointment Notices

C\_CLIENTS where sysdate equals the a\_appointments.cd\_calendar\_date minus 14 days for the c\_client.

The O\_ORGANIZATIONAL\_UNITS.PRINT\_APPT\_NOTICES must be checked on the O\_ORGANIZATIONAL\_UNITS screen in Operations Management in order to print the appointment notices during End of Day.

### Pending Food Package Approval

C\_CLIENTS where C\_FOOD\_PACKAGE\_PRESCRIPTION.REQUIRES\_APPROVAL = 'Y' and C\_FOOD\_PACKAGE\_PRESCRIPTION.NUTRITIONIST\_FLAG and PHYSICIAN\_FLAG = 'N' and C\_CERTIFICATIONS.TERMINATION\_DATE is null.

#### 1.9.1.3 Processing

##### Ineligibility Notice

```
For each C_CERTIFICATIONS record found meeting the above criteria
Loop
  if C_FAMILY_ECONOMIC_UNITS.LANG_LANGUAGE_CODE = '2' (Spanish)
    Print the MS Word document cs_letter_of_inelig_sp.doc
  else
    Print the MS Word document cs_letter_of_inelig.doc.
  end if
End loop
```

##### Notice to Reapply for CSF Participants

```
For each C_CERTIFICATIONS record found meeting the above criteria
Loop
  if C_FAMILY_ECONOMIC_UNITS.LANG_LANGUAGE_CODE = '2' (Spanish)
    Print the MS Word document cs_csf_reapply_notice_sp.doc
  else
    Print the MS Word document cs_csf_reapply_notice.doc
  end if
End loop
```

##### CSFP Certifications Due Report

Invoke and print the Oracle Report file cr\_csf\_recerts\_due

### Appointment Notices

For each C\_CLIENTS record found meeting the above criteria  
Loop

Print the MS Word document appt\_notice.doc

End Loop

Pending Food Package Approval

Invoke and print the Oracle report file cr\_pend\_fp\_appr

#### **1.9.1.4** Outputs

Reference the following sections of the DTSD for layouts of the notices and reports:

Section 2 - 4.1.10	Notice to Reapply
Section 2 - 4.1.4	Notice of Ineligibility
Section 2- 4.3.37	CSFP Certifications Due
Section 1-1.1.1.10	Notice of Appointment.
Section 2- 4.3.45	Pending Food Package Approval Report

**1.10** Gather updated client information, new issuance information, void information, and requests for transfer

**1.10.1** EA1\_TAB\_UP.SQL

***Overview***

This script extracts client information that has been changed or created. It compares the change or created date in each of the following tables to the last day that End of Day Processes ran successfully.

- Staff members
- Staff phones
- Staff time studies
- Job descriptions
- Food instruments
- Food instrument types
- Food instrument foods
- Food instrument formulas
- Food packages
- Food package food instrument types
- Food package foods
- Food package prescriptions
- FI formulas
- Package distributions
- Organization units
- Organization programs
- Organization unit phones
- Outreach organizations
- Outreach comms
- Outreach organizations phones
- Outreach programs
- Outreach Org LAs
- Family economic units
- Family phones
- Family referrals
- FEU communications
- Clients
- Clients BP Issuances
- Client communications
- Client goals
- Client notes
- Client programs
- Client referrals
- Client services

- Client more ethnic groups
- Client NE topics
- Client services NE materials
- Client previous names
- Certified peer counsels
- Counsel notes
- Client allergy foods
- Peer Rbfends
- Contacts
- Income histories
- Economic unit members
- Incomes
- Immunizations
- Infant data
- Infant child medicals
- Resolutions
- Resolved clients
- Transfer information
- Transfer histories
- Certifications
- Certification term reasons
- Cert nutr eds
- Bloodwork data
- Dietary assessments
- Diet nutrients
- Health risk factors
- C Food Package Prescriptions
- C B W healths
- C B W HH reasons bfends
- C I C healths
- C I C HH reasons bfends
- C P healths
- C P S responses
- C food box distributions
- C infant child medicals
- C woman medicals
- F case assignments
- F caseload restrictions
- F wait lists
- F wait list contacts
- I BF promotional vouchers
- I BF voucher items
- I FI inventories
- I FI ranges
- I inventories
- I inventory organizational units
- I stock inventories
- I inv org units

- V complaints
- V monthly activities
- V activity findings
- Delete statements
- Batch runs
- Printer Addresses

### 1.10.1.1 Inputs

EOD\_CONTROLS  
O\_STAFF\_MEMBERS  
O\_STAFF\_TIMSTUDIES  
I\_FOOD\_INSTRUMENT\_TYPES  
O\_ORGANIZATIONAL\_UNITS  
I\_FOOD\_INSTRUMENT\_FOODS  
O\_ORG\_PROGRAMS  
I\_FOOD\_INSTRUMENTS  
O\_ORG\_UNIT\_PHONES  
I\_FI\_FORMULAS  
O\_OUTREACH\_ORGANIZATIONS  
O\_OUT\_ORG\_LAS  
I\_FOOD\_PACKAGFI\_TYPES  
O\_OUTREACH\_COMMS  
O\_OUTREACH\_ORG\_PHONES  
O\_OUTREACH\_PROGRAMS  
O\_STAFF\_PHONES  
O\_JOB\_DESCRIPTIONS  
C\_FAMILY\_ECONOMIC\_UNITS  
C\_FAMILY\_PHONES  
C\_FAM\_REFERRALS  
C\_FEU\_COMMUNICATIONS  
C\_SMOKING\_HISTORIES  
C\_PREV\_FAMILIES  
C\_CLIENTS  
C\_CERT\_PEER\_COUNSELS  
C\_CONTACTS  
C\_PEER\_RBFENDS  
C\_P\_COUNS\_NOTES  
C\_CLIENT\_COMMUNICATIONS  
C\_CLIENT\_GOALS  
C\_CLIENT\_NOTES  
C\_CLIENT\_PROGS  
C\_CLIENT\_REFERRALS  
C\_CLIENT\_SERVICES  
C\_CLIENT\_NTOPICS  
C\_CLIENT\_SVC\_NMATERIALS  
C\_MORETHNIC\_GROUPS  
C\_PREV\_NAMES  
C\_BF\_PROMO\_ISSUANCES

C\_ALLERGY\_FOODS  
C\_W\_HEALTHS  
C\_W\_HH\_REASONS\_BFENDS  
C\_PS\_RESPONSES  
C\_CLIENT\_BP\_ISSUANCES  
C\_INCOMHISTORIES  
C\_ECONOMIC\_UNIT\_MEMBERS  
C\_INCOMES  
C\_INFANT\_DATA  
C\_RESOLUTIONS  
C\_RESOLVED\_CLIENTS  
C\_TRANSFERS\_INFO  
C\_TRANSFER\_HISTORIES  
C\_CERTIFICATIONS  
C\_CERT\_NUTR\_EDS  
C\_FOOD\_BOX\_DISTs  
C\_BLOODWORK\_DATA  
C\_CERT\_TERM\_REASONS  
C\_DIETARY\_ASSESSMENTS  
C\_DIET\_NUTRIENTS  
C\_HEALTH\_RISK\_FACTORS  
C\_INFANT\_CHILD\_MEDICALS  
C\_WOMAN\_MEDICALS  
C\_I\_C\_HEALTHS  
C\_I\_C\_HH\_REASONS\_BFENDS  
F\_CASASSIGNMENTS  
F\_WAIT\_LISTS  
F\_WAIT\_LIST\_CONTACTS  
F\_CASELOAD\_RESTRICTIONS  
I\_FOOD\_PACKAGES  
I\_FOOD\_PACKAGFOODS  
I\_PACKAGDISTRIBUTIONS  
C\_FOOD\_PACKAGPRESCRIPTIONS  
I\_BF\_PROMO\_VCHRS  
I\_BF\_VCHR\_ITEMS  
I\_FI\_INVENTORIES  
I\_INVENTORIES  
I\_STOCK\_INVENTORIES  
I\_INV\_ORG\_UNITS  
V\_COMPLAINTS  
V\_MON\_ACTIVITIES  
V\_ACTIVITY\_FINDINGS  
I\_BATCH\_RUNS  
S\_PRINTER\_ADDRESSES  
DEL\_STATEMENTS

#### **1.10.1.2 Selection**

Only update records that have changed since the last successful end of day process:

All of the input tables WHERE (NVL(date\_modified, date\_created) > (SELECT good\_proc\_date FROM eod\_controls))  
Date\_modified or date\_created is greater than good\_proc\_date from eod\_controls table.

### **1.10.1.3** Processing

1. Each record in all of the input tables matching the selection criteria are inserted into the corresponding output table.

2. INSERT INTO EODADM.E\_EOD\_CONTROLS(TO\_DATE, FROM\_DATE,  
GOOD\_PROC\_DATE)

**1.10.1.4** Outputs

E\_EOD\_CONTROLS  
E\_O\_STAFF\_MEMBERS  
E\_O\_STAFF\_TIME\_STUDIES  
E\_I\_FOOD\_INSTRUMENT\_TYPES  
E\_O\_ORGANIZATIONAL\_UNITS  
E\_I\_FOOD\_INSTRUMENT\_FOODS  
E\_O\_ORG\_PROGRAMS  
E\_I\_FOOD\_INSTRUMENTS  
E\_O\_ORG\_UNIT\_PHONES  
E\_I\_FI\_FORMULAS  
E\_O\_OUTREACH\_ORGANIZATIONS  
E\_O\_OUT\_ORG\_LAS  
E\_I\_FOOD\_PACKAGE\_FI\_TYPES  
E\_O\_OUTREACH\_COMMS  
E\_O\_OUTREACH\_ORG\_PHONES  
E\_O\_OUTREACH\_PROGRAMS  
E\_O\_STAFF\_PHONES  
E\_O\_JOB\_DESCRIPTIONS  
E\_C\_FAMILY\_ECONOMIC\_UNITS  
E\_C\_FAMILY\_PHONES  
E\_C\_FAM\_REFERRALS  
E\_C\_FEU\_COMMUNICATIONS  
E\_C\_SMOKING\_HISTORIES  
E\_C\_PREV\_FAMILIES  
E\_C\_CLIENTS  
E\_C\_CERT\_PEER\_COUNSELS  
E\_C\_CONTACTS  
E\_C\_PEER\_RBFENDS  
E\_C\_P\_COUNS\_NOTES  
E\_C\_CLIENT\_COMMUNICATIONS  
E\_C\_CLIENT\_GOALS  
E\_C\_CLIENT\_NOTES  
E\_C\_CLIENT\_PROGS  
E\_C\_CLIENT\_REFERRALS  
E\_C\_CLIENT\_SERVICES  
E\_C\_CLIENT\_NE\_TOPICS  
E\_C\_CLIENT\_SVC\_NE\_MATERIALS  
E\_C\_MORE\_ETHNIC\_GROUPS  
E\_C\_PREV\_NAMES  
E\_C\_BF\_PROMO\_ISSUANCES  
E\_C\_ALLERGY\_FOODS  
E\_C\_W\_HEALTHS  
E\_C\_W\_HH\_REASONS\_BFENDS  
E\_C\_PS\_RESPONSES  
E\_C\_CLIENT\_BP\_ISSUANCES  
E\_C\_INCOME\_HISTORIES

E\_C\_ECONOMIC\_UNIT\_MEMBERS  
E\_C\_INCOMES  
E\_C\_INFANT\_DATA  
E\_C\_RESOLUTIONS  
E\_C\_RESOLVED\_CLIENTS  
E\_C\_TRANSFERS\_INFO  
E\_C\_TRANSFER\_HISTORIES  
E\_C\_CERTIFICATIONS  
E\_C\_CERT\_NUTR\_EDS  
E\_C\_FOOD\_BOX\_DIST  
E\_C\_BLOODWORK\_DATA  
E\_C\_CERT\_TERM\_REASONS  
E\_C\_DIETARY\_ASSESSMENTS  
E\_C\_DIET\_NUTRIENTS  
E\_C\_HEALTH\_RISK\_FACTORS  
E\_C\_INFANT\_CHILD\_MEDICALS  
E\_C\_WOMAN\_MEDICALS  
E\_C\_I\_C\_HEALTHS  
E\_C\_I\_C\_HH\_REASONS\_BFENDS  
E\_F\_CASE\_ASSIGNMENTS  
E\_F\_WAIT\_LISTS  
E\_F\_WAIT\_LIST\_CONTACTS  
E\_F\_CASELOAD\_RESTRICTIONS  
E\_I\_FOOD\_PACKAGES  
E\_I\_FOOD\_PACKAGE\_FOODS  
E\_I\_PACKAGE\_DISTRIBUTIONS  
E\_C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
E\_I\_BF\_PROMO\_VCHRS  
E\_I\_BF\_VCHR\_ITEMS  
E\_I\_FI\_INVENTORIES  
E\_I\_INVENTORIES  
E\_I\_STOCK\_INVENTORIES  
E\_I\_INV\_ORG\_UNITS  
E\_V\_COMPLAINTS  
E\_V\_MON\_ACTIVITIES  
E\_V\_ACTIVITY\_FINDINGS  
E\_I\_BATCH\_RUNS  
E\_S\_PRINTER\_ADDRESSES  
E\_DEL\_STATEMENTS

## **1.10.2 EA1\_EXPORT\_TAB.TXT**

### ***Overview***

This is a text file containing the parameters used by the Oracle Export command that is executed by the EA1\_EXPORT.BAT batch file. The output tables listed above in section 1.10.1.4 are listed in this parameter file and are exported to a file that is later copied to the central server to be imported into the central database.

### **1.10.2.1 Inputs**

N/A

### **1.10.2.2 Selection**

N/A

### **1.10.2.3 Processing**

N/A

### **1.10.2.4 Outputs**

N/A

## 1.11 Initiate Polling Process

### 1.11.1 EC1.BAT

#### Overview

This batch file begins EOD processing at Central.

#### Processing

This script is called from the Systems Administration End of Day form when the proceed button is pressed and the AGCY\_SEQ is equal to '00'.

The script reads in the EC1.INI file to initialize and set the temporary environmental variables.

The script begins by running the SQL script EC1.SQL which updates EOD\_CONTROLS and ENV\_VARIABLES and then runs the following SQL scripts:

- EC1\_OTHER\_PURG.SQL
- EC1\_PRG\_ARCHV.SQL
- EC1\_CTRL\_CHCK.SQL
- EC1\_STALE\_DATE\_FI.SQL

This it calls the following scripts to prepare the vendor PGA bank files.

- EC1\_OUTBOUND\_PGA\_FSMC.SQL
- EC1\_PGA\_FSMC.SQL
- EC1\_OUTBOUND\_VENDOR\_FSMC.SQL
- EC1\_VENDOR\_FSMC.SQL
- EC1\_VENDOR\_FSMC.BAT

This script then calls EC2.BAT, EC4.BAT, and EC5.BAT to complete the EOD process at central.

## 1.12 Delete information no longer used by the system

### 1.12.1 EC1.SQL

#### Overview

Executes SQL scripts for EC1.BAT.

#### 1.12.1.1 Inputs

eod\_controls  
env\_variables

#### 1.12.1.2 Selection

env\_variables  
where code = 'EOD\_BY'

#### 1.12.1.3 Processing

update eod\_controls, setting TO\_DATE to the system date  
update env\_variables, setting env\_value to 'EC1' where code = 'EOD\_BY'

run SQL script EC1\_OTHER\_PURG  
run SQL script EC1\_PRG\_ARCHV  
run SQL script EC1\_CTRL\_CHCK  
run SQL script EC1\_STALE\_DATE\_FI

#### 1.12.1.4 Outputs

end\_controls  
env\_variables

Log files: EC1\_SQL.LOG

**1.12.2 EC1\_OTHER\_PURG.SQL****Overview**

This script deletes various records that are no longer used by the system. The following records are removed if conditions are met (does not apply to clients classified in check out clinic status):

<u>Records Deleted</u>	<u>Conditions</u>
Delete statements	Del-statements with date_created more than 2 months in the past
Income history records	Delete records that don't have corresponding records in c_incomes table
Temp FI records	Delete any records remaining in i_temp_fis because of printing problems
Archived/client records	Delete r_archived_clients records with a date_created more than 15 days in the past

**1.12.2.1 Inputs**

DEL\_STATEMENTS

**1.12.2.2 Selection**

Determines which records are no longer used by the system. This is done in the following processing step in the where clause.

**1.12.2.3 Processing**

Deletes unused records:

```
DELETE FROM DEL_STATEMENTS
WHERE date_created < min_date (min date is two months prior to the current date).
```

```
DELETE FROM C_ECONOMIC_UNIT_MEMBERS
WHERE cih_income_history_id = inc_rec.income_history_id (inc_rec.income_history_id is any
income_history_id that does not have corresponding c_income records)
```

```
DELETE from C_INCOME_HISTORIES
WHERE income_history_id=inc_rec.income_history_id
```

```
DELETE from I_TEMP_FIS
```

```
DELETE from R_ARCHIVED_CLIENTS
WHERE NVL (date_modified, date_created) < (sysdate - 15)
```

#### 1.12.2.4 Outputs

del\_statements

### 1.12.3 EC1\_STALE\_DATE\_FL.SQL

#### Overview

Updates i\_food\_instruments for issued food instruments which have passed their stale date and are no longer redeemable.

#### 1.12.3.1 Input

N/A

#### 1.12.3.2 Selection

i\_food\_instruments  
where idis\_disposition\_code is 1 or 2 (printed not issued, issued) or 5 (rejected)  
and void\_date is null (not voided)  
and cleared\_date is null (not redeemed)  
and stale\_date is before the system date (stale date in the past)

#### 1.12.3.3 Processing

Select the serial\_number for i\_food\_instruments records meeting the selection criteria

Loop through the i\_food\_instruments records with the serial\_number values:

```
update i_food_instruments, setting
  ivr_void_reason_code = 'Z' (stale),
  idis_disposition_code = '3' (voided),
  void_date = current date
  date_modified = Sysdate
  voided_by = 'EODADM'
```

```
Insert the i_food_instrument into i_bank_reports, setting:
  post_code = 'Y'
  send_code = 'B'
  message = 'STALE DATE WITH NORMAL POST'
  rec_status = 'U'
```

end;

**1.12.3.4** Outputs

i\_bank\_reports

i\_food\_instruments

**1.13** Compile direct payment FI information and send to bank**1.13.1** EC1\_CTRL\_CHK.SQL*Overview*

This script initializes the I\_BANK\_REPORTS table and creates I\_BANK\_REPORTS records for food instruments created at the state for compliance cases and replacement checks (issue method 'R').

**1.13.1.1** Inputs

I\_BANK\_REPORTS  
I\_FOOD\_INSTRUMENTS  
EOD\_CONTROLS

**1.13.1.2** Selection

## Selection Criteria 1:

Determines which bank report records need to be created from food instruments and sets the record status to "P" (Insert) to indicate the food instrument is new:

```
FROM i_food_instruments
WHERE (compliance_buy_flag = 'Y'
      OR issue_method = 'P')
      AND date_created >
      (SELECT from_date
       FROM eod_controls)
      AND cleared date is NULL
```

## Selection Criteria 2:

Determines which bank report records need to be created from food instruments that were updated and sets the record status to "P" to indicate the food instrument was updated:

```
FROM i_food_instruments
WHERE date_modified >
      (SELECT from_date
       FROM eod_controls)
      AND cleared_date is null
      AND (compliance_buy_flag = 'Y' or issue_method = 'P')
```

## Selection Criteria 3:

Determines the bank reports that need to be created from food instruments that were revalidated and sets the record status to 'P' to indicate the food instrument was updated.

```
FROM I_food_instruments
WHERE revalidation_code = '2'
      AND cleared_date is null
      AND date_modified >
      (select from_date from eod_control)
```

### 1.13.1.3 Processing

The scripts performs the following delete to initialize the i\_bank\_reports table prior to the insert:  
DELETE FROM i\_bank\_reports

Then the following insert is performed based upon selection criteria 1 above:

```
INSERT INTO i_bank_reports, setting post_code to 'Y', send_code to 'B', message to 'POSTED
COMPLIANCE/REPLACEMENT DRAFT', and rec_status to 'P'.
```

Then the following insert is performed based upon selection criteria 2 above

```
INSERT INTO i_bank_reports, setting post_code to 'Y', send_code to 'B', message to 'DUPLICATE
COMPLIANCE/REPLACEMENT DRAFT', update no post necessary, and rec_status to 'P'.
```

The following is performed based upon selection criteria 3:

```
INSERT INTO i_bank_reports, setting post_code to 'Y', send_code to 'B', manage to 'NORMAL
POSTING OF REVALIDATION CHECK' and' rec_status to 'P'.
```

### 1.13.1.4 Outputs

I\_Bank\_Reports

## **1.14** Compile vendor information to send to bank

### **1.14.1** EC1\_VENDOR\_FSMC.BAT

#### ***Overview***

This batch process creates a vendor text file that will be sent to the bank. If the text file already exists then it will delete it. If the file doesn't exist it will be created. The file is deleted at the end of the batch process. Creates a file of Vendor data to be sent to FSMC: ANVENDOR.DAT.

#### **1.14.1.1** Inputs

vendor\_fsmc.dat (see 1.14.2 for file format)

#### **1.14.1.2** Selection

N/A

#### **1.14.1.3** Processing

Creates a vendor text file that will be sent to the bank. If the previous file has not been sent then the new data is appended to the old file.

if the file ANVENDOR.DAT exists,

```
    Delete VENDOR_FSMC.DAT file
else
    copy the file VENDOR.FSMC.DAT to ANVENDOR.DAT
end;
```

delete the file VENDOR\_FSMC.DAT

#### **1.14.1.4** Outputs

anvendor\_fsmc.dat (see 1.14.2 for file format)

### 1.14.2 EC1\_VENDOR\_FSMC.SQL

#### *Overview*

This script creates a temporary vendor data set to be used for bank processing. The temporary data set is created based upon all records in the vendor table. Creation or change to a record is indicated by the status code of the record.

#### 1.14.2.1 Inputs

V\_VENDORS  
 V\_AUTHORIZATIONS  
 S\_GEO\_LOCATIONS  
 V\_VENDOR\_ACCOUNTS  
 V\_BANK\_BRANCHES  
 V-DISQUALIFICATIONS

#### 1.14.2.2 Selection

A record is created for a Vendor if the following conditions are met:

the v\_vendors.vendor\_id is not null (the Vendor has been given a Vendor ID),  
 the Vendor's status does not indicate "Authorization in Progress".

Select Vendors where:

```

vendor_id is not null
and vendors.vs_status_code != 5
and vendors.id = v_authorizations.ven_id
and vendors.sgeo_geo_location_id_mailed = s_geo_locations.geo_location_id
and vendors.id = v_vendor_accounts.ven_id
and v_vendor_accounts.va_vb_routing_number = v_bank_branches.routing_number
and v_vendor_accounts.wic_checks_deposited_flag = 'y'
and v_authorizations.start_date = (select max(start_date)
                                   from v_authorizations
                                   where v_vendor_accounts.ven_id = ven_id)

```

#### 1.14.2.3 Processing

Creates a temporary vendor data set to be used for bank processing. This includes the vendor id, name, peer group, status, mailing address, and bank information.

The detail information includes the following:

```

SELECT DISTINCT(LPAD(ven.vendor_id,5,0))
,RPAD(NVL(ven.vendor_name,' '),35,' ')

```

```

,RPAD(NVL(SUBSTR(ven.mailing_address1,1,35),' '),35,' ')
,RPAD(NVL(SUBSTR(ven.mailing_address2,1,35),' '),35,' ')
,RPAD(NVL(sgeo.sc_description,' '),35,' ')
,RPAD(NVL(sgeo.ss_state_id,' '),2,' ')
,RPAD(NVL(sgeo.sz_zip5||ven.mailing_zip4,0),9,0)
',' '
,LPAD(TO_NUMBER(ven.pee_peer_group_id),2,0)
,DECODE(vauth.termination_date,NULL,
  DECODE(vd.start_date,NULL,'00000000',
    DECODE(SIGN(vd.end_date - SYSDATE),-1,'00000000',
      RPAD(TO_CHAR(NVL(vd.start_date,SYSDATE),'mmddyyyy'),8,'0'))),
  DECODE(SIGN(vauth.termination_date - SYSDATE),+1,'00000000',
    RPAD(TO_CHAR(NVL(vauth.termination_date,SYSDATE),'mmddyyyy'),8,'0'))))
,RPAD(NVL(SUBSTR(vb.bank_name,1,30),' '),30,' ')
,LPAD(vb.routing_number,9,0)
,RPAD(NVL(va.account_number,' '),10,' ')

```

Also sends another line to the temporary data set, containing the total count of records, preceded by the word "TOTAL".

The Summary information includes the following:

```

'TOTAL',
lpad(ltrim(to_char(count(*))),6,'0')

```

#### 1.14.2.4 Outputs

vendor\_fsmc.dat file

Note: For file layout properties, please refer to Appendix B

**1.14.3 EC1\_PGA\_FSMC.BAT****Overview**

This script reates a peer group averages data file that will be sent to the bank. If the text file already exists then it will delete it. Otherwise it will create and delte the text file. It creates a file of PGA data to be sent to FSMC: azpgaver.dat.

**1.14.3.1 Inputs**

p\_g\_avgs\_fsmc.dat

**1.14.3.2 Selection**

N/A

**1.14.3.3 Processing**

Creates a PGA text file that will be sent to the bank.

```

if p_g_avgs_fsmc.dat exists,
    delete p_g_avgs_fsmc.dat
else
    copy the p_g_avgs_fsmc.dat to azpgaver.dat
and;
delete p_g_avgs_fsmc.dat

```

**1.14.3.4 Outputs**

azpgaver.dat

```

SELECT DISTINCT(LPAD(ven.vendor_id,5,0))
,RPAD(NVL(ven.vendor_name,' '),35,' ')
,RPAD(NVL(SUBSTR(ven.mailing_address1,1,35),' '),35,' ')
,RPAD(NVL(SUBSTR(ven.mailing_address2,1,35),' '),35,' ')
,RPAD(NVL(sgeo.sc_description,' '),35,' ')
,RPAD(NVL(sgeo.ss_state_id,' '),2,' ')
,RPAD(NVL(sgeo.sz_zip5||ven.mailing_zip4,0),9,0)
',' '
,LPAD(TO_NUMBER(ven.pee_peer_group_id),2,0)
,DECODE(vauth.termination_date,NULL,
DECODE(vd.start_date,NULL,'00000000',
DECODE(SIGN(vd.end_date - SYSDATE),-1,'00000000',
RPAD(TO_CHAR(NVL(vd.start_date,SYSDATE),'mmddyyyy'),8,'0'))),
DECODE(SIGN(vauth.termination_date - SYSDATE),+1,'00000000',
RPAD(TO_CHAR(NVL(vauth.termination_date,SYSDATE),'mmddyyyy'),8,'0'))))

```

```
,RPAD(NVL(SUBSTR(vb.bank_name,1,30),' '),30,' ')
,LPAD(vb.routing_number,9,0)
,RPAD(NVL(va.account_number,' '),10,' ')
```

#### 1.14.4 EC1\_PGA\_FSMC.SQL

##### *Overview*

This script creates a text file of vendor peer group averages data for bank processing. All records are sent to the bank each time EOD is run.

##### 1.14.4.1 Inputs

V\_PEER\_GROUP\_AVGS

##### 1.14.4.2 Selection

All peer group averages information are included in the output file.

##### 1.14.4.3 Processing

Creates a text file of vendor peer group averages data for bank processing. This includes the peer group ID, FI type, and maximum redemption amount.

The detail information includes the following:

```
lpad(to_number(pga.pee_peer_group_id),2,0)
,lpad(nvl(substr(pga.ifit_fi_type_code,1,6),0),6,0)
,RPAD(NVL(substr(pga.ifit_fi_type_code,7,2),'XX'),2,'XX')
, '***'
'000000'
,substr(lpad(ltrim(to_char(nvl(pga.max_redemption_amt,0),'9999.99')),7,'0'),1,4)
,substr(lpad(ltrim(to_char(nvl(pga.max_redemption_amt,0),'9999.99')),7,'0'),6,2)
```

Also sends another line to the temporary data set, containing the total count of records, preceded by the word "TOTAL".

The summary information includes the following:

```
'TOTAL'
,lpad(ltrim(to_char(count(*))),6,'0')
```

##### 1.14.4.4 Outputs

ANPGAVER.DAT file

Note: For file layout properties, please refer to Appendix B

## 1.15 EC2.BAT

### Overview

This batch file loads data into Central from each Agency.

### Processing

This script first reads in the EC2.INI file to initialize and set temporary environmental variables.

Runs a truncate section to prepare tables for the Agency loading process which calls the SQL scripts:

```
EC_TRGOFF.SQL
EC2_SP_TRNC.SQL
```

For each Agency:

The script first checks to see if the Agency DMP file exists by looking for the completion flag %.don (wildcard represents the Agency) returns an 'OK' in the %.flg file and deletes the completion flag. If the .dmp file does not exist processing proceeds to the next Agency.

The following SQL scripts are then initiated:

```
EC2_TRNC.SQL
EC2_AGENCY.SQL
EC2B.SQL
```

If the dump file is missing, then it inserts a record in the e\_no\_eod table for next end of day run reference by running EC2\_NO\_EOD.SQL script.

After the Agency dump has been completed the script runs EC2.SQL which runs the following SQL scripts:

```
EC2_SYNC_CLINIC_ASSIGNMENT.SQL
EC2_DUPLICATE_FI_RANGE.SQL
EC2_FOOD_UP.SQL
EC2_FI_UPD.SQL
EC2_FAILED_FI.SQL
EC2_OUTBOUND_ISSUE_FSMC.SQL
EC2_ISSUE_FSMC.SQL
```

**Note:** The Arizona WIC System's End of Day Processing doesn't generate the following reports.

Produces Vendor reports that run quarterly, listed by DFDD section number. The quarterly reports are:

**Section 7 - 3.1.9:** Vendor Quarterly Profiles

**Section 7 - 3.4.4:** High Risk Vendor (AZW350, AZW355)

**Section 7 - 3.5.9:** Vendor Formula Redemption (AZW388)

Produces Vendor reports that run monthly, listed by DFDD section number. The monthly reports are:

- Section 7 - 3.1.5:** Compliance Cases and Sanctions Summary
- Section 7 - 3.1.6:** Monitoring and Sanctions History
- Section 7 - 3.4.11:** Vendor Alpha within Agency (AZW168)
- Section 7 - 3.4.12:** Vendor Application Data
- Section 7 - 3.4.13:** Vendor Authorization
- Section 7 - 3.4.14:** Vendor ID within State (AZW165)
- Section 7 - 3.5.1:** FI Type File Listing (AZW981)
- Section 7 - 3.5.2:** FI Type Price Update (AZW305)
- Section 7 - 3.5.3:** FI's Paid/Voided Index (AZW370)
- Section 7 - 3.5.4:** FI's Paid/Voided by Type Index (AZW370x)
- Section 7 - 3.5.5:** Flagged FI's (AZW320)
- Section 7 - 3.5.6:** Redemption Error (AZW702)
- Section 7 - 3.5.7:** Vendor Food Cost (AZW310)
- Section 7 - 3.5.8:** Vendor Food Cost - Issuing Agency Analysis (AZW315)
- Section 7 - 3.5.10:** Vendor Reject (AZW340)
- Section 7 - 3.5.11:** Vendor Report Card (AZW330)

## 1.16 Consolidate participant information

### 1.16.1 EC\_TRGOFF.SQL

#### Overview

Sets database triggers off, so that table contents can be modified without restrictions.

#### 1.16.1.1 Input

N/A

#### 1.16.1.2 Selection

N/A

#### 1.16.1.3 Processing

Alter the following tables with 'DISABLE ALL TRIGGERS'

F\_ANNUAL\_FACTORS  
I\_FOOD\_INSTRUMENT\_TYPES  
C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
C\_CERTIFICATIONS  
C\_CLIENT\_SERVICES

Alter the following tables with 'DISABLE CONSTRAINT'

I\_FOOD\_INSTRUMENT\_TYPES with constraint IFIT\_IFIT\_FK  
I\_FOOD\_PACKAGE\_FI\_TYPES with constraint IFPFT\_IFPFT\_FK

#### 1.16.1.4 Outputs

N/A

### 1.16.2 EC2\_SP\_TRNC.SQL

**Overview**

This script initializes the following tables by clearing all data in each of the temporary tables:

**1.16.2.1 Inputs**

E\_I\_FOOD\_INSTRUMENT\_TYPES  
E\_I\_FOOD\_INSTRUMENT\_FOODS  
E\_I\_FOOD\_INSTRUMENTS  
E\_I\_FI\_FORMULAS  
E\_I\_FOOD\_PACKAGE\_FI\_TYPES  
A\_C\_STATE\_CLIENTS  
E\_C\_STATE\_CLIENTS

**1.16.2.2 Selection**

N/A

**1.16.2.3 Processing**

The script deletes all data from the tables listed above using the following commands:

```
TRUNCATE TABLE E_I_FOOD_INSTRUMENT_TYPES
TRUNCATE TABLE E_I_FOOD_INSTRUMENT_FOODS
TRUNCATE TABLE E_I_FOOD_INSTRUMENTS
TRUNCATE TABLE E_I_FI_FORMULAS
TRUNCATE TABLE E_I_FOOD_PACKAGE_FI_TYPES
TRUNCATE TABLE A_C_STATE_CLIENTS
TRUNCATE TABLE E_C_STATE_CLIENTS
```

**1.16.2.4 Outputs**

EC2\_SP\_TRNC.LOG

### 1.16.3 EC2\_TRNC.SQL

#### *Overview*

This script initializes the following tables:

- \* End of day controls
- \* Staff members
- \* Organizational units
- \* Organizational programs
- \* Organizational unit phones
- \* Outreach organizations
- \* Outreach organizations LAs
- \* Outreach comms
- \* Outreach organization phones
- \* Outreach programs
- \* Staff phones
- \* Job descriptions
- \* Family economic units
- \* Family phones
- \* Family referrals
- \* FEU communications
- \* Smoking Histories
- \* Previous families
- \* Clients
- \* Client communications
- \* Client goals
- \* Client notes
- \* Client programs
- \* Client referrals
- \* Client Services
- \* Client NE topics
- \* Client services NE materials
- \* More Ethnic Groups
- \* Previous Names
- \* Breastfeeding Promo Issuances
- \* Allergy Foods
- \* PS Responses
- \* Client BP Issuances
- \* Certification Nutr Ed Topics
- \* Food Box Distributions
- \* Certificated peer counsels
- \* Contacts
- \* Peer Fbfends
- \* Counsels notes
- \* Incomes
- \* Income histories
- \* Economic unit members
- \* Immunizations

- \* Infant data
- \* Infant child medicals
- \* Resolutions
- \* Resolved clients
- \* Transfers information
- \* Transfer histories
- \* Certifications
- \* Bloodwork data
- \* Certification term reasons
- \* Dietary assessments
- \* Diet nutrients
- \* Health risk factors
- \* Woman medicals
- \* C W healths
- \* C W HH reasons bfends
- \* C I C healths
- \* C I C HH reasons bfends
- \* Case assignments
- \* F Wait lists
- \* F Wait lists contacts
- \* I Food packages
- \* I Food packages food
- \* I Food package prescriptions
- \* I Package distributions
- \* I BF promotional vouchers
- \* I BF voucher items
- \* I FI inventories
- \* I FI ranges
- \* I inventories
- \* I inventory organizational units
- \* V complaints
- \* V monthly activities
- \* V activity findings
- \* Delete statements

### 1.16.3.1 Inputs

A\_STATE\_CLIENTS  
A\_C\_STATE\_CLIENTS  
A\_F\_POVERTY\_BASES  
A\_EOD\_CONTROLS  
A\_A\_ACTIVITIES  
A\_A\_APPT\_ITEMS  
A\_AAS\_CLASS\_CATEGORIES  
A\_AAS\_ITEMS  
A\_A\_ATTEND\_STATUSES  
A\_A\_OFFICE\_CLOSED  
A\_A\_SERVICES  
A\_I\_AGE\_RANGES  
A\_C\_ANSWERS  
A\_C\_ANSWER\_TYPES

A\_C\_BLOODWORK\_TYPES  
A\_C\_BREAST\_PUMP\_REASONS  
A\_C\_BREAST\_PUMP\_TYPES  
A\_C\_CATEGORIES  
A\_C\_CAT\_BLOOD\_FACTORS  
A\_C\_CAT\_GOALS  
A\_C\_CAT\_NUTR\_EDS  
A\_C\_CAT\_REFERRALS  
A\_C\_COMMUNICATION\_TYPES  
A\_C\_CP\_MESSAGES  
A\_C\_DIAGNOSES  
A\_C\_DIETARY\_REQUIREMENTS  
A\_C\_DIET\_NUTRIENT\_TYPES  
A\_C\_DISABILITIES  
A\_C\_EDUCATION\_LEVELS  
A\_C\_ELEVATIONS  
A\_C\_ETHNIC\_GROUPS  
A\_C\_GOALS  
A\_C\_HC\_PAYEES  
A\_C\_HH\_QUESTIONS  
A\_C\_HH\_RESPONSES  
A\_C\_INCOME\_INTERVALS  
A\_C\_INCOME\_LEVELS  
A\_C\_INCOME\_SOURCES  
A\_C\_INCOME\_VERIFICATIONS  
A\_C\_INFANT\_STATUSES  
A\_C\_LANGUAGES  
A\_C\_MARITAL\_STATUSES  
A\_C\_NCHS\_CLASSIFICATIONS  
A\_C\_NCHS\_TYPES  
A\_C\_NCHS\_DATA  
A\_C\_NO\_CONTACT\_REASONS  
A\_C\_NUTR\_ED\_MATERIALS  
A\_C\_NUTR\_ED\_TOPICS  
A\_C\_PICKUP\_INTERVALS  
A\_C\_PILOT\_QUESTIONS  
A\_C\_PILOT\_STUDIES  
A\_C\_PRIORITIES  
A\_C\_PROOF\_ADDRESSES  
A\_C\_PROOF\_IDENTITIES  
A\_C\_PS\_QUESTIONS  
A\_C\_RACES  
A\_C\_REASONS\_BF\_ENDED  
A\_C\_RISK\_FACTORS  
A\_C\_RISK\_FACTOR\_DIAGNOSES  
A\_C\_RISK\_FACTOR\_TYPES  
A\_C\_RF\_GOALS  
A\_C\_RF\_NUTR\_EDS  
A\_C\_RF\_REFERRALS  
A\_C\_RF\_SERVICES  
A\_C\_HH\_RESPONSE\_RFS  
A\_C\_BMI\_DATA  
A\_C\_BMI\_ANTHROPOMETRIC  
A\_C\_BMI\_WEIGHT\_GAIN  
A\_C\_SCHEDULE\_DAYS  
A\_C\_SMOKING\_CHANGES

A\_C\_SMOKING\_STAGES  
A\_C\_SOURCES\_HEALTH\_CARE  
A\_C\_SYMPTOMS  
A\_C\_TERM\_REASONS  
A\_C\_TOPICS  
A\_C\_VOTER\_REGISTRATIONS  
A\_C\_CAT\_BLOODWORKS  
A\_C\_DESIREABLE\_WEIGHTS  
A\_C\_IMMUNIZATIONS\_NOT\_ASSESSED  
A\_F\_CONTROLS  
A\_F\_CASELOAD\_TYPES  
A\_F\_FUND\_SOURCES  
A\_F\_POVERTY\_LEVELS  
A\_F\_WAIT\_LIST\_RESPONSES  
A\_I\_BANK\_DISPOSITIONS  
A\_I\_CATEGORY\_GROUPS  
A\_I\_PACKAGES  
A\_I\_PRODUCTS  
A\_I\_REJECT\_REASONS  
A\_I\_CONTAINERS  
A\_I\_DISPOSITIONS  
A\_I\_UNITS\_OF\_MEASURE  
A\_I\_VOID\_REASONS  
A\_I\_FOOD\_GROUPS  
A\_I\_FOODS  
A\_I\_FOOD\_DISTRIBUTIONS  
A\_I\_MAXIMUM\_FOODS  
A\_I\_FOOD\_PACKAGES  
A\_I\_CATEGORY\_GROUP\_PKGS  
A\_I\_FOOD\_PACKAGE\_FL\_TYPES  
A\_I\_FOOD\_PACKAGE\_FOODS  
A\_I\_PACKAGE\_DISTRIBUTIONS  
A\_I\_WS\_FOOD\_GROUPS  
A\_I\_FOOD\_INSTRUMENT\_TYPES  
A\_I\_FOOD\_INSTRUMENT\_FOODS  
A\_C\_FOOD\_PKG\_RISK\_FACTORS  
A\_O\_OUTREACH\_ORG\_TYPES  
A\_O\_OUTREACH\_COMM\_TYPES  
A\_O\_OUTREACH\_ORGANIZATIONS  
A\_O\_OUTREACH\_ORG\_PHONES  
A\_O\_OUTREACH\_COMMS  
A\_O\_PROGRAMS  
A\_O\_PROGRAM\_DATES  
A\_O\_TITLE\_CATEGORIES  
A\_O\_STAFF\_TITLES  
A\_S\_CITIES  
A\_S\_CONTACT\_METHODS  
A\_S\_CONTACT\_TITLES  
A\_S\_COUNTIES  
A\_S\_PHONE\_TYPES  
A\_S\_STATES  
A\_S\_ZIPS  
A\_V\_ACTIVITY\_TYPES  
A\_V\_APPLICATION\_MILESTONES  
A\_V\_APPLICATION\_TYPES  
A\_V\_BANK\_BRANCHES

A\_V\_CASE\_STATUSES  
A\_V\_COLLECTION\_TYPES  
A\_V\_COMMUNICATION\_TYPES  
A\_V\_COMPLAINT\_SOURCE\_TYPES  
A\_V\_COMPLAINT\_STATUSES  
A\_V\_COMPLAINT\_SUBJECTS  
A\_V\_COMPLIANCE\_CASE\_DESIGNATNS  
A\_V\_COMPLIANCE\_CASE\_TYPES  
A\_V\_DELIVERY\_TYPES  
A\_V\_DENIAL\_REASONS  
A\_V\_DISQUAL\_REASONS  
A\_V\_FINDING\_CODES  
A\_V\_FSP\_REGIONAL\_OFFICES  
A\_V\_FSP\_VIOLATION\_CODES  
A\_V\_LEGAL\_FIRMS  
A\_V\_LEGAL\_REPRESENTATIVES  
A\_V\_MILESTONE\_TYPES  
A\_V\_OWNER\_TYPES  
A\_V\_PEER\_GROUPS  
A\_V\_RISK\_LEVELS  
A\_V\_SANCTION\_TYPES  
A\_V\_STATUSES  
A\_V\_SUSPENSION\_REASONS  
A\_V\_VIOLATION\_ACTION  
A\_V\_WHOLESALERS  
A\_V\_WIC\_CODES  
A\_V\_OWNERS  
A\_V\_VENDORS  
A\_O\_STAFF\_MEMBERS  
A\_O\_ORGANIZATIONAL\_UNITS  
A\_O\_ANNUAL\_WIC\_COST\_SUMMARIES  
A\_F\_ANNUAL\_FACTORS  
A\_F\_BUDGETS  
A\_F\_CASE\_ASSIGNMENTS  
A\_F\_CASELOADS  
A\_F\_CASELOAD\_RESTRICTIONS  
A\_F\_MFR\_CONTACTS  
A\_F\_MANUFACTURERS  
A\_I\_FI\_INVENTORIES  
A\_I\_STOCK\_INVENTORIES  
A\_I\_FOOD\_INSTRUMENTS  
A\_I\_FI\_REJECT\_REASONS  
A\_C\_RESOLUTIONS  
A\_C\_RESOLVED\_CLIENTS  
A\_S\_GEO\_LOCATIONS;  
A\_S\_PRINTER\_ADDRESSES;  
A\_DEL\_STATEMENTS  
R\_S\_CLIENT\_ARCHIVES  
PROMPT table containing clients who have been retrieved from the archives  
R\_ARCHIVED\_CLIENTS  
PROMPT table containing clients who have been archived at central

### 1.16.3.2 Selection

N/A

### **1.16.3.3 Processing**

The script deletes all data from the tables listed above using the following command for each table:

```
TRUNCATE TABLE TABLE_NAME
```

### **1.16.3.4 Outputs**

N/A

#### 1.16.4 EC2\_AGENCY.SQL

##### Overview

Executes SQL scripts for EC2.BAT.

##### 1.16.4.1 Input

env\_variables

##### 1.16.4.2 Selection

env\_variables  
where code = 'EOD\_BY'

##### 1.16.4.3 Processing

update eod\_controls, setting TO\_DATE to the system date  
update env\_variables, setting env\_value to 'EC2' where code = 'EOD\_BY'

run SQL script EC2\_UPD\_E\_C\_CLIENTS  
run SQL script EC2\_DELTRIG  
run SQL script EC2\_IN\_TAB  
run SQL script EC2\_TAB\_UP  
run SQL script EC2\_DELTRIG\_AU  
run SQL script EC2\_ST\_CLIENTS\_SUM

##### 1.16.4.4 Output

env\_variables

Log file: EC2\_AGENCY.LOG

EC2\_UPD\_E\_C\_CLIENTS.SQL

**Overview**

This script updates the last\_first\_bdate column of the table e\_c\_clients table to speed up the dual enrollment process.

**1.16.4.5 Inputs**

E\_C\_CLIENTS

**1.16.4.6 Selections**

N/A

**1.16.4.7 Processing**

The script updates the last\_first\_bdate column of the e\_c\_clients table using the following command.

```
UPDATE E_C_CLIENTS
SET LAST_FIRST_BDATE=SUBSTR (LAST_NAME, 1, 4) ||
                        SUBSTR(first_name, 1, 6) ||
                        to_char(birth_date, 'MMYYYY')
where compliance_flag = 'N'
```

**1.16.4.8 Outputs**

N/A

### 1.16.5 EC2\_DELTRIG.SQL

#### *Overview*

This script deletes data from the central database that had been deleted from the local agency database since the last end of day run excluding the c\_family\_economic\_units and c\_clients table.

#### 1.16.5.1 Inputs

E\_DEL\_STATEMENTS  
EOD\_CONTROLS

#### 1.16.5.2 Selection

```
E_DEL_STATEMENTS WHERE target_proc_date IS NULL  
AND table_name not in('C_FAMILY_ECONOMIC_UNITS', 'C_CLIENTS')  
ORDER BY origin_agcy, del_seq
```

#### 1.16.5.3 Processing

Loop through each E\_DEL\_STATEMENTS record meeting the selection criteria

Execute the SQL delete statement found in the del\_statement column.

End loop

```
UPDATE e_del_statements SET target_proc_date = eod_controls.to_date  
WHERE target_proc_date IS NULL  
AND table_name not in('C_FAMILY_ECONOMIC_UNITS', 'C_CLIENTS')
```

#### 1.16.5.4 Outputs

Log file (EC2\_AGENCY.LOG) containing a list of all delete statements executed.

### 1.16.6 EC2\_DELTRIG\_AU.SQL

#### *Overview*

This script deletes data from the central database that had been deleted from the local agency database since the last end of day run for the c\_family\_economic\_units and c\_clients tables only.

#### 1.16.6.1 Inputs

E\_DEL\_STATEMENTS  
EOD\_CONTROLS

#### 1.16.6.2 Selection

```
E_DEL_STATEMENTS WHERE target_proc_date is NULL  
AND table_name = 'CLIENTS'  
ORDER BY origin_agcy, del_seq
```

```
E_DEL_STATEMENTS WHERE target_proc_date IS NULL  
AND table_name = 'C_FAMILY_ECONOMIC_UNITS'  
ORDER BY origin_agcy, del_seq
```

#### 1.16.6.3 Processing

Loop through each E\_DEL\_STATEMENTS record meeting the selection criteria for c\_clients first.

Execute the SQL delete statement found in the del\_statement column.

End loop

Loop through each E\_DEL\_STATEMENTS record meeting the selection criteria.

Execute the SQL delete statement found in the del\_statement column.

End loop

```
UPDATE e_del_statements SET target_proc_date = eod_controls.to_date  
WHERE target_proc_date IS NULL  
AND table_name in ('C_FAMILY_ECONOMIC_UNITS', 'C_CLIENTS')
```

**1.16.6.4** Outputs

Log file (EC2\_AGENCY.LOG) containing a list of all delete statements executed.

### 1.16.7 EC2\_IN\_TAB.SQL

#### *Overview*

This script inserts client information stored at the central database that has been created at the local agencies, copied to the central database, and stored in the E\_\* tables. It compares the created date in each of the following tables to the last day that the end of day processes ran successfully.

- Staff members
- Staff time studies
- Organizational units
- Organizational programs
- Organizational unit phones
- Outreach organizations
- Outreach\_org las
- Outreach comms
- Outreach organization phones
- Outreach programs
- Staff phones
- Job descriptions
- Family economic units
- Family phones
- Family referrals
- FEU communications
- C Food Package Prescriptions
- Client PS Responses
- C\_food\_box distributions
- Clients
- Client BP Issuances
- Client More Ethnic Groups
- Client Previous Names
- Client communications
- Client goals
- Client notes
- Client programs
- Client referrals
- Client Services
- Client NE topics
- Client services NE materials
- Cert Nutr Eds
- Certificated peer counsels
- Contacts
- C Allergy foods
- Peer Fbfends
- Counsels notes

- Incomes
- Income histories
- Economic unit members
- Immunizations
- Infant data
- Infant child medicals
- Resolutions
- Resolved clients
- Transfers information
- Transfer histories
- Certifications
- Bloodwork data
- Certification term reasons
- Dietary assessments
- Diet nutrients
- Health risk factors
- Woman medicals
- C B W healths
- C B W HH reasons bfends
- C I C healths
- C I C HH reasons bfends
- C P healths
- C infant child medicals
- C woman medicals
- Case assignments
- F Wait lists
- F Wait lists contacts
- I Food packages
- I Food packages food
- I Food package prescriptions
- I Package distributions
- I BF promotional vouchers
- I BF voucher items
- I FI inventories
- I FI ranges
- I FI formulas
- I inventories
- I inventory organizational units
- I stock inventories
- Batch Runs
- Printer Addresses
- V complaints
- V monthly activities
- V activity findings
- Delete statements

#### 1.16.7.1 Inputs

E\_EOD\_CONTROLS  
E\_O\_STAFF\_MEMBERS  
E\_O\_ORGANIZATIONAL\_UNITS  
E\_O\_ORG\_PROGRAMS  
E\_O\_ORG\_UNIT\_PHONES  
E\_O\_OUTREACH\_ORGANIZATIONS  
E\_O\_OUT\_ORG\_LAS  
E\_O\_OUTREACH\_COMMS  
E\_O\_OUTREACH\_ORG\_PHONES  
E\_O\_OUTREACH\_PROGRAMS  
E\_O\_STAFF\_PHONES  
E\_O\_JOB\_DESCRIPTIONS  
E\_C\_FAMILY\_ECONOMIC\_UNITS  
E\_C\_FAMILY\_PHONES  
E\_C\_FAM\_REFERRALS  
E\_C\_FEU\_COMMUNICATIONS  
E\_C\_SMOKING\_HISTORIES  
E\_C\_PREV\_FAMILIES  
E\_C\_CLIENTS  
E\_C\_CERT\_PEER\_COUNSELS  
E\_C\_CONTACTS  
E\_C\_PEER\_RBFENDS  
E\_C\_P\_COUNS\_NOTES  
E\_C\_CLIENT\_COMMUNICATIONS  
E\_C\_CLIENT\_GOALS  
E\_C\_CLIENT\_NOTES  
E\_C\_CLIENT\_PROGS  
E\_C\_CLIENT\_REFERRALS  
E\_C\_CLIENT\_SERVICES  
E\_C\_CLIENT\_NE\_TOPICS  
E\_C\_CLIENT\_SVC\_NE\_MATERIALS  
E\_C\_MORE\_ETHNIC\_GROUPS  
E\_C\_PREV\_NAMES  
E\_C\_BF\_PROMO\_ISSUANCES  
E\_C\_ALLERGY\_FOODS  
E\_C\_W\_HEALTHS  
E\_C\_W\_HH\_REASONS\_BFENDS  
E\_C\_PS\_RESPONSES  
E\_C\_CLIENT\_BP\_ISSUANCES  
E\_C\_INCOME\_HISTORIES  
E\_C\_ECONOMIC\_UNIT\_MEMBERS  
E\_C\_INCOMES  
E\_C\_INFANT\_DATA  
E\_C\_RESOLUTIONS  
E\_C\_RESOLVED\_CLIENTS  
E\_C\_TRANSFERS\_INFO  
E\_C\_TRANSFER\_HISTORIES  
E\_C\_CERTIFICATIONS  
E\_C\_CERT\_NUTR\_EDS  
E\_C\_FOOD\_BOX\_DISTS

E\_C\_BLOODWORK\_DATA  
 E\_C\_CERT\_TERM\_REASONS  
 E\_C\_DIETARY\_ASSESSMENTS  
 E\_C\_DIET\_NUTRIENTS  
 E\_C\_HEALTH\_RISK\_FACTORS  
 E\_C\_INFANT\_CHILD\_MEDICALS  
 E\_C\_WOMAN\_MEDICALS  
 E\_C\_I\_C\_HEALTHS  
 E\_C\_I\_C\_HH\_REASONS\_BFENDS  
 E\_F\_CASE\_ASSIGNMENTS  
 E\_F\_WAIT\_LISTS  
 E\_F\_WAIT\_LIST\_CONTACTS  
 E\_I\_FOOD\_PACKAGES  
 E\_I\_FOOD\_PACKAGE\_FOODS  
 E\_I\_PACKAGE\_DISTRIBUTIONS  
 E\_C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
 E\_I\_BF\_PROMO\_VCHRS  
 E\_I\_BF\_VCHR\_ITEMS  
 E\_I\_FI\_INVENTORIES  
 E\_I\_INVENTORIES  
 E\_I\_INV\_ORG\_UNITS  
 E\_V\_COMPLAINTS  
 E\_V\_MON\_ACTIVITIES  
 E\_V\_ACTIVITY\_FINDINGS  
 E\_DEL\_STATEMENTS

#### 1.16.7.2 Selection

Only records that have been created since the last successful end of day process (This selection is done for each of the above tables):

```
TABLE_NAME WHERE date_created > (SELECT good_proc_date FROM EODADM.e_eod_controls)
```

#### 1.16.7.3 Processing

For each table listed in the outputs section the system inserts the records found in the corresponding E\_\* table found in the inputs section

Loop

```
Insert into TABLE_NAME (select * from E_TABLE_NAME where date_created > (SELECT
good_proc_date FROM EODADM.e_eod_controls));
```

End Loop

For example the insert statement for the O\_STAFF\_MEMBERS table is:

```
Insert into O_STAFF_MEMBERS as select * from E_O_STAFF_MEMBERS;
WHERE date_created > (select good_proc_date from eodadm.e_eod_controls);
```

**1.16.7.4** Outputs

O\_STAFF\_MEMBERS  
O\_STAFF\_TIME\_STUDIES  
O\_ORG\_PROGRAMS  
O\_ORG\_UNIT\_PHONES  
O\_OUTREACH\_ORGANIZATIONS  
O\_OUT\_ORG\_LAS  
O\_OUTREACH\_COMMS  
O\_OUTREACH\_ORG\_PHONES  
O\_OUTREACH\_PROGRAMS  
O\_STAFF\_PHONES  
O\_JOB\_DESCRIPTIONS  
C\_FAMILY\_ECONOMIC\_UNITS  
C\_FAMILY\_PHONES  
C\_FAM\_REFERRALS  
C\_FEU\_COMMUNICATIONS  
C\_SMOKING\_HISTORIES  
C\_CLIENTS  
C\_CLIENTS  
C\_PREV\_FAMILIES  
C\_CERT\_PEER\_COUNSELS  
C\_CONTACTS  
C\_PEER\_RBFENDS  
C\_P\_COUNS\_NOTES  
C\_CLIENT\_COMMUNICATIONS  
C\_CLIENT\_GOALS  
C\_CLIENT\_NOTES  
C\_CLIENT\_PROGS  
C\_CLIENT\_REFERRALS  
C\_CLIENT\_SERVICES  
C\_CLIENT\_NE\_TOPICS  
C\_CLIENT\_SVC\_NE\_MATERIALS  
C\_MORE\_ETHNIC\_GROUPS  
C\_PREV\_NAMES  
C\_BF\_PROMO\_ISSUANCES  
C\_ALLERGY\_FOODS  
C\_W\_HEALTHS  
C\_W\_HH\_REASONS\_BFENDS  
C\_PS\_RESPONSES  
C\_CLIENT\_BP\_ISSUANCES  
C\_INCOME\_HISTORIES  
C\_ECONOMIC\_UNIT\_MEMBERS  
C\_INCOMES  
C\_INFANT\_DATA  
C\_RESOLVED\_CLIENTS  
C\_RESOLUTIONS  
C\_TRANSFERS\_INFO

C\_TRANSFER\_HISTORIES  
C\_CERTIFICATIONS  
C\_CERT\_NUTR\_EDS  
C\_FOOD\_BOX\_DIST  
C\_BLOODWORK\_DATA  
C\_CERT\_TERM\_REASONS  
C\_DIETARY\_ASSESSMENTS  
C\_DIET\_NUTRIENTS  
C\_HEALTH\_RISK\_FACTORS  
C\_INFANT\_CHILD\_MEDICALS  
C\_WOMAN\_MEDICALS  
C\_I\_C\_HEALTHS  
C\_I\_C\_HH\_REASONS\_BFENDS  
F\_CASE\_ASSIGNMENTS  
F\_WAIT\_LISTS  
F\_WAIT\_LIST\_CONTACTS  
I\_FOOD\_PACKAGES  
I\_FOOD\_PACKAGE\_FOODS  
I\_PACKAGE\_DISTRIBUTIONS  
C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
I\_BF\_PROMO\_VCHRS  
I\_BF\_VCHR\_ITEMS  
I\_FI\_INVENTORIES  
I\_INVENTORIES  
I\_STOCK\_INVENTORIES  
I\_INV\_ORG\_UNITS  
V\_COMPLAINTS  
V\_MON\_ACTIVITIES  
V\_ACTIVITY\_FINDINGS  
F\_CASELOAD\_RESTRICTIONS  
I\_BATCH\_RUNS  
S\_PRINTER\_ADDRESSES  
DEL\_STATEMENTS

### 1.16.8 EC2\_TAB\_UP.SQL

#### *Overview*

This script updates client information stored at the central database that has been modified at the local agencies, copied to the central database, and stored in the E\_\* tables. It compares the modified date in each of the following tables to the last day that the end of day processes ran successfully.

- Staff members
- Staff time studies
- Organizational units
- Organizational programs
- Organizational unit phones
- Outreach organizations
- Outreach\_org las
- Outreach comms
- Outreach organization phones
- Outreach programs
- Staff phones
- Job descriptions
- Family economic units
- Family phones
- Family referrals
- FEU communications
- C Food Package Prescriptions
- Client PS Responses
- C\_food\_box distributions
- Clients
- Client BP Issuances
- Client More Ethnic Groups
- Client Previous Names
- Client communications
- Client goals
- Client notes
- Client programs
- Client referrals
- Client Services
- Client NE topics
- Client services NE materials
- Cert Nutr Eds
- Certificated peer counsels
- Contacts
- C Allergy foods
- Peer Fbfends
- Counsels notes

- Incomes
- Income histories
- Economic unit members
- Immunizations
- Infant data
- Infant child medicals
- Resolutions
- Resolved clients
- Transfers information
- Transfer histories
- Certifications
- Bloodwork data
- Certification term reasons
- Dietary assessments
- Diet nutrients
- Health risk factors
- Woman medicals
- C B W healths
- C B W HH reasons bfends
- C I C healths
- C I C HH reasons bfends
- C P healths
- C infant child medicals
- C woman medicals
- Case assignments
- F Wait lists
- F Wait lists contacts
- I Food packages
- I Food packages food
- I Food package prescriptions
- I Package distributions
- I BF promotional vouchers
- I BF voucher items
- I FI inventories
- I FI ranges
- I FI formulas
- I inventories
- I inventory organizational units
- I stock inventories
- Batch Runs
- Printer Addresses
- V complaints
- V monthly activities
- V activity findings
- Delete statements

#### 1.16.8.1 Inputs

E\_EOD\_CONTROLS  
E\_O\_STAFF\_MEMBERS  
E\_O\_ORGANIZATIONAL\_UNITS  
E\_O\_ORG\_PROGRAMS  
E\_O\_ORG\_UNIT\_PHONES  
E\_O\_OUTREACH\_ORGANIZATIONS  
E\_O\_OUT\_ORG\_LAS  
E\_O\_OUTREACH\_COMMS  
E\_O\_OUTREACH\_ORG\_PHONES  
E\_O\_OUTREACH\_PROGRAMS  
E\_O\_STAFF\_PHONES  
E\_O\_JOB\_DESCRIPTIONS  
E\_C\_FAMILY\_ECONOMIC\_UNITS  
E\_C\_FAMILY\_PHONES  
E\_C\_FAM\_REFERRALS  
E\_C\_FEU\_COMMUNICATIONS  
E\_C\_SMOKING\_HISTORIES  
E\_C\_PREV\_FAMILIES  
E\_C\_CLIENTS  
E\_C\_CERT\_PEER\_COUNSELS  
E\_C\_CONTACTS  
E\_C\_PEER\_RBFENDS  
E\_C\_P\_COUNS\_NOTES  
E\_C\_CLIENT\_COMMUNICATIONS  
E\_C\_CLIENT\_GOALS  
E\_C\_CLIENT\_NOTES  
E\_C\_CLIENT\_PROGS  
E\_C\_CLIENT\_REFERRALS  
E\_C\_CLIENT\_SERVICES  
E\_C\_CLIENT\_NE\_TOPICS  
E\_C\_CLIENT\_SVC\_NE\_MATERIALS  
E\_C\_MORE\_ETHNIC\_GROUPS  
E\_C\_PREV\_NAMES  
E\_C\_BF\_PROMO\_ISSUANCES  
E\_C\_ALLERGY\_FOODS  
E\_C\_W\_HEALTHS  
E\_C\_W\_HH\_REASONS\_BFENDS  
E\_C\_PS\_RESPONSES  
E\_C\_CLIENT\_BP\_ISSUANCES  
E\_C\_INCOME\_HISTORIES  
E\_C\_ECONOMIC\_UNIT\_MEMBERS  
E\_C\_INCOMES  
E\_C\_INFANT\_DATA  
E\_C\_RESOLUTIONS  
E\_C\_RESOLVED\_CLIENTS  
E\_C\_TRANSFERS\_INFO  
E\_C\_TRANSFER\_HISTORIES  
E\_C\_CERTIFICATIONS  
E\_C\_CERT\_NUTR\_EDS  
E\_C\_FOOD\_BOX\_DISTS

E\_C\_BLOODWORK\_DATA  
E\_C\_CERT\_TERM\_REASONS  
E\_C\_DIETARY\_ASSESSMENTS  
E\_C\_DIET\_NUTRIENTS  
E\_C\_HEALTH\_RISK\_FACTORS  
E\_C\_INFANT\_CHILD\_MEDICALS  
E\_C\_WOMAN\_MEDICALS  
E\_C\_I\_C\_HEALTHS  
E\_C\_I\_C\_HH\_REASONS\_BFENDS  
E\_F\_CASE\_ASSIGNMENTS  
E\_F\_WAIT\_LISTS  
E\_F\_WAIT\_LIST\_CONTACTS  
E\_I\_FOOD\_PACKAGES  
E\_I\_FOOD\_PACKAGE\_FOODS  
E\_I\_PACKAGE\_DISTRIBUTIONS  
E\_C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
E\_I\_BF\_PROMO\_VCHRS  
E\_I\_BF\_VCHR\_ITEMS  
E\_I\_FI\_INVENTORIES  
E\_I\_INVENTORIES  
E\_I\_INV\_ORG\_UNITS  
E\_V\_COMPLAINTS  
E\_V\_MON\_ACTIVITIES  
E\_V\_ACTIVITY\_FINDINGS  
E\_DEL\_STATEMENTS

#### 1.16.8.2 Selection

Only records that have been updated since the last successful end of day process (This selection is done for each of the above tables):

```
TABLE_NAME WHERE date_modified > (SELECT good_proc_date FROM  
EODADM.e_eod_controls)
```

#### 1.16.8.3 Processing

The system updates each table listed in the outputs section to match the records found in the corresponding E\_\* table from the inputs section.

Loop

```
Update TABLE_NAME set (column list) = (select * from E_TABLE_NAME where  
date_modified > (SELECT good_proc_date FROM EODADM.e_eod_controls);
```

End Loop

**1.16.8.4** Outputs

O\_STAFF\_MEMBERS  
O\_STAFF\_TIME\_STUDIES  
O\_ORG\_PROGRAMS  
O\_ORG\_UNIT\_PHONES  
O\_OUTREACH\_ORGANIZATIONS  
O\_OUT\_ORG\_LAS  
O\_OUTREACH\_COMMS  
O\_OUTREACH\_ORG\_PHONES  
O\_OUTREACH\_PROGRAMS  
O\_STAFF\_PHONES  
O\_JOB\_DESCRIPTIONS  
C\_FAMILY\_ECONOMIC\_UNITS  
C\_FAMILY\_PHONES  
C\_FAM\_REFERRALS  
C\_FEU\_COMMUNICATIONS  
C\_SMOKING\_HISTORIES  
C\_CLIENTS  
C\_CLIENTS  
C\_PREV\_FAMILIES  
C\_CERT\_PEER\_COUNSELS  
C\_CONTACTS  
C\_PEER\_RBFENDS  
C\_P\_COUNS\_NOTES  
C\_CLIENT\_COMMUNICATIONS  
C\_CLIENT\_GOALS  
C\_CLIENT\_NOTES  
C\_CLIENT\_PROGS  
C\_CLIENT\_REFERRALS  
C\_CLIENT\_SERVICES  
C\_CLIENT\_NE\_TOPICS  
C\_CLIENT\_SVC\_NE\_MATERIALS  
C\_MORE\_ETHNIC\_GROUPS  
C\_PREV\_NAMES  
C\_BF\_PROMO\_ISSUANCES  
C\_ALLERGY\_FOODS  
C\_W\_HEALTHS  
C\_W\_HH\_REASONS\_BFENDS  
C\_PS\_RESPONSES  
C\_CLIENT\_BP\_ISSUANCES  
C\_INCOME\_HISTORIES  
C\_ECONOMIC\_UNIT\_MEMBERS  
C\_INCOMES  
C\_INFANT\_DATA  
C\_RESOLVED\_CLIENTS  
C\_RESOLUTIONS  
C\_TRANSFERS\_INFO  
C\_TRANSFER\_HISTORIES  
C\_CERTIFICATIONS

C\_CERT\_NUTR\_EDS  
C\_FOOD\_BOX\_DIST  
C\_BLOODWORK\_DATA  
C\_CERT\_TERM\_REASONS  
C\_DIETARY\_ASSESSMENTS  
C\_DIET\_NUTRIENTS  
C\_HEALTH\_RISK\_FACTORS  
C\_INFANT\_CHILD\_MEDICALS  
C\_WOMAN\_MEDICALS  
C\_I\_C\_HEALTHS  
C\_I\_C\_HH\_REASONS\_BFENDS  
F\_CASE\_ASSIGNMENTS  
F\_WAIT\_LISTS  
F\_WAIT\_LIST\_CONTACTS  
I\_FOOD\_PACKAGES  
I\_FOOD\_PACKAGE\_FOODS  
I\_PACKAGE\_DISTRIBUTIONS  
C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
I\_BF\_PROMO\_VCHRS  
I\_BF\_VCHR\_ITEMS  
I\_FI\_INVENTORIES  
I\_INVENTORIES  
I\_STOCK\_INVENTORIES  
I\_INV\_ORG\_UNITS  
V\_COMPLAINTS  
V\_MON\_ACTIVITIES  
V\_ACTIVITY\_FINDINGS  
F\_CASELOAD\_RESTRICTIONS  
I\_BATCH\_RUNS  
S\_PRINTER\_ADDRESSES  
DEL\_STATEMENTS

### 1.16.9 EC2\_ST\_CLIENTS\_SUM.SQL

#### *Overview*

The c\_state\_clients table is updated based upon records in the family, client, and certification temporary tables that came up from the agencies. The c\_state\_clients table is sent down during the ec4 process for dual enrollment checks at the agencies.

#### **1.16.9.1** 1.17.9.1 Inputs

e\_c\_family\_economic\_units  
c\_family\_economic\_units  
e\_c\_clients  
c\_clients  
e\_c\_certifications  
c\_certifications

#### **1.16.9.2** Selection

All records in e\_c\_family\_economic\_units, e\_c\_clients, and e\_c\_certifications are selected.

#### **1.16.9.3** Processing

Insert client records into c\_state\_clients that exist in e\_c\_clients but not c\_state\_clients.  
Update all records found in c\_state\_clients with a corresponding record in e\_c\_family\_economic\_units, e\_c\_clients, or e\_c\_certifications.

#### **1.16.9.4** Outputs

c\_state\_clients

**1.16.10 EC2B.SQL****Overview**

Executes SQL scripts for EC2.BAT.

**1.16.10.1 Inputs**

env\_variables

**1.16.10.2 Selection**

env\_variables  
where code = 'EOD\_BY'

**1.16.10.3 Processing**

update eod\_controls, setting TO\_DATE to the system date  
update env\_variables, setting env\_value to 'EC2b' where code = 'EOD\_BY'

run SQL script EC2\_CL\_REC\_PROCESS  
run SQL script EC2\_DEL\_CL\_FUT\_RECS  
run SQL script EC2\_DUAL\_ENROLL

**1.16.10.4 Output**

env\_variables

Log file: EC2b\_SQL.LOG

### 1.16.11 EC2\_DEL\_CL\_FUT\_RECS.SQL

#### *Overview*

This script is used to delete future records of a client with respect to the old Local Agency when the client is transferred, or when the client is terminated. This needs to be executed at central site after EOD data from all local agencies are received and before EC2\_STATE\_CTLI.SQL is run.

#### 1.16.11.1 Inputs

C\_CLIENT\_SERVICES  
C\_CLIENT\_SVC\_NE\_MATERIALS  
C\_CLIENT\_NE\_TOPICS

#### 1.16.11.2 Selection

Select all clients from the C\_CLIENT\_SERVICES table where the following criteria are met:  
C\_CLIENT\_SERVICES.CLIENT\_ID equals ID of client to be transferred, the service date is on or after the termination date, the appointment flag is set to 'Y', and the date created and date modified are before the termination date.

#### 1.16.11.3 Processing

For all clients to be transferred:

- Delete record from C\_CLIENT\_SVC\_NE\_MATERIALS table,
- Delete record from C\_CLIENT\_NE\_TOPICS table,
- Delete record from C\_CLIENT\_SERVICES table.

#### 1.16.11.4 Outputs

C\_CLIENT\_SERVICES  
C\_CLIENT\_SVC\_NE\_MATERIALS  
C\_CLIENT\_NE\_TOPICS

## 1.16.12 EC\_TRGON.SQL

### Overview

Sets database triggers on, so that triggers and foreign keys are re-enabled.

#### 1.16.12.1 Input

N/A

#### 1.16.12.2 Selection

N/A

#### 1.16.12.3 Processing

Alter the following tables with 'ENABLE ALL TRIGGERS'

F\_ANNUAL\_FACTORS  
I\_FOOD\_INSTRUMENT\_TYPES  
C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
C\_CERTIFICATIONS  
C\_CLIENT\_SERVICES

Alter the following tables with 'ENABLE CONSTRAINT'

I\_FOOD\_INSTRUMENT\_TYPES with constraint IFIT\_IFIT\_FK  
I\_FOOD\_PACKAGE\_FI\_TYPES with constraint IFPFT\_IFPFT\_FK

#### 1.16.12.4 Output

N/A

**1.17** Financial - process totals and store in F\_CASELOAD table. Totals are based on priority, language, caseload type, ethnic group and poverty level

### **1.17.1** EC2\_CL\_REC\_PROCESS.SQL

#### **Overview**

This end of day script is used to insert updated records into caseload type detail and caseloads for each certification record that has been created or modified. For more information, see the Financial Management Introduction.

#### **1.17.1.1** Inputs

env\_variables  
eod\_controls  
e\_eod\_controls  
e\_c\_certifications  
c\_clients  
c\_family\_economic\_units  
f\_caseload\_type\_details  
f\_incomes\_to\_poverties  
f\_caseloads

### **1.17.2** PROCEDURE EODP\_CL\_REC\_PROC\_P1

This EOD script is to process any changes in c\_certification / c\_client data. Inserts and Updates are performed on the F\_CASELOAD\_TYPE\_DETAILS and F\_CASELOADS tables. Executed at Central site only.

#### **1.17.2.1** Processing

Select all C\_CERTIFICATIONS records where date\_created or date\_modified is between EOD\_CONTROLS.FROM\_DATE and EOD\_CONTROLS.TO\_DATE.

If C\_CERTIFICATIONS.TERMINATION\_DATE is null, then  
For all months in the C\_CERTIFICATION record ranging from the CERT\_START\_DATE to the CERT\_END\_DATE

Select the max(F\_INCOMES\_TO\_POVERTIES.FPL\_ID) for the Participant's income and family size.

Select the F\_CASELOAD\_TYPE\_DETAILS record for the Federal Fiscal Month and Year, Program, Clinic, Category, Priority, Language, Ethnic Group, Migrant Flag, Refugee Flag, Adjunctive Eligibility Flag, Income, Family Size, and

Poverty Level from the C\_CERTIFICATIONS record where PARTICIPANT\_FLAG = 'N'.

If the F\_CASELOAD\_TYPE\_DETAILS record does not exist, then  
Insert the information from the select statement into a new record.

If EOD\_CONTROLS.IPL\_DATE is in the same or a later month than the C\_CERTIFICATION month and the C\_CERTIFICATION month is in the same month as the month when the C\_CERTIFICATION record was last modified or created, then

Run the EODP\_I\_U\_CASELOAD\_P1('ENROLLEE')

If the MIGRANT\_FLAG for the C\_CERTIFICATIONS record equals 'Y', then

Run the EODP\_I\_U\_CASELAOD\_P1('MIGRANT')

If the REFUGEE\_FLAG for the C\_CERTIFICATIONS record equals 'Y', then

Run the EODP\_I\_U\_CASELAOD\_P1('REFUGEE')

If the C\_CERTIFICATION record is for a Category of 'IEN' with a Rec Status = 'A' and a Mother's ID where that Mother's Client ID is in a current C\_FOOD\_PACKAGE\_PRESCRIPTION record, then

Run the EODP\_I\_U\_CASELAOD\_P1('PARTICIPANT')

### 1.17.3 PROCEDURE EODP\_I\_U\_CASELOAD\_P1

This EOD function is used to insert update changes into the caseload table. It is called by the client record processing procedure (EODP\_CL\_REC\_PROC\_P1).

**1.17.3.1 Processing**

For the Caseload Type passed in the parameter of the procedure call  
Select the Client Count from F\_CASELOADS for the Fiscal Month and Year, Caseload Type, Program, Clinic, Language, Priority, Category, Ethnic Group, and Poverty Level from the F\_CASELOAD\_TYPE\_DETAILS record in the EODP\_C1\_REC\_PROC\_P1 procedure.

If the F\_CASELOAD record does not exist, then  
Insert the information from the select statement into a new record. The Client Count will be set to 1.

Else

Update the F\_CASELOAD record by incrementing the Client Count by 1.

**1.17.3.2 Outputs**

f\_caseloads

f\_caseload\_type\_details

## 1.17.4 EC2.SQL

### Overview

Executes SQL scripts for EC2.BAT.

#### 1.17.4.1 Input

env\_variables

#### 1.17.4.2 Selection

```
env_variables
  where code = 'EOD_BY'
```

#### 1.17.4.3 Processing

```
update eod_controls, setting TO_DATE to the system date
update env_variables, setting env_value to 'EC2' where code = 'EOD_BY'
```

```
run SQL script EC2_SYNC_ASSIGNMENT
run SQL script EC2_DUPLICATE_FI RANGE
run SQL script EC2_FOOD_UP
run SQL script EC2_FI_UPD
run SQL script EC2_FAILED_FI.SQL
run SQL script EC2_OUTBOUND_ISSUE_FSMC.SQL
run SQL script EC2_ISSUE_FSMC
```

#### 1.17.4.4 Output

env\_variables

Log files:

```
EC2_SYNC_CLINIC.LOG
EC2_FOOD_UP.LOG
EC2_FI_UPD.LOG
ISSUEANDAT
```

**1.18** Update caseload assignment information from the clinics**1.18.1** EC2\_SYNC\_CLINIC\_ASSIGNMENT.SQL**Overview**

This script is used at the state site for end of day processes to synchronize the caseload data. The caseload assignments are re-allocated by the local agency to their clinics. After this processing the records are deleted from the eodadm.e\_f\_case\_assignment table.

**1.18.1.1** Inputs

eodadm.e\_f\_case\_assignments  
f\_case\_assignments

**1.18.1.2** Selection

Checks to see if caseloads records exist at the local agency level that have been moved up to the state level and locks those records for update. If they do, then the caseload records are updated. If they don't, then new caseload records are inserted.

```
Select 'x'
From f_case_assignments
Where fbu_ffy_month equals eodadm.e_f_case_assignments.fbu_ffy_month
And fbu_fff_ffy equals eodadm.e_f_case_assignments.fbu_fff_ffy
And ou_seq_id equals eodadm.e_f_case_assignments.category_code
And program equals eodadm.e_f_case_assignments.program
FOR UPDATE OF ASSIGNED
```

**1.18.1.3** Processing

```
If f_case_assignments is found, then
    Update f_case_assignments and
        Set assigned equal to eodadm.e_f_case_assignments.assigned
        alloc_factor equal to eodadm.e_f_case_assignments.alloc_factor
        date_modified_by equal to eodadm.e_f_case_assignments.date_modified
        note equal to eodadm.e_f_case_assignments.note
Else
    Insert INTO f_case_assignment the following values
        eodadm.e_f_case_assignments.fbu_ffy_month
        eodadm.e_f_case_assignments.fbu_fff_ffy
        eodadm.e_f_case_assignments.category.code
        eodadm.e_f_case_assignments.assigned
        eodadm.e_f_case_assignments.alloc_factor
```

eodadm.e\_f\_case\_assignments.created\_by  
eodadm.e\_f\_case\_assignments.date\_created  
eodadm.e\_f\_case\_assignments.date\_modified  
eodadm.e\_f\_case\_assignments.modified\_by  
eodadm.e\_f\_case\_assignments.note  
eodadm.e\_f\_case\_assignments.program

Delete from eodadm.a\_f\_case\_assignments

#### **1.18.1.4** Outputs

eodadm.a\_f\_case\_assignments  
f\_case\_assignments

## 1.19 Consolidate and update all food instrument data to calculate FI obligation value

### 1.19.1 EC2\_FOOD\_UP.SQL

#### *Overview*

This sql script will call the ec2\_fi\_upd function from the eod\_fi\_process database package. This function moves all the food instruments that were created at the clinic level up to the state agency and into the I\_BANK\_REPORTS table for bank processing. Initially it will call the procedure: FI\_TYPE\_PROCESS which will copy FI types from the Local Agency to the State database.

#### 1.19.1.1 Inputs

eodadm.e\_i\_food\_instrument\_types  
eodadm.e\_i\_food\_instrument\_foods  
i\_food\_instrument\_types  
i\_food\_instrument\_foods

#### 1.19.1.2 Processing

For each record in EODADM.E\_I\_FOOD\_INSTRUMENTS

    If the food instrument does not exist in AIM.I\_FOOD\_INSTRUMENTS, then

        Insert the FI from EODADM.E\_I\_FOOD\_INSTRUMENTS into  
        AIM.I\_FOOD\_INSTRUMENTS.

        Insert the FI from EODADM.E\_I\_FOOD\_INSTRUMENTS into  
        AIM.I\_BANK\_REPORTS.

        Delete the FI from EODADM.E\_I\_FOOD\_INSTRUMENTS.

    Else

        If the AIM and EODADM FIs are different, then

            Update AIM.I\_FOOD\_INSTRUMENTS with the data from  
            EODADM.E\_I\_FOOD\_INSTRUMENTS

            Insert the FI from EODADM.E\_I\_FOOD\_INSTRUMENTS into  
            AIM.I\_BANK\_REPORTS

#### 1.19.1.3 Outputs

i\_food\_instruments  
i\_bank\_reports

## 1.19.2 EC2\_FI\_UPD.SQL

### Overview

This sql script calls the ec2\_fi\_updates procedure, which updates the financial management tables with the obligation information on the food instruments that were created at the clinic/local agency level and moved to the state agency by the ec2\_food\_up sql file.

### 1.19.2.1 Inputs

- eod\_controls
- i\_bank\_reports
- f\_obligations
- c\_certifications
- c\_food\_package\_prescriptions
- i\_food\_instrument\_foods
- f\_reb\_contracts
- f\_outlays
- f\_reb\_details
- o\_organizational\_units
- i\_dispositions
- f\_annual\_factors
- f\_annual\_category\_factors
- f\_budgets
- i\_food\_instruments
- f\_obligations
- i\_bank\_reports

### 1.19.2.2 Processing

For all records in I\_BANK\_REPORTS where the I\_BANK\_REPORTS.REC\_STATUS != 'P' and I\_BANK\_REPORTS.POST\_CODE = 'Y' and I\_BANK\_REPORTS.IVR\_VOID\_REASON\_CODE != ('A','E') and I\_BANK\_REPORTS.SEND\_CODE != 'L'

If I\_BANK\_REPORTS.REC\_STATUS = 'I' or if I\_BANK\_REPORTS.REC\_STATUS = 'U' and I\_BANK\_REPORTS.MISSING\_ISSUANCE\_FLAG = 'Y', then

If I\_BANK\_REPORTS.ISSUE\_DATE is not null and I\_BANK\_REPORTS.OU\_SEQ\_ID is not the State Agency and I\_BANK\_REPORTS.IFIT\_FI\_TYPE\_CODE is not null, then

Update F\_CASELOAD\_TYPE\_DETAILS.PARTICIPANT\_FLAG = 'Y' for the F\_CASELOAD\_TYPE\_DETAILS.CC\_CLIENT\_ID of the I\_BANK\_REPORTS record.

Select the Client Count from F\_CASELOADS for the Fiscal Month and Year, Program, Clinic, Language, Priority, Category, Ethnic Group, and

Poverty Level where the Caseload Type = 'PARTICIPANT' from the F\_CASELOAD\_TYPE\_DETAILS record that was updated.

If the F\_CASELOAD record does not exist, then

Insert the information from the select statement into a new record. The Client Count will be set to 1.

Else

Update the F\_CASELOAD record by incrementing the Client Count by 1.

Fetch the F\_OBLIGATIONS record for the I\_BANK\_REPORTS.OU\_SEQ\_ID and I\_BANK\_REPORTS.IFIT\_FI\_TYPE\_CODE where I\_BANK\_REPORTS.FIRST\_DATE\_TO\_USE is in F\_OBLIGATIONS.FBU\_FFY\_MONTH and F\_OBLIGATIONS.FBU\_FFF\_FFY.

If the F\_OBLIGATIONS record is not found, then

Insert into F\_OBLIGATIONS

The Category and Priority are derived from I\_BANK\_REPORTS.CC1\_CLIENT\_ID.

The FI Type is set to

I\_BANK\_REPORTS.IFIT\_FI\_TYPE\_CODE

The Fiscal Year and Month are derived from I\_BANK\_REPORTS.FIRST\_DATE\_TO\_USE

The Organizational Unit is set to I\_BANK\_REPORTS.OU\_SEQ\_ID

The FI Created Count and FI Obligated Count

are set to '1'.

If three months of prior data exist in the F\_OUTLAYS table for months where F\_BUDGETS.CLOSED\_OUT\_MONTH\_FLAG = 'Y' then F\_OBLIGATIONS.EST\_REDEEM\_FACTOR is calculated from the actual redemption rate for those three months. This is defined as the (SUM(F\_OUTLAYS.REDEEMED\_FI\_COUNT) for the three months) / (SUM(F\_OBLIGATIONS.FI\_CREATED\_COUNT) - (SUM(F\_OUTLAYS.ALL\_VOIDS) - SUM(F\_OUTLAYS.STALE\_DATED\_VOIDS))). This is the total number of FIs redeemed during the three month period divided by the total number of FIs that were created during the three month period that were either unvoided or voided for stale date purposes only. If three months worth of closed out data does not exist,

then  
 F\_OBLIGATIONS.EST\_REDEEM\_FACTOR  
 is set to  
 F\_ANNUAL\_FACTORS.START\_REDEEM\_FACTOR.

If three months of prior data exist in the  
 F\_OUTLAYS table for months where  
 F\_BUDGETS.CLOSED\_OUT\_MONTH\_FLAG = 'Y' then  
 F\_OBLIGATIONS.EST\_PER\_FI\_VALUE is  
 calculated from the actual per FI value for those  
 three months. This is defined as  
 (SUM(F\_OUTLAYS.REDEEMED\_FI\_VALUE  
 ) for the three months) /  
 (SUM(F\_OUTLAYS.REDEEMED\_FI\_COUNT  
 ) for the three months). If three months worth of  
 closed out data does not exist, then  
 F\_OBLIGATIONS.EST\_PER\_FI\_VALUE is  
 set to  
 F\_ANNUAL\_FACTORS.START\_FPKG\_COST.  
 F\_OBLIGATIONS.ADJ\_PER\_FI\_VALUE is  
 set to  
 F\_OBLIGATIONS.EST\_PER\_FI\_VALUE \*  
 F\_OBLIGATIONS.EST\_REDEEM\_FACTOR.

Else

Update F\_OBLIGATIONS  
 Increment FI Created Count and FI Obligated

Count by 1.

Update I\_FOOD\_INSTRUMENTS  
 Set the Obligation Amount to  
 F\_OBLIGATIONS.EST\_PER\_FI\_VALUE.

If the I\_BANK\_REPORTS record contains an FI Type that  
 consists of a Food for which an  
 F\_REB\_CONTRACTS.IF\_FOOD\_ID exists, then  
 Update I\_FOOD\_INSTRUMENTS  
 Set the Estimated Rebate Value to the  
 SUM(I\_FOOD\_INSTRUMENTS\_FOODS.QUANTITY \*  
 F\_REB\_CONTRACTS.REBATE\_FOR\_UNIT).

If the I\_BANK\_REPORTS record is a Missing Issuance or was  
 Issued and then subsequently Voided, then

Update F\_OBLIGATIONS  
 Decrement the FI Obligated Count by 1

If the I\_BANK\_REPORTS.EST\_REB\_VALUE

is not null, then

by 1.

Decrement the Rebate Value FI Counter

Subtract the  
I\_BANK\_REPORTS.EST\_REB\_VALU  
E from the  
F\_OBLIGATIONS.EST\_REB\_VALUE  
.

### **1.19.2.3** Outputs

See section 1.17.4.4

**1.20** Search for potential dual enrollment clients**1.20.1** EC2\_DUAL\_ENROLL.SQL**Overview**

This script is used to verify clients do not have dual enrollments. The script inserts records into the C\_RESOLVED\_CLIENTS table and C\_RESOLUTIONS table for each potential dual enrollment candidate leaving the C\_RESOLUTIONS.DATE\_RESOLVED column blank to indicate that this potential dual enrollment should be investigated.

**1.20.1.1** Inputs

EOD\_CONTROLS  
 ENV\_VARIABLES  
 C\_RESOLVED\_CLIENTS  
 C\_FAMILY\_ECONOMIC\_UNITS  
 C\_CLIENTS CC  
 C\_CLIENTS CC1  
 C\_CLIENTS CC2  
 C\_RESOLUTIONS  
 E\_C\_CLIENTS  
 C\_CLIENTS\_DUAL

**1.20.1.2** Selection

To retrieve the clients with potential dual enrollments, excluding compliance clients and clients that have already been resolved:

```
C_CLIENTS_DUAL CCD, E_C_CLIENTS ECC
  WHERE ccd.client_id <> ecc.client_id
  AND ccd.lasst_first_bdate <> ec.last_first_bdate
  AND not exists (ccd.client_id, ecc.client_id)
  In (slect cc.client_id, crc.client_id from c_resolutions)
```

Select more details of client from  
 C\_clients & c\_family\_economic\_units table

**1.20.1.3** Processing

For each potential dual enrollment client found Loop

Insert a record into c\_resolved\_clients setting the columns as follows:

CLIENT_ID	= ECC.CLIENT_ID
CAT_CATEGORY_CODE	= C_CLIENTS.CAT_CATEGORY_CODE
OU_SEQ_ID	= CFEU.OU_SEQ_ID
LAST_NAME	= C_CLIENTS.LAST_NAME
FIRST_NAME	= C_CLIENTS.FIRST_NAME
BIRTH_DATE	= C_CLIENTS.BIRTH_DATE
GENDER	= C_CLIENTS.GENDER
DATE_CREATED	= EOD_CONTROLS.TO_DATE
CREATED_BY	= ENV_VARIABLES.ENV_VALUE where code = 'EOD_BY'
MI1	= C_CLIENTS.MI1
MI2	= C_CLIENTS.MI2

Insert a record into c\_resolutions setting the columns as follows:

CRC_CLIENT_ID	= ECC.CLIENT_ID
CC_CLIENT_ID	= CCD.CLIENT_ID
DATE_CREATED	= EOD_CONTROLS.TO_DATE
CREATED_BY	= ENV_VARIABLES.ENV_VALUE where code = 'EOD_BY'

End Loop

#### 1.20.1.4 Outputs

C\_Resolutions  
C\_Resolved\_clients

**1.21** Compile new issuance and void information to send to bank**1.21.1** EC2\_ISSUE\_FSMC.SQL**Overview**

This script creates a text file of food instruments that should be sent to the bank.

**1.21.1.1** Inputs

I\_bank\_reports ibr  
 O\_organizational\_units ou1  
 O\_organizational\_units ou2

**1.21.1.2** Selection

I\_BANK\_REPORTS where send\_code = 'B' (The FI has not been sent to the Bank yet)  
 And ((idis\_disposition\_code = 2 (Issued)  
 And issue\_method in ('B', 'C', 'O', 'R', 'P')) (Batch, Compliance, On-Demand, Re-issued or  
 Replaced)  
 OR idis\_disposition\_code = 3 (Voided)  
 OR idis\_disposition\_code = '5' and revalidation\_code = '2')

**1.21.1.3** Processing

spool issuean.dat

```
select substr (ibr.serial_number, 1, 10)
      select substr(ibr.serial_number,1,10)
      ,lpad(nvl(substr(ibr.ifit_fi_type_code,1,6),0),6,0)
      ,rpad(nvl(substr(ibr.ifit_fi_type_code,7,2),'XX'),2,'XX')
      , '**'
      ,rpad(ou2.org_code,2, ' ')
      , ' '
      , '0000000000'
      ,rpad(to_char(nvl(ibr.issue_date,sysdate),'mmddyyyy'),8,'0')
--      the comment below explains the decode mess that follows it. Kevin M
11/24/2000
--      if disposition = 2 (for issued) then
--          if issue-method = 'O' or 'B' (for On-demand or batch) then
--              return 'D'
```

```

--         elsif issue-method = 'C' (for Compliance-buy) then
--             return 'C'
--         elsif issue-method = 'R' (for Reissued checks) then
--             return 'A'
--         elsif issue-method = 'P' (for Replacement checks) then
--             return 'A'
--     elsif disposition = 3 (for voided) then
--         if void reason = 'B' (for stolen) then
--             return 'S'
--         else return 'V'
--     elsif disposition = 5 (for rejected) then
--         if revalidation code = 2 then
--             return 'O' (over-ride)
--     else return ' '
, rpad(nvl(decode(ibr.idis_disposition_code,'2',decode(ibr.issue_method,'O','D','B','D',
                                                    'C','C','R','A','P','A')
                                                    , '3',decode(ibr.ivr_void_reason_code,'B','S','V')
                                                    , '5', decode(ibr.revalidation_code,'2','O')), ' '),1, ' ')
, rpad(nvl(decode(ibr.idis_disposition_code,'3',ibr.ivr_void_reason_code), ' '),2, ' ')
, rpad(to_char(nvl(ibr.first_date_to_use,sysdate),'mmddyyyy'),8,'0')
, rpad(to_char(nvl(ibr.last_date_to_use,sysdate),'mmddyyyy'),8,'0')
, substr(lpad(ltrim(to_char(nvl(ibr.maximum_amt,0),'9999.99')),7,'0'),1,4)
, substr(lpad(ltrim(to_char(nvl(ibr.maximum_amt,0),'9999.99')),7,'0'),6,2)
, '000000000000'
from
    i_bank_reports ibr
    ,o_organizational_units ou1
    ,o_organizational_units ou2
where ibr.send_code = 'B'
and    ibr.ou_seq_id = ou1.seq_id
and    NVL(ou1.ou_seq_id,10) = ou2.seq_id
and    ((ibr.idis_disposition_code = '2'
and    ibr.issue_method in ('B','C','O','R','P'))
or    ibr.idis_disposition_code = '3'
or    (ibr.idis_disposition_code = '5' and ibr.revalidation_code = '2'));

select 'TOTAL',lpad(ltrim(to_char(count(*))),6,'0')
from
    i_bank_reports ibr
    ,o_organizational_units ou1
    ,o_organizational_units ou2
where ibr.send_code = 'B'
and    ibr.ou_seq_id = ou1.seq_id
and    NVL(ou1.ou_seq_id,10) = ou2.seq_id
and    ((ibr.idis_disposition_code = '2'
and    ibr.issue_method in ('B','C','O','R','P'))
or    ibr.idis_disposition_code = '3'
or    (ibr.idis_disposition_code = '5' and ibr.revalidation_code = '2'));

```

#### 1.21.1.4 Outputs

issuean.dat

### 1.22 EC4.BAT

#### Overview

This batch file is called by EC2.BAT and is part of the Agency upload process.

#### Processing

This script first reads the EC4.INI file to initialize and set temporary environmental variables.

The script then runs the executable file FTPTRANS.EXE which is a visual basic script that does all of the bank interface processing and does the following:

1. Renames the 'issuean.dat' file in the 'dump' directory to 'isanmdd.dat'.
2. Logs into the web site and puts them into the 'load' directory.
3. Copies the three sending files in the 'dump' directory to the web site and moves them to the 'processed' directory.
4. Creates the import\_file.bat file consisting of sql-loads and a move command for all files that haven't previously processed (not in the 'processed' directory).
5. Executes the import\_file.bat script.

Then runs the following SQL scripts:

EC4\_BANK.SQL  
EC4.SQL

EC4.SQL updates ENV\_VARIABLES and EOD\_CONTROLS and then runs the following scripts:

EC4\_CASHFLOW\_UPD.SQL  
EC4\_MON\_END.SQL  
EC4\_VEN\_RISK\_HISTORY.SQL  
EC4\_VEN\_PEER\_FI.SQL  
EC4\_VEN\_PRICE.SQL  
EC4\_VEN\_LOC\_AGCY\_FI.SQL  
EC4\_C1\_SER\_REP.SQL  
EC4\_SYCN\_STATE\_CLIENTS.SQL

For each agency; searches for %.flg (wildcard represents the Agency Id) to see if the Agency for exists, returns 'OK', and then deletes the .flg file.

Then runs the following SQL scripts:

EC4\_STATE\_CLIENTS.SQL  
EC4\_TRNC.SQL  
EC4\_AGCY\_BEING\_PROCESSED.SQL  
EC4\_RETRIEVE.SQL

EC4\_TAB\_DOWN.SQL  
EC4\_STATE\_CLIENTS.SQL  
EC4\_GET\_UPDATES.SQL

Outputs

See Appendix B.

**1.23** Create consolidated record of all FI issuance and redemption information and create vendor payment records

**1.23.1** EC4\_BANK.SQL

**1.23.1.1** Inputs

n/a

**1.23.1.2** Selection

n/a

**1.23.1.3** Processing

Executes the ec4\_bnk\_post.sql script

**1.23.1.4** Outputs

n/a

### 1.23.2 EC4\_BNK\_POST.SQL

#### *Overview*

This script creates consolidated food instrument issuance and redemption information and creates vendor payment records.

The bank data Status values translate into i\_food\_instruments.idis\_disposition\_code values:

'P', 'M', and 'S'	translate into i_food_instruments.idis_disposition_code = '4'
'R'	translates into i_food_instruments.idis_disposition_code = '5'

The bank data Accept Code values are copied directly into i\_food\_instruments.bd\_bank\_disposition\_code.

### 1.23.3 Inputs

E\_BANK\_DATAS  
 I\_FOOD\_INSTRUMENTS  
 F\_OBLIGATIONS  
 F\_OUTLAYS  
 I\_FOOD\_INSTRUMENT\_FOODS  
 F\_REB\_CONTRACTS  
 F\_REB\_DETAILS  
 V\_VENDORS  
 O\_ORGANIZATIONAL\_UNITS  
 I\_DISPOSITIONS

#### 1.23.3.1 Selection

For IDIS\_DISPOSITION\_CODE, I\_FOOD\_INSTRUMENTS.EST\_REB\_VALUE, REDEMPTION\_AMT:

I\_FOOD\_INSTRUMENTS where the serial number equals the serial number from E\_BANK\_DATAS.

For FI\_OBLIG\_COUNT, REB\_FI\_COUNTER, EST\_REB\_VALUE

F\_OBLIGATIONS where F\_OBLIGATIONS FI type code and sequence id equal the food instruments FI type code and sequence id and the F\_OBLIGATIONS category code and priority equal the food instruments category code and priority decoded from the certifications table for the specified time period.

For REDEEMED\_FI\_COUNT, REDEEMED\_FI\_VALUE:

F\_OUTLAYS where F\_OUTLAYS and F\_OBLIGATIONS have equivalent values for category code, FI type, sequence id, priority, and fiscal month and year.

### 1.23.3.2 Processing

Read all records returned from the bank: bank\_data\_rec

Find the i\_food\_instruments record whose serial\_number matches the bank\_data\_rec.serial\_number.

If the i\_food\_instruments.process\_date is not null and i\_food\_instrument.redemption\_amt is not null or bank\_data\_rec.status = 'R'

Insert a record into i\_bank\_datas with the i\_food\_instruments data and set  
Post\_code = 'N'  
Send\_code = 'N'  
Message = 'rejected - duplicate redeemed check from bank'

End

Find the v\_vendors.id from bank\_data\_rec.vendor\_id

If bank\_data\_rec has a null process\_date, amount, or status, write an error message and do not process the record.

Update I\_Food\_Instruments with the following information received from the bank\_data\_rec, received from the bank:

If the bank\_data\_rec.status is 'P', 'M', or 'S' (paid, manual, skeleton),  
    idis\_disposition\_code := 4 (indicating "redeemed")  
    redemption\_amt := bank\_data\_rec.amount / 100  
    cleared\_date := bank\_data\_rec.process\_date  
    reject\_date := null  
    requested\_amt := null

else

    idis\_disposition\_code := 5 (indicating "rejected")  
    redemption\_amt := null  
    cleared\_date := null  
    reject\_date := bank\_data\_rec.process\_date  
    requested\_amt := bank\_data\_rec.amount / 100

end if

process\_date := bank\_data\_rec.process\_date  
obligated\_amt := null  
bd\_bank\_disposition\_code := bank\_data\_rec.accept\_code

If i\_food\_instruments.idis\_disposition\_code = '3',  
    Generate l\_fiscal\_year and l\_fiscal\_month from void\_date

else

    If idis\_disposition\_code = '3' (redeemed)  
        Generate l\_fiscal\_year and l\_fiscal\_month from redeemed\_date

if `i_food_instruments.idis_disposition_code = '4'` (redeemed), perform five steps:

1. Reduce obligations:

Get `f_obligations` record for the `i_food_instruments.ifit_fi_type`,  
`i_food_instrument_types.ou_seq_id` and  
`f_obligations.cat_category_code` equal to the  
category from the client using the `fi`,  
where the `FI` prescription matches the `FI`  
and the `f_obligations` fiscal year and  
month.

Update `f_obligations` with `fi_oblig_count = fi_oblig_count - 1`

if `i_food_instruments.est_reb_value` is not null,

Update `f_obligations` with `reb_fi_counter = reb_fi_counter + 1`

2. Increase Outlays:

Get `f_outlays` record that matches the `f_obligations` record for `l_fiscal_year` and  
`l_fiscal_month`

if not found,

Create a record to match the `f_obligations` with `l_fiscal_year` and  
`l_fiscal_month`

Get the newly created record

Update `f_outlays`, incrementing `redeemed_fi_count` and adding  
`i_food_instrument.redemption_amount` to `redeemed_fi_value`

3. Increase `f_reb_details` quantity and value:

Search for `f_reb_contracts` records for the `i_food_instrument_foods` that make up  
the `i_food_instruments.ifit_fi_type_code` where the `cleared_date` is  
between the `f_reb_contracts.start_date`, `end_date`

Set local variable `l_total_rebates = sum(f_reb_contracts.rebate_per_unit *  
i_food_instrument_foods.quantity)` where the `cleared_date` is between the  
`f_reb_contracts.start_date`, `end_date`

If `l_total_rebates` is not null then (rebate information for the food instrument was  
found)

Find the `f_reb_details` records

where `f_reb_contracts.if_food_id` matches the  
`i_food_instrument_foods.if_food_id`

for the `i_food_instruments.ifit_fi_type_code` and the  
`i_food_instruments.cleared_date` is between  
`f_reb_contracts.start_date` and `end_date`

Set local variables:

`l_unit_quantity = f_reb_details.unit_quantity +  
f_reb_details.quantity`

`l_value_for_units = f_reb_details.value_for_units +`

```

f_reb_details.quantity)
                                (f_reb_details.unit_quantity *
                                if f_reb_details.replacement_flag is null,
                                create the f_reb_details record:
                                fol_activity_year = l_fiscal_year
                                fol_activity_month = l_fiscal_month
                                if i_food_instrument.issue_method = 'R'
                                    Replacement_flag = 'Y'
                                Else
                                    Replacement_flag = 'N'
                                Else
                                    Update the f_reb_details record with:
                                    Unit_quantity = l_unit_quantity
                                    Value_for_units = l_value_for_units
                                    if i_food_instrument.issue_method = 'R'
                                        Replacement_flag = 'Y'
                                    Else
                                        Replacement_flag = 'N'
                                End;

```

#### 4. Delete reject reasons for revalidated FI's

If the bank\_data\_rec.accept\_code = 'U' (record was revalidated by the bank),  
delete from i\_fi\_reject\_reasons for the bank\_data\_rec.serial\_number

#### 5. Insert a record into i\_bank\_data with the i\_food\_instruments data and set

Post\_code = 'Y'  
Send\_code = 'L'  
Message = 'This check received from bank'

If the record is missing issuance, then create the I\_Food\_Instrument record:

Insert i\_food\_instruments, setting the following values:

serial\_number = bank\_data\_rec.serial\_number  
ou\_seq\_id = o\_organizational\_units.seq\_id for the State Agency  
compliance\_buy\_flag = 'N'  
idis\_disposition\_code = decode (bank\_data\_rec.status, 'R',5,'P',4,'M',4,'S',4)  
bd\_bank\_disposition\_code = bank\_data\_rec.accept\_code  
ven\_id = v\_vendors.id where bank\_data\_rec.vendor\_number =

v\_vendors.vendor\_id

if bank\_data\_rec.status = 'R'

```

        requested_amt := bank_data_rec.amount / 100
        reject_date := bank_data_rec.process_date
        redemption_amt = null
        cleared_date = ' '
    else
        redemption_amt := bank_data_rec.amount / 100
        cleared_date = bank_data_rec.process_date
        requested_amt = null
        reject_date = null

        missing_issuance_flag = 'N'
        process_date = bank_data_rec.process_date

```

Insert a record into i\_bank\_datas with the i\_food\_instruments data and set

```

    Post_code = 'Y'
    Send_code = 'N'
    Message = 'This is missing issuance check'

```

Add rejection reasons:

```

        if length(bank_data_rec.rejection_code) <> 0
            For each character in the rejection_code (for i = 1 to
length(bank_data_rec.rejection_code))
                insert into i_fi_reject_reasons, setting:
                    ifi_serial_number = bank_data_rec.serial_number
                    irr_reject_reason_code =
substr(bank_data_rec.rejection_code,i,1)
            End;

```

### 1.23.3.3 Outputs

I\_Food\_Instruments  
 I\_Bank\_Reports  
 F\_Obligations  
 F\_Outlays  
 F\_Reb\_Details  
 I\_FI\_Reject\_Reasons

Failure messages for:

- Obligations table
- deleting reject reasons records
- creating records in I\_FOOD\_INSTRUMENTS
- creating I\_FI\_REJECT\_REASONS

### 1.23.4 EC4.SQL

**Overview**

Runs SQL scripts for EC4.BAT.

**1.23.4.1 Input**

eod\_controls  
env\_variables

**1.23.4.2 Selection**

env\_variables  
where code = 'EOD\_BY'

**1.23.4.3 Processing**

update eod\_controls, setting TO\_DATE to the system date  
update env\_variables, setting env\_value to 'EC4' where code = 'EOD\_BY'

run SQL script EC4\_CASHFLOW\_UPD  
run SQL script EC4\_MON\_END  
run SQL script EC4\_VEN\_PEER\_FI  
run SQL script EC4\_VEN\_LOC\_AGCY\_FI  
run SQL script EC4\_VEN\_PRICE

update eod\_controls, setting FROM\_DATE = TO\_DATE, GOOD\_PROC\_DATE = TO\_DATE

**1.23.4.4 Outputs**

end\_controls  
env\_variables  
Log file: EC4\_SQL.LOG

**1.24** Create date of current food funds available from all sources as well as current food obligations and expenditures to establish current cash position

**1.24.1 EC4\_CASHFLOW\_UPD.SQL*****Overview***

This EOD script is used to update summary information in the caseloads table for each distinct month/year where data has been changed in obligations and outlays.

#### 1.24.1.1 Inputs

ENV\_VARIABLES  
EOD\_CONTROLS  
F\_OBLIGATIONS  
F\_ANNUAL\_FACTORS  
F\_OUTLAYS  
I\_FOOD\_INSTRUMENTS  
C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
C\_CERTIFICATIONS  
F\_REB\_DETAILS  
F\_BUDGETS

#### 1.24.1.2 Processing

The F\_CASHFLOWS table is populated by summarizing information from other Financial tables for the F\_CASHFLOWS.ACTIVITY\_MONTH and F\_CASHFLOWS.ACTIVITY\_YEAR. The F\_CASHFLOWS.A\_NADJ\_\* represents unadjusted obligations and outlays, and is used when the F\_ANNUAL\_FACTORS.REDEEM\_RATE\_METHOD = '3'. The F\_CASHFLOWS.A\_NADJ\_PRERB\_OBLIG\_FROM is set to the  $\text{SUM}(\text{F\_OBLIGATIONS.FI\_OBLIG\_COUNT} \times \text{F\_OBLIGATIONS.EST\_PER\_FI\_VALUE})$  for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON0 and F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON1 are both set to 0. F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON2 is set to F\_CASHFLOWS.A\_NADJ\_PRERB\_OBLIG\_FROM. If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON0 is set to 0.  $(\text{F\_BUDGETS.EST\_FI\_MON1\_FACTOR} + \text{F\_BUDGETS.EST\_FI\_MON2\_FACTOR})$  is stored as a variable V\_FACTORS\_TOTAL. F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON1 is set to  $(\text{F\_CASHFLOWS.A\_NADJ\_PRERB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON1\_FACTOR} / \text{V\_FACTORS\_TOTAL})$ . F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON2 is set to  $(\text{F\_CASHFLOWS.A\_NADJ\_PRERB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON2\_FACTOR} / \text{V\_FACTORS\_TOTAL})$ . If the Fiscal Month is greater than or equal to the (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON0 is set to  $(\text{F\_CASHFLOWS.A\_NADJ\_PRERB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON0\_FACTOR})$ . F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON1 is set to  $(\text{F\_CASHFLOWS.A\_NADJ\_PRERB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON1\_FACTOR})$ . F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON2 is set to  $(\text{F\_CASHFLOWS.A\_NADJ\_PRERB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON2\_FACTOR})$ .

The F\_CASHFLOWS.A\_NADJ\_REB\_OBLIG\_FROM is set to the  $\text{SUM}(\text{F\_OBLIGATIONS.EST\_REB\_VALUE})$  for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON0 and F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON1 are both set to 0. F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON2 is set to F\_CASHFLOWS.A\_NADJ\_REB\_OBLIG\_FROM. If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON0 is set to 0.  $(\text{F\_BUDGETS.EST\_FI\_MON1\_FACTOR} + \text{F\_BUDGETS.EST\_FI\_MON2\_FACTOR})$  is stored as a variable V\_FACTORS\_TOTAL. F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON1 is set to  $(\text{F\_CASHFLOWS.A\_NADJ\_REB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON1\_FACTOR} / \text{V\_FACTORS\_TOTAL})$ . F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON2 is set to  $(\text{F\_CASHFLOWS.A\_NADJ\_REB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON2\_FACTOR} / \text{V\_FACTORS\_TOTAL})$ . If the Fiscal Month is greater than or equal to the (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON0 is set to  $(\text{F\_CASHFLOWS.A\_NADJ\_REB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON0\_FACTOR})$ . F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON1 is set to  $(\text{F\_CASHFLOWS.A\_NADJ\_REB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON1\_FACTOR})$ . F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON2 is set to  $(\text{F\_CASHFLOWS.A\_NADJ\_REB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON2\_FACTOR})$ .

The F\_CASHFLOWS.A\_FADJ\_\* represent obligations adjusted for redemption probability by FI Type, and is used when the F\_ANNUAL\_FACTORS.REDEEM\_RATE\_METHOD = '1'. The F\_CASHFLOWS.A\_FADJ\_PRERB\_OBLIG\_FROM is set to the  $\text{SUM}(\text{F\_OBLIGATIONS.FI\_OBLIG\_COUNT} \times \text{F\_OBLIGATIONS.ADJ\_PER\_FI\_VALUE})$  for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON0 and F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON1 are both set to 0. F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON2 is set to F\_CASHFLOWS.A\_FADJ\_PRERB\_OBLIG\_FROM. If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON0 is set to 0.  $(\text{F\_BUDGETS.EST\_FI\_MON1\_FACTOR} + \text{F\_BUDGETS.EST\_FI\_MON2\_FACTOR})$  is stored as a variable V\_FACTORS\_TOTAL. F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON1 is set to  $(\text{F\_CASHFLOWS.A\_FADJ\_PRERB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON1\_FACTOR} / \text{V\_FACTORS\_TOTAL})$ . F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON2 is set to  $(\text{F\_CASHFLOWS.A\_FADJ\_PRERB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON2\_FACTOR} / \text{V\_FACTORS\_TOTAL})$ . If the Fiscal Month is greater than or equal to the (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON0 is set to  $(\text{F\_CASHFLOWS.A\_FADJ\_PRERB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON0\_FACTOR})$ . F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON1 is set to  $(\text{F\_CASHFLOWS.A\_FADJ\_PRERB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON1\_FACTOR})$ . F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON2 is set to  $(\text{F\_CASHFLOWS.A\_FADJ\_PRERB\_OBLIG\_FROM} \times \text{F\_BUDGETS.EST\_FI\_MON2\_FACTOR})$ .

The F\_CASHFLOWS.A\_FADJ\_REB\_OBLIG\_FROM is set to the  $\text{SUM}(\text{F\_OBLIGATIONS.EST\_REB\_VALUE} \times \text{F\_OBLIGATIONS.EST\_REDEEM\_FACTOR})$  for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON0 and

$F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON1$  are both set to 0.  
 $F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON2$  is set to  
 $F\_CASHFLOWS.A\_FADJ\_REB\_OBLIG\_FROM$ . If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then  $F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON0$  is set to 0.  $(F\_BUDGETS.EST\_FI\_MON1\_FACTOR + F\_BUDGETS.EST\_FI\_MON2\_FACTOR)$  is stored as a variable  $V\_FACTORS\_TOTAL$ .  $F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON1$  is set to  $(F\_CASHFLOWS.A\_FADJ\_REB\_OBLIG\_FROM \times F\_BUDGETS.EST\_FI\_MON1\_FACTOR / V\_FACTORS\_TOTAL)$ .  $F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON2$  is set to  $(F\_CASHFLOWS.A\_FADJ\_REB\_OBLIG\_FROM \times F\_BUDGETS.EST\_FI\_MON2\_FACTOR / V\_FACTORS\_TOTAL)$ . If the Fiscal Month is greater than or equal to the (Fiscal Month of SYSDATE), then  $F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON0$  is set to  $(F\_CASHFLOWS.A\_FADJ\_REB\_OBLIG\_FROM \times F\_BUDGETS.EST\_FI\_MON0\_FACTOR)$ .  $F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON1$  is set to  $(F\_CASHFLOWS.A\_FADJ\_REB\_OBLIG\_FROM \times F\_BUDGETS.EST\_FI\_MON1\_FACTOR)$ .  $F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON2$  is set to  $(F\_CASHFLOWS.A\_FADJ\_REB\_OBLIG\_FROM \times F\_BUDGETS.EST\_FI\_MON2\_FACTOR)$ .

The  $F\_CASHFLOWS.A\_TADJ\_*$  represent obligations adjusted by a typical redemption rate, and is used when the  $F\_ANNUAL\_FACTORS.REDEEM\_RATE\_METHOD = '2'$ . The  $F\_CASHFLOWS.A\_TADJ\_PRERB\_OBLIG\_FROM$  is set to the  $(SUM(F\_OBLIGATIONS.FI\_OBLIG\_COUNT \times F\_OBLIGATIONS.EST\_PER\_FI\_VALUE) \times SUM(F\_OBLIGATIONS.FI\_OBLIG\_COUNT \times F\_OBLIGATIONS.EST\_REDEEM\_FACTOR) / SUM(F\_OBLIGATIONS.FI\_OBLIG\_COUNT))$  for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then  $F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON0$  and  $F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON1$  are both set to 0.  $F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON2$  is set to  $F\_CASHFLOWS.A\_TADJ\_PRERB\_OBLIG\_FROM$ . If the Fiscal Month is 1 month before the (Fiscal Month of SYSDATE), then  $F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON0$  is set to 0.  $(F\_BUDGETS.EST\_FI\_MON1\_FACTOR + F\_BUDGETS.EST\_FI\_MON2\_FACTOR)$  is stored as a variable  $V\_FACTORS\_TOTAL$ .  $F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON1$  is set to  $(F\_CASHFLOWS.A\_TADJ\_PRERB\_OBLIG\_FROM \times F\_BUDGETS.EST\_FI\_MON1\_FACTOR / V\_FACTORS\_TOTAL)$ .  $F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON2$  is set to  $(F\_CASHFLOWS.A\_TADJ\_PRERB\_OBLIG\_FROM \times F\_BUDGETS.EST\_FI\_MON2\_FACTOR / V\_FACTORS\_TOTAL)$ . If the Fiscal Month is greater than or equal to the (Fiscal Month of SYSDATE), then  $F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON0$  is set to  $(F\_CASHFLOWS.A\_TADJ\_PRERB\_OBLIG\_FROM \times F\_BUDGETS.EST\_FI\_MON0\_FACTOR)$ .  $F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON1$  is set to  $(F\_CASHFLOWS.A\_TADJ\_PRERB\_OBLIG\_FROM \times F\_BUDGETS.EST\_FI\_MON1\_FACTOR)$ .  $F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON2$  is set to  $(F\_CASHFLOWS.A\_TADJ\_PRERB\_OBLIG\_FROM \times F\_BUDGETS.EST\_FI\_MON2\_FACTOR)$ .

The  $F\_CASHFLOWS.A\_TADJ\_REB\_OBLIG\_FROM$  is set to the  $((SUM(F\_OBLIGATIONS.EST\_REB\_VALUE) / SUM(F\_OBLIGATIONS.REB\_FI\_COUNTER)) \times (SUM(F\_OBLIGATIONS.FI\_OBLIG\_COUNT \times F\_OBLIGATIONS.EST\_REDEEM\_RATE) / SUM(F\_OBLIGATIONS.FI\_OBLIG\_COUNT)))$  for the Fiscal Month and Year. If the Fiscal Month is 2 or more months before the (Fiscal Month of SYSDATE), then

F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON0 and  
 F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON1 are both set to 0.  
 F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON2 is set to  
 F\_CASHFLOWS.A\_TADJ\_REB\_OBLIG\_FROM. If the Fiscal Month is 1 month before the  
 (Fiscal Month of SYSDATE), then F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON0 is set to 0.  
 (F\_BUDGETS.EST\_FI\_MON1\_FACTOR + F\_BUDGETS.EST\_FI\_MON2\_FACTOR) is stored  
 as a variable V\_FACTORS\_TOTAL. F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON1 is set to  
 (F\_CASHFLOWS.A\_TADJ\_REB\_OBLIG\_FROM x F\_BUDGETS.EST\_FI\_MON1\_FACTOR /  
 V\_FACTORS\_TOTAL). F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON2 is set to  
 (F\_CASHFLOWS.A\_TADJ\_REB\_OBLIG\_FROM x F\_BUDGETS.EST\_FI\_MON2\_FACTOR /  
 V\_FACTORS\_TOTAL). If the Fiscal Month is greater then or equal to the (Fiscal Month of  
 SYSDATE), then F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON0 is set to  
 (F\_CASHFLOWS.A\_TADJ\_REB\_OBLIG\_FROM x F\_BUDGETS.EST\_FI\_MON0\_FACTOR).  
 F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON1 is set to  
 (F\_CASHFLOWS.A\_TADJ\_REB\_OBLIG\_FROM x F\_BUDGETS.EST\_FI\_MON1\_FACTOR).  
 F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON2 is set to  
 (F\_CASHFLOWS.A\_TADJ\_REB\_OBLIG\_FROM x F\_BUDGETS.EST\_FI\_MON2\_FACTOR).

F\_CASHFLOWS.A\_NADJ\_OBLIG\_IN\_ACTIV\_MON is set to the sum of  
 F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON0 +  
 F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON1 +  
 F\_CASHFLOWS.A\_NADJ\_OBLIG\_FOR\_MON2.  
 F\_CASHFLOWS.A\_FADJ\_OBLIG\_IN\_ACTIV\_MON is set to the sum of  
 F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON0 +  
 F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON1 +  
 F\_CASHFLOWS.A\_FADJ\_OBLIG\_FOR\_MON2.  
 F\_CASHFLOWS.A\_TADJ\_OBLIG\_IN\_ACTIV\_MON is set to the sum of  
 F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON0 +  
 F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON1 +  
 F\_CASHFLOWS.A\_TADJ\_OBLIG\_FOR\_MON2.

F\_CASHFLOWS.A\_NADJ\_REB\_IN\_ACTIV\_MON is set to the sum of  
 F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON0 +  
 F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON1 +  
 F\_CASHFLOWS.A\_NADJ\_REB\_FOR\_MON2.  
 F\_CASHFLOWS.A\_FADJ\_REB\_IN\_ACTIV\_MON is set to the sum of  
 F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON0 +  
 F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON1 +  
 F\_CASHFLOWS.A\_FADJ\_REB\_FOR\_MON2.  
 F\_CASHFLOWS.A\_TADJ\_REB\_IN\_ACTIV\_MON is set to the sum of  
 F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON0 +  
 F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON1 +  
 F\_CASHFLOWS.A\_TADJ\_REB\_FOR\_MON2.

The actual pre-rebate outlays are stored in F\_CASHFLOWS.A\_PRERB\_OUTLAY\_FROM,  
 which is set to SUM(F\_OUTLAYS.REDEEMED\_FI\_VALUE) for the Fiscal Month and Year.  
 F\_CASHFLOWS.A\_OUTLAY\_FOR\_MON0 is set to  
 SUM(F\_OUTLAYS.REDEEMED\_FI\_VALUE) for the Fiscal Month and Year where the  
 F\_OUTLAYS.ACTIVITY\_MONTH and F\_OUTLAYS.ACTIVITY\_YEAR are equal to the  
 Fiscal Month and Year. F\_CASHFLOWS.A\_OUTLAY\_FOR\_MON1 is set to  
 SUM(F\_OUTLAYS.REDEEMED\_FI\_VALUE) for the Fiscal Month and Year where the

F\_OUTLAYS.ACTIVITY\_MONTH and F\_OUTLAYS.ACTIVITY\_YEAR are equal to the previous Fiscal Month and Year. F\_CASHFLOWS.A\_OUTLAY\_FOR\_MON2 is set to SUM(F\_OUTLAYS.REDEEMED\_FI\_VALUE) for the Fiscal Month and Year where the F\_OUTLAYS.ACTIVITY\_MONTH and F\_OUTLAYS.ACTIVITY\_YEAR are equal to the Fiscal Month and Year of two months ago. F\_CASHFLOWS.A\_OUTLAY\_IN\_ACTIV\_MON is set to the SUM(F\_OUTLAYS.REDEEMED\_FI\_VALUE) for all Fiscal Months and Years where F\_OUTLAYS.ACTIVITY\_MONTH and F\_OUTLAYS.ACTIVITY\_YEAR are equal to the F\_CASHFLOWS.ACTIVITY\_MONTH and F\_CASHFLOWS.ACTIVITY\_YEAR.

The actual billable rebates are stored in F\_CASHFLOWS.A\_OUTLAY\_REB\_FROM, which is set to SUM(F\_REB\_DETAILS.VALUE\_FOR\_UNITS) for the Fiscal Month and Year. F\_CASHFLOWS.A\_OUTLAY\_REB\_FOR\_MON0 is set to SUM(F\_REB\_DETAILS.VALUE\_FOR\_UNITS) for the Fiscal Month and Year where the F\_REB\_DETAILS.FOL\_ACTIVITY\_MONTH and F\_REB\_DETAILS.FOL\_ACTIVITY\_YEAR are equal to the Fiscal Month and Year. F\_CASHFLOWS.A\_OUTLAY\_REB\_FOR\_MON1 is set to SUM(F\_REB\_DETAILS.VALUE\_FOR\_UNITS) for the Fiscal Month and Year where the F\_REB\_DETAILS.FOL\_ACTIVITY\_MONTH and F\_REB\_DETAILS.FOL\_ACTIVITY\_YEAR are equal to the previous Fiscal Month and Year. F\_CASHFLOWS.A\_OUTLAY\_REB\_FOR\_MON2 is set to SUM(F\_REB\_DETAILS.VALUE\_FOR\_UNITS) for the Fiscal Month and Year where the F\_REB\_DETAILS.FOL\_ACTIVITY\_MONTH and F\_REB\_DETAILS.FOL\_ACTIVITY\_YEAR are equal to the Fiscal Month and Year of two months ago. F\_CASHFLOWS.A\_OUTLAY\_REB\_IN\_ACTIV\_MON is set to the SUM(F\_REB\_DETAILS.VALUE\_FOR\_UNITS) for all Fiscal Months and Years where F\_REB\_DETAILS.FOL\_ACTIVITY\_MONTH and F\_REB\_DETAILS.FOL\_ACTIVITY\_YEAR are equal to the F\_CASHFLOWS.ACTIVITY\_MONTH and F\_CASHFLOWS.ACTIVITY\_YEAR.

Both F\_BUDGETS.ACT\_PARTICIP and F\_CASHFLOWS.A\_REDEEM\_PARTICIP are generated from SUM(F\_CASELOADS.CLIENT\_COUNT) for the Fiscal Month and Year where F\_CASELOAD\_TYPES.DESCRPTION = 'PARTICIPANT'. The F\_MONTHLY\_CATEGORY\_FACTORS.ACT\_PARTICIPANT is populated with the SUM(F\_CASELOADS.CLIENT\_COUNT) for the Fiscal Month and Year, and Category where F\_CASELOAD\_TYPES.DESCRPTION = 'PARTICIPANT'.

### 1.24.1.3 Outputs

States when processing begins and ends for obligations and outlays.

F\_Budgets  
F\_Cashflows  
F\_Monthly\_Category\_Factors

**1.25** Financial (end of month) - Populate F\_INCOME to POVERTY table. These values are used in populating of the caseload table.

### 1.25.1 EC4\_MON\_END.SQL

#### *Overview*

This end of day script is used to perform population of income and poverty tables at the end of month. It will insert the junction of the C\_INCOME\_LEVELS and the F\_POVERTY\_LEVELS tables into the F\_INCOMES\_TO\_POVERTIES table. It will update the caseload type details table with latest available data and create a new set of records in the caseloads table for the following month.

#### 1.25.1.1 Inputs

eod\_controls  
c\_income\_levels  
f\_poverty\_levels  
f\_poverty\_basis

#### 1.25.1.2 Processing

If the SYSDATE is the after the last day of the month EOD\_CONTROLS.IPL\_DATE, then  
Select the max(F\_POVERTY\_LEVELS.BEGIN\_DATE), the  
max(F\_POVERTY\_BASES.BEGIN\_DATE), and the  
max(C\_INCOME\_LEVELS.BEGIN\_DATE) that is on or before the first date of the  
month after the EOD\_CONTROLS.IPL\_DATE.

If any of these three dates are the first date of the month after the  
EOD\_CONTROLS.IPL\_DATE, then

For all C\_INCOME\_LEVELS records for the Begin Date,

Insert into F\_INCOMES\_TO\_POVERTIES

The foreign keys are derived from the primary keys of  
the F\_POVERTY\_LEVELS and C\_INCOME\_LEVELS  
records

The Begin Date is derived from the  
C\_INCOME\_LEVELS.BEGIN\_DATE

The Income High is derived from the  
F\_POVERTY\_LEVELS.HIGH\_PERCENTAGE \*  
C\_INCOME\_LEVELS.INCOME\_HIGH /  
(F\_POVERTY\_BASES.POVERTY\_BASIS \* 100)

The Income Low is derived from the  
F\_POVERTY\_LEVELS.LOW\_PERCENTAGE \*

$$\frac{C\_INCOME\_LEVELS.INCOME\_HIGH}{(F\_POVERTY\_BASES.POVERTY\_BASIS * 100)}$$

Update EOD\_CONTROLS

Set the IPL\_DATE to the month after the current EOD\_CONTROLS.IPL\_DATE

Set the MONTHLY\_SW to 'Y'

If the max(F\_INCOMES\_TO\_POVERTIES.BEGIN\_DATE) equals  
EOD\_CONTROLS.IPL\_DATE, then

For F\_INCOMES\_TO\_POVERTIES records where the Begin Date is in the  
month of EOD\_CONTROLS.IPL\_DATE

Update F\_CASELOAD\_TYPE\_DETAILS where the Family Size,  
Income, and Fiscal Year and Month are equal to the  
F\_INCOMES\_TO\_POVERTIES record

Set the Poverty Level foreign key to

F\_INCOMES\_TO\_POVERTIES.FPL\_ID

For caseload processing please refer to Section 1.38

### 1.25.1.3 Outputs

eod\_controls

f\_incomes\_to\_poverties

**1.26** Vendor (end of month) calculate and update peer group averages**1.26.1** EC4\_VEN\_PEER\_FI.SQL**Overview**

Updates or inserts into peer group averages for the food instruments disposition code of '4' (Redeemed). The following data elements are updated:

- \* Total redemption amount
- \* Average participants
- \* Average redemption amount
- \* Maximum redemption amount
- \* Total redeemed FI's
- \* Survey flag
- \* Date modified
- \* Modified by

**1.26.1.1** Inputs

V\_VENDORS  
 I\_FOOD\_INSTRUMENTS  
 V\_PEER\_GROUP\_AVGS  
 V\_PEER\_GROUPS

**1.26.1.2** Selection

To calculate V\_PEER\_GROUP\_AVGS.AVG\_PARTICIPANTS:

I\_FOOD\_INSTRUMENTS where I\_FOOD\_INSTRUMENTS.CLEARED DATE is in the past three months, retrieved separately  
 and I\_FOOD\_INSTRUMENTS.IDIS\_DISPOSITION\_CODE = 4 (Redeemed)  
 for the Vendor and Peer Group

For all other calculations:

I\_FOOD\_INSTRUMENTS where I\_FOOD\_INSTRUMENTS.CLEARED\_DATE is within the last 30 calendar days  
 and I\_FOOD\_INSTRUMENTS.IDIS\_DISPOSITION\_CODE = 4 (Redeemed)  
 for the Vendor and Peer Group  
 grouped by V\_VENDORS.PEE\_PEER\_GROUP,  
 I\_FOOD\_INSTRUMENTS.IFIT\_FI\_TYPE\_CODE

**1.26.1.3 Processing**

```

end_date := today's date
start_date := end_date minus 30 days
  For I_FOOD_INSTRUMENTS where cleared_date is between start_date and end_date and
    idis_disposition_code = 4 (Redeemed)
  grouped by V_VENDORS.PEE_PEER_GROUP,
    I_FOOD_INSTRUMENTS.IFIT_FI_TYPE_CODE

  fi_count := count(i_food_instruments.serial_number)
  total_red_amt := sum(i_food_instruments.redemption_amt)

-- if there are not three data points, do not recalculate:
  if count(I_FOOD_INSTRUMENTS.SERIAL_NUMBER) for the Peer Group and FI Type is < 3,
    set the AVG_REDEMPTION_AMT to null
    and set I_FOOD_INSTRUMENTS.RECALCULATE_FAILED = 'Y'

-- if recalculating would change the average by more than 15%, do not recalculate:
  new_avg_redemption_amt := total_red_amt / fi_count

  Else if abs ( (i_food_instruments.avg_redemption_amt - new_avg_redemption_amt) /
    i_food_instruments.avg_redemption_amt) > .15
    set the AVG_REDEMPTION_AMT to null
    and set I_FOOD_INSTRUMENTS.RECALCULATE_FAILED = 'Y'

-- otherwise, go ahead:
  Else
    set I_FOOD_INSTRUMENTS.RECALCULATE_FAILED = 'N'

-- count the average number of participants for the past three months
  For I_FOOD_INSTRUMENTS where I_FOOD_INSTRUMENTS.CLEARED DATE is
  in the past three months, retrieved separately by month, and
  I_FOOD_INSTRUMENTS.IDIS_DISPOSITION_CODE = 4
  for the Vendor and Peer Group

    client_count := distinct count(I_FOOD_INSTRUMENTS.CC_CLIENT_ID),
      averaged for the cleared_date in the past three months, for the Vendor
      and Peer Group

-- find the percent override for the FI type, in case a record must be added to v_peer_group_avgs:
  temp_percent := select default_percent from v_peer_groups for the peer group

-- update/insert into the v_peer_group_avgs table
  if v_peer_group_avgs record exists already for the Peer Group/FI,

    Update v_peer_group_avgs set
      total_redemption_amt = total_red_amt,
      avg_participants = client_count,
      total_redeemed_fis = fi_count,
      avg_redemption_amt = total_red_amt / fi_count,

```

```
max_redemption_amt = ((Total_Red_Amt/FI_Count)*  
((100+PERCENT)/100)),  
update_survey_flag = 'Y'
```

Else

```
Insert into v_peer_group_avgs, setting  
total_redemption_amt = total_red_amt,  
avg_participants = client_count,  
total_redeemed_fis = fi_count,  
avg_redemption_amt = total_red_amt / fi_count,  
percent = temp_percent,  
max_redemption_amt = ((Total_Red_Amt/FI_Count)*  
((100+temp_percent)/100)),  
update_survey_flag = 'Y'
```

#### 1.26.1.4 Outputs

##### V\_PEER\_GROUP\_AVGS

Printed log of start, insert/update, and end messages:

```
dbms_output.put_line('Starting Peer Group Average EOD');  
dbms_output.put_line('Inserting Peer / FI :||Temp_Peer_Group_ID||' / '||Fi_Type);  
dbms_output.put_line('Updating Peer / FI :||Temp_Peer_Group_ID||' / '||Fi_Type);  
dbms_output.put_line('Peer Group Average Data Updated Successfully');
```

On failure:

```
dbms_output.put_line('Peer Group Average Table Update Failed');
```

See Appendix A for further information.

## 1.26.2 EC4\_VEN\_LOC\_AGCY\_FI

### *Overview*

Updates or inserts into the local agency averages for the food instruments disposition code of '4' (Redeemed). The following data elements are updated:

- \* Total redemption amount
- \* Average redemption amount
- \* Total redeemed FI's
- \* Survey flag
- \* Date modified
- \* Modified by

### 1.26.2.1 Input

I\_FOOD\_INSTRUMENTS  
O\_ORGANIZATIONAL\_UNITS  
V\_VENDORS

### 1.26.2.2 Selection

I\_FOOD\_INSTRUMENTS where I\_FOOD\_INSTRUMENTS.CLEARED\_DATE is within the  
last 30 calendar days  
and I\_FOOD\_INSTRUMENTS.IDIS\_DISPOSITION\_CODE = 4  
for the Vendor and Local Agency  
grouped by V\_VENDOR.OU\_SEQ\_ID\_PRI\_LA,  
I\_FOOD\_INSTRUMENTS.IFIT\_FI\_TYPE\_CODE

**1.26.2.3 Processing**

```

end_date := today's date
start_date := end_date minus 30 days
For I_FOOD_INSTRUMENTS where cleared_date is between start_date and end_date and
    idis_disposition_code = 4 (redeemed)
    grouped by the redeeming Vendor's local agency and the FI Type

    fi_count := count(i_food_instruments.serial_number)
    total_red_amt := sum(i_food_instruments.redemption_amt)

-- if there are not three data points, do not recalculate:
    if count(I_FOOD_INSTRUMENTS.SERIAL_NUMBER) for the Local Agency and FI Type is <
        3, set the AVG_REDEMPTION_AMT to null

-- if recalculating would change the average by more than 15%, do not recalculate:
    new_avg_redemption_amt := total_red_amt / fi_count

    ElseIf abs ((v_la_fi_avgs.avg_redemption_amt - new_avg_redemption_amt) /
        v_la_fi_avgs.avg_redemption_amt) > .15
        set the AVG_REDEMPTION_AMT to null

        -- otherwise, go ahead:
-- update/insert into the V_LA_FI_AVGS table
    else

        if V_LA_AVGS record exists already for the Local Agency/FI,

            Update V_LA_FI_AVGS set
                total_redemption_amt = total_red_amt,
                total_redeemed_fis = fi_count,
                avg_redemption_amt = total_red_amt / fi_count,
                update_survey_flag = 'Y'

        Else

            Insert into V_LA_FI_AVGS, setting
                total_redemption_amt = total_red_amt,
                total_redeemed_fis = fi_count,
                avg_redemption_amt = total_red_amt / fi_count,
                update_survey_flag = 'Y'

```

**1.26.2.4 Output**

```

V_LA_FI_AVGS
Printed log of start, insert/update, and end messages
    dbms_output.put_line('Starting LA Average EOD');

```

```
dbms_output.put_line('Inserting LA / FI :'||Temp_LA_ou_ID||' / '||Fi_Type);  
dbms_output.put_line('Updating Peer / FI :'||Temp_LA_ou_ID||' / '||Fi_Type);  
dbms_output.put_line('LA Average Data Updated Successfully');
```

On failure:

```
dbms_output.put_line('LA Average Table Update Failed');
```

See Appendix A for further information

## 1.27 Vendor (end of month) run and print analysis factor reports

### 1.27.1 EC4\_VEN\_PRICE.SQL

#### *Overview*

Vendors purchasing analysis and ranking reports are produced from this script. The FI cost for Vendors is calculated and stored, based on their Food Price Surveys, and stored in V\_FI\_PRICES. Vendors are ranked by average maximum price, redemption value, and peer group identification.

A recalculation for a v\_fi\_prices record is performed if:

- the user checked a new compare flag on the Vendor Food Price Survey, Price Analysis, Compare/Risk window, or
- the user changed a Vendor survey price that is a component of an FI that has the compare flag checked on the Vendor Food Price Survey, Price Analysis, or Compare/Risk window. This would be indicated by the update\_survey\_flag set to 'Y' on the v\_fi\_prices table.

#### 1.27.1.1 Inputs

i\_food\_instrument\_foods  
i\_food\_instruments  
i\_foods  
v\_fi\_prices  
v\_peer\_group\_avgs  
v\_prices  
v\_vendors

#### 1.27.1.2 Selection

#### 1.27.1.3 Processing

Create a temporary table Temp\_Ven\_Price\_Survey

Vendor ID  
FI Type  
Peer Group ID  
FI Count  
Avg Redemption Value  
Redemption Rank \*  
Minimum Price

Maximum Price \*  
Rank \*

Load the following data into the Temp\_Ven\_Price\_Survey table without creating duplicate records:

- a. Vendor/FI combinations that were checked on the Price Analysis Compare/Risk window since the last successful end of day run:

Select FI type from i\_food\_instrument\_type where compare\_flag = 'Y', for Vendors having any v\_surveys record with rpt\_rcvd\_date or rpt\_postmarked\_date not null, and rep\_req\_review = 'N'. Include the Peer Group ID for the Vendor.

- b. Vendors/FI combinations whose survey prices changed:

Select Vendor Id, FI Type from v\_fi\_prices where update\_survey\_flag = 'Y'. Include the Peer Group ID for the Vendor.

1. For each Vendor/FI combination on the Temp\_Ven\_Price\_Survey table, calculate and store the maximum price for the FI's, based on Vendors' Price Survey:

Set local variables:

d\_ifit\_fi\_type\_code (FI Type)  
d\_peek\_peer\_group\_id (peer group)

For each food in the FI Type, set local variables:

d\_if\_food\_id := if\_food\_id (Food ID)  
d\_quantity := quantity (of the food in the FI)  
d\_unit\_size := unit\_size (size of the food)  
d\_iuom\_unit\_of\_measure\_code := iuom\_unit\_of\_measure\_code  
If a record exists on i\_foods with if\_food\_id\_equiv = d\_if\_food\_id  
d\_other\_foods\_use\_as\_equiv := 'Y'

v\_cost := 0 -- cost of the d\_if\_food\_id from the Vendor's food price survey (v\_prices)

If d\_other\_foods\_use\_as\_equiv = 'Y' (value is generic, like "cheese")

For each food that has if\_food\_id\_equiv equal to d\_if\_food\_id, (like "cheddar")

t\_items = trunc(d\_unit\_size / unit\_size)-- finds how many items can be combined without exceeding the d\_unit\_size.

Compute t\_cost := v\_prices.max\_price for the food \* t\_items

If t\_cost > v\_cost  
v\_cost := t\_cost

Else (the food is an actual food like "Kix")

```
v_cost = v_prices.max_price * t_items
```

```
End if
```

```
Update temp_ven_price_survey set maximum_price = maximum_price +
(V_cost * quantity)
where ven_id = d_ven_id and fi_type = d_ifit-fi_type_code
```

Delete records on the temporary table Temp\_Ven\_Price\_Survey where the maximum\_price is zero/null.

2. Rank by maximum FI price from Price Survey and generate/store FI redemption information for the Vendor/FI Type combination:

For the unique peer group and FI type combinations from Temp\_Ven\_Price\_Survey, set local variables:

```
d_peek_peer_group := peer group
d_ifit-fi_type_code := FI type
```

```
d_rank := 1
```

Get the Vendor ID's for the peer group/FI type combination from Temp\_Ven\_Price\_Survey, in descending order of the vendors' maximum\_price for the FI type. Set local variable:

```
d_ven_d = vendor id
```

From the Vendor's redeemed FI's (idis\_disposition\_code = '4') for the FI Type and Peer Group, set local variables:

```
d_sum := sum(redemption_amt)
d_count := count(ifi.serial_number)
d_average := sum(redemption_amt)/count(ifi.serial_number)
```

```
Update temp_ven_price_survey set rank = d_rank,
redemption_volume = d_sum, fi_count = d_count,
avg_redemption_value = d_average
```

```
d_rank := d_rank + 1
```

3. Rank by average redeemed FI price for the Vendor

```
d_rank := 1
```

Get the Vendor ID's for the peer group/FI type combination from Temp\_Ven\_Price\_Survey, in descending order of the vendors' avg\_redemption\_value for the FI type

```
Update temp_ven_price_survey set redemption_rank = d_rank
```

```
d_rank := d_rank + 1
```

4. Insert/Update v\_fi\_prices (Vendor's FI cost from Price Survey) and v\_surveys (Vendor's Price Survey) tables

For all records in the temporary table Temp\_Ven\_Price\_Survey, set local variables:

```
d_ven_id := Ven_id  
d_ifit_fi_type_code := fi_type
```

Find the date of the most recent Price Survey for the Vendor: max(survey\_date)

```
If no price survey (v_surveys) record was found for the Vendor, Insert into v_surveys set  
    survey_date = EOD run date,  
    submitted = 'Y'
```

```
If no FI Price Survey (v_fi_prices) record was found for the Vendor/FI, insert into v_fi_prices  
    set fi_maximum_price = Temp_Ven_Price_Survey.maximum_price,  
    redemption_rank = Temp_Ven_Price_Survey.redemption_rank,  
    rank_by_max = Temp_Ven_Price_Survey.rank
```

```
else  
    update v_fi_prices  
    set fi_maximum_price = Temp_Ven_Price_Survey.maximum_price,  
    redemption_rank = Temp_Ven_Price_Survey.redemption_rank,  
    rank_by_max = Temp_Ven_Price_Survey.rank
```

(Redemption\_volume, fi\_count, and avg\_redemption\_value are not stored in v\_fi\_prices.)

Delete all v\_fi\_prices records where (date\_created is not the current date and date\_modified is not the current date.

Drop Temp\_Ven\_Price\_Survey

#### 1.27.1.4 Outputs

```
v_fi_prices  
v_surveys  
Temp_Ven_Price_Survey
```

### 1.27.2 EC4\_VEN\_RISK\_HISTORY.SQL

#### *Overview*

At the end of the fiscal year, a new V\_RISK\_LEVEL\_HISTORY record must be created for the Applicant/Vendor, containing the new fiscal year and the current V\_RISK\_LEVELS.RISK\_LEVEL\_ID value.

#### Input

v\_risk\_level\_histories

#### Selection

all v\_risk\_level\_histories records for the current fiscal year

#### Processing

If the date of the end-of-day is the last day of the fiscal year (September 30, or the most recent business day)

find all v\_risk\_level\_histories records with current fiscal year in  
v\_risk\_level\_histories.fy\_of\_risk,

if a record does not exist for the current fiscal year plus 1,

duplicate the record, setting v\_risk\_level\_histories.fy\_of\_risk to the current fiscal year,  
plus 1.

#### Outputs

v\_risk\_level\_histories

**1.28** Gather new/updated data to be sent to the agencies**1.28.1** EC4\_TRNC.SQL**Overview**

This script initializes the following tables:

- E\_EOD\_CONTROLS
- A\_EOD\_CONTROLS
- A\_STATE\_CLIENTS

**1.28.1.1** Inputs

N/A

**1.28.1.2** Selection

N/A

**1.28.1.3** Processing

```
TRUNCATE TABLE E_EOD_CONTROLS  
TRUNCATE TABLE A_EOD_CONTROLS  
TRUNCATE TABLE A_STATE_CLIENTS
```

**1.28.1.4** Outputs

EC4\_TRNC.LOG

### 1.28.2 EC4\_TAB\_DOWN.SQL

#### *Overview*

This script runs for each Local Agency.,

The primary purpose of this script is to propagate down to Local Agencies changes made to selected tables at the State level. Temporary tables are utilized to facilitate data recovery efforts should they be needed.

For each temporary table, the old data in the table is truncated and new rows and values are inserted where either the date modified or the date created column of the “source” table is after the last successful run date for End Of Day Processing. For some temporary tables, additional criteria is employed.

#### 1.28.2.1 Inputs

N/A

#### 1.28.2.2 Selection

For all tables in this script, inserts are based on the date\_created or date\_modified being after the last successful run date of EOD. Some tables contain other conditions listed below.

For A\_I\_FOOD INSTRUMENTS:

If the Process Date is not null, and the Food Instrument is not associated with a Compliance Buy. The date modified is after the last successful run date of EOD Processing, and the clinic is in the local agency for which the script is being run

For A\_I\_FI\_REJECT\_REASONS:

If the Date Modified or Date Created is after the last successful run date of EOD Processing and the Food Instrument was issued by a clinic in the Local Agency for which the script is being run

For A\_F\_CASELOADS:

If the Date Modified or Date Created is after the last successful run date of EOD Processing and the Clinic is in the Local Agency for which the script is being run

For A\_F\_CASE\_ASSIGNMENTS:

If the Date Modified or Date Created is after the last successful run date of EOD Processing and the Clinic is in the Local Agency for which the script is being run

For A\_DEL\_STATEMENTS:

If the Date Modified or Date Created is after the last successful run date of EOD Processing and target\_proc\_date is null and the originating agency is Central

For all other tables:

If the Date Modified or Date Created is after the last successful run date of EOD Processing

### 1.28.2.3 Processing

Truncate old data from temporary table.

Copy rows/columns into temporary table based on above selection criteria.

### 1.28.2.4 Outputs

1. A log file is generated for each script/Local Agency, i.e. 01\_EC4\_TAB.LOG, 02\_EC4\_TAB.LOG, etc. An example of this log file can be found in Appendix A.

2. The following temporary tables are created:

- A\_EOD\_CONTROLS
- A\_A\_ACTIVITIES
- A\_AAS\_CLASS\_CATEGORIES
- A\_AAS\_ITEMS
- A\_A\_APPT\_ITEMS
- A\_A\_ATTEND\_STATUSES
- A\_A\_OFFICE\_CLOSED
- A\_A\_SERVICES
- A\_I\_AGE\_RANGES
- A\_C\_ANSWERS
- A\_C\_ANSWER\_TYPES
- A\_C\_BLOODWORK\_TYPES
- A\_C\_BREAST\_PUMP\_REASONS
- A\_C\_BREAST\_PUMP\_TYPES
- A\_C\_CATEGORIES
- A\_C\_CAT\_BLOOD\_FACTORS
- A\_C\_CAT\_GOALS
- A\_C\_CAT\_NUTR\_EDS
- A\_C\_CAT\_REFERRALS
- A\_C\_COMMUNICATION\_TYPES
- A\_C\_CP\_MESSAGES
- A\_C\_DIAGNOSES
- A\_C\_DIETARY\_REQUIREMENTS
- A\_C\_DIET\_NUTRIENT\_TYPES
- A\_C\_DISABILITIES
- A\_C\_EDUCATION\_LEVELS
- A\_C\_ELEVATIONS
- A\_C\_ETHNIC\_GROUPS
- A\_C\_GOALS
- A\_C\_HC\_PAYEES
- A\_C\_HH\_QUESTIONS
- A\_C\_HH\_RESPONSES
- A\_C\_INCOME\_INTERVALS

A\_C\_INCOME\_LEVELS  
A\_F\_POVERTY\_BASES  
A\_C\_INCOME\_SOURCES  
A\_C\_INCOME\_VERIFICATIONS  
A\_C\_INFANT\_STATUSES  
A\_C\_LANGUAGES  
A\_C\_MARITAL\_STATUSES  
A\_C\_NCHS\_CLASSIFICATIONS  
A\_C\_NCHS\_TYPES  
A\_C\_NCHS\_DATA  
A\_C\_NO\_CONTACT\_REASONS  
A\_C\_NUTR\_ED\_MATERIALS  
A\_C\_NUTR\_ED\_TOPICS  
A\_C\_PICKUP\_INTERVALS  
A\_C\_PILOT\_QUESTIONS  
A\_C\_PILOT\_STUDIES  
A\_C\_PRIORITIES  
A\_C\_PROOF\_ADDRESSES  
A\_C\_PROOF\_IDENTITIES  
A\_C\_PS\_QUESTIONS  
A\_C\_RACES  
A\_C\_REASONS\_BF\_ENDED  
A\_C\_RESOLUTIONS  
A\_C\_RESOLVED\_CLIENTS  
A\_C\_RISK\_FACTORS  
A\_C\_RISK\_FACTOR\_DIAGNOSES  
A\_C\_RISK\_FACTOR\_TYPES  
A\_C\_RF\_GOALS  
A\_C\_RF\_NUTR\_EDS  
A\_C\_RF\_REFERRALS  
A\_C\_RF\_SERVICES  
A\_C\_HH\_RESPONSE\_RFS  
A\_C\_BMI\_DATA  
A\_C\_BMI\_ANTHROPOMETRIC  
A\_C\_BMI\_WEIGHT\_GAIN  
A\_C\_SCHEDULE\_DAYS  
A\_C\_SMOKING\_CHANGES  
A\_C\_SMOKING\_STAGES  
A\_C\_SOURCES\_HEALTH\_CARE  
A\_C\_SYMPTOMS  
A\_C\_TERM\_REASONS  
A\_C\_TOPICS  
A\_C\_VOTER\_REGISTRATIONS  
A\_C\_CAT\_BLOODWORKS  
A\_C\_DESIREABLE\_WEIGHTS  
PROMPT EA1 inserting :A\_C\_IMMUNIZATIONS\_NOT\_ASSESSED  
A\_F\_CONTROLS  
A\_F\_CASELOAD\_TYPES  
A\_F\_FUND\_SOURCES  
A\_F\_POVERTY\_LEVELS  
A\_F\_WAIT\_LIST\_RESPONSES

A\_I\_BANK\_DISPOSITIONS  
A\_I\_CATEGORY\_GROUPS  
A\_I\_PACKAGES  
A\_I\_PRODUCTS  
A\_I\_REJECT\_REASONS  
A\_I\_CONTAINERS  
A\_I\_DISPOSITIONS  
A\_I\_UNITS\_OF\_MEASURE  
A\_I\_VOID\_REASONS  
A\_I\_FOOD\_GROUPS  
A\_I\_FOODS  
A\_I\_FOOD\_DISTRIBUTIONS  
A\_I\_MAXIMUM\_FOODS  
A\_I\_FOOD\_PACKAGES  
A\_I\_CATEGORY\_GROUP\_PKGS  
A\_I\_FOOD\_PACKAGE\_FI\_TYPES  
A\_I\_FOOD\_PACKAGE\_FOODS  
A\_I\_PACKAGE\_DISTRIBUTIONS  
A\_I\_WS\_FOOD\_GROUPS  
A\_I\_FOOD\_INSTRUMENT\_TYPES  
A\_I\_FOOD\_INSTRUMENT\_FOODS  
A\_C\_FOOD\_PKG\_RISK\_FACTORS  
A\_O\_OUTREACH\_ORG\_TYPES  
A\_O\_OUTREACH\_COMM\_TYPES  
--A\_O\_OUTREACH\_ORGANIZATIONS  
--A\_O\_OUTREACH\_ORG\_PHONES  
--A\_O\_OUTREACH\_COMMS  
A\_O\_PROGRAMS  
A\_O\_PROGRAM\_DATES  
A\_O\_TITLE\_CATEGORIES  
A\_O\_STAFF\_TITLES  
A\_S\_CITIES  
A\_S\_CONTACT\_METHODS  
A\_S\_CONTACT\_TITLES  
A\_S\_COUNTIES  
A\_S\_PHONE\_TYPES  
A\_S\_STATES  
A\_S\_ZIPS  
A\_S\_GEO\_LOCATIONS  
A\_V\_ACTIVITY\_TYPES  
A\_V\_APPLICATION\_MILESTONES  
A\_V\_APPLICATION\_TYPES  
A\_V\_BANK\_BRANCHES  
A\_V\_CASE\_STATUSES  
A\_V\_COLLECTION\_TYPES  
A\_V\_COMMUNICATION\_TYPES  
A\_V\_COMPLAINT\_SOURCE\_TYPES  
A\_V\_COMPLAINT\_STATUSES  
A\_V\_COMPLAINT\_SUBJECTS  
A\_V\_COMPLIANCE\_CASE\_DESIGNATNS  
A\_V\_COMPLIANCE\_CASE\_TYPES

A\_V\_DELIVERY\_TYPES  
A\_V\_DENIAL\_REASONS  
A\_V\_DISQUAL\_REASONS  
A\_V\_FINDING\_CODES  
A\_V\_FSP\_REGIONAL\_OFFICES  
A\_V\_FSP\_VIOLATION\_CODES  
A\_V\_LEGAL\_FIRMS  
A\_V\_LEGAL\_REPRESENTATIVES  
A\_V\_MILESTONE\_TYPES  
A\_V\_OWNER\_TYPES  
A\_V\_PEER\_GROUPS  
A\_V\_RISK\_LEVELS  
A\_V\_SANCTION\_TYPES  
A\_V\_STATUSES  
A\_V\_SUSPENSION\_REASONS  
A\_V\_VIOLATION\_ACTION  
A\_V\_WHOLESALERS  
A\_V\_WIC\_CODES  
A\_V\_OWNERS  
A\_V\_VENDORS  
--A\_O\_STAFF\_MEMBERS  
A\_O\_ORGANIZATIONAL\_UNITS  
A\_O\_ANNUAL\_WIC\_COST\_SUMMARIES  
A\_F\_ANNUAL\_FACTORS  
A\_F\_BUDGETS  
A\_F\_CASE\_ASSIGNMENTS  
A\_F\_CASELOADS  
A\_F\_CASELOAD\_RESTRICTIONS  
A\_F\_MFR\_CONTACTS  
A\_F\_MANUFACTURERS  
A\_I\_FI\_INVENTORIES  
A\_I\_STOCK\_INVENTORIES  
A\_I\_FOOD\_INSTRUMENTS  
A\_I\_FI\_REJECT\_REASONS  
S\_PRINTER\_ADDRESSES  
A\_DEL\_STATEMENTS

### 1.28.3 EC4\_AGCY\_BEING\_PROCESSED.SQL

#### *Overview*

This script will indicate the agency for which data is updated on the State Database in the End of Day Process.

#### 1.28.3.1 Inputs

Agency ID

### 1.28.3.2 Selection

N/A

### 1.28.3.3 Processing

Update env\_variables

```
Set env_value = decode (agcy_id, '01', '2', '02', '6', '03', '17', '04', '27', '05', '33', '06', '37', '07', '41',  
'08', '58', '09', '65', '10', '73', '11', '93', '13', '107', '14', '119', '15', '297', '17', '130', '25', '27',  
'133', '28', '141', '29', '151', '30', '155', '32', 318)
```

```
Where code= 'AGCY_BEING_PROCESSED';
```

### 1.28.3.4 OUTPUTS

N/A

**1.29** Prepare archival retrieval data**1.29.1** EC1\_PRG\_ARCHV.SQL**Overview**

Purge archived client records marked with an archive flag of 'Y'. The deletes have been ordered from the lowest level (Level 6) child up to the parent level (Level 1). Purge the archived vendor records marked with an archive flag of 'Y'. The deletes have been ordered from the smallest level child up to the parent (v\_vendors).

**1.29.1.1** Inputs

s\_client\_archives sc  
s\_vendor\_archives s  
c\_clients

**1.29.1.2** Selection

Marks client records and vendor records that are to be deleted:  
WHERE S.ARCHIVE\_FLAG = 'Y'

**1.29.1.3** Processing

Any client record marked with an archive flag of 'Y' is deleted from the following tables where the client ID is equal to the S\_CLIENT\_ARCHIVES.CLIENT\_ID unless otherwise specified

## LEVEL 6:

No deletions made at this level.

## LEVEL 5:

A\_APPT\_TOPIC\_MATERIALS

## LEVEL 4:

A\_APPT\_TOPICS

A\_APPT\_APPT\_ITEMS

C\_CLIENT\_SVC\_NE\_MATERIALS

AAS\_APPTS\_CLIENTS

F\_WAIT\_LIST\_CONTACTS

C\_DIET\_NUTRIENTS

C\_INCOMES where the Economic Member Code

WHERE ceum\_economic\_member\_code in

```
(SELECT economic_member_code
FROM c_economic_unit_members
WHERE cih_income_history_id in
(SELECT income_history_id
FROM c_income_histories
WHERE cc_client_id = clid));
```

## LEVEL 3:

```
A_APPOINTMENTS
C_CLIENT_NE_TOPICS
C_WOMAN_MEDICALS
C_BLOODWORK_DATA
C_CERT_TERM_REASONS
C_B_N_HEALTHS
C_FOOD_PACKAGE_PRESCRIPTIONS
F_WAIT_LISTS
C_INFANT_CHILD_MEDICALS
C_DIETARY_ASSESSMENTS
C_HEALTH_RISK_FACTORS
C_I_C_HEALTHS
C_ECONOMIC_UNIT_MEMBERS
    WHERE cih_income_history_id in
    (SELECT income_history_id
    FROM c_income_histories
    WHERE cc_client_id = clid);
C_PEER_RBFENDS
C_CONTACTS
C_P_COUNS_NOTES
I_BF_VCHR_ITEMS
    (SELECT voucher_no from I_BF_PROMO_VCHRS
    where cc_client_id = clid);
```

## LEVEL 2:

```
I_BF_PROMO_VCHRS
C_WOMAN_MEDICALS
C_W_HEALTHS
C_INFANT_CHILD_MEDICALS
C_I_C_HEALTHS
C_BLOODWORK_DATA
C_PREV_FAMILIES
C_PREV_NAMES
C_ALLERGY_FOODS
C_MORE_ETHNIC_GROUPS
C_CLIENT_SERVICES
C_CERTIFICATIONS
C_TRANSFER_HISTORIES
C_IMMUNIZATIONS
C_CLIENT_REFERRALS
C_CLIENT_COMMUNICATIONS
C_RESOLUTIONS
```

C\_CLIENT\_NOTES  
C\_CLIENT\_PROGS  
C\_CLIENT\_GROUPS  
C\_INCOME\_HISTORIES  
C\_INFANT\_DATA  
C\_CLIENT\_GOALS  
C\_CERT\_PEER\_COUNSELS

LEVEL 1:

C\_CLIENTS  
C\_INCOMES  
C\_ECONOMIC\_UNIT\_MEMBERS  
C\_INCOME\_HISTORIES

The S\_CLIENT\_ARCHIVES table is then updated by setting the ARCHIVE\_FLAG = 'N'.

Insert record in R\_ARCHIVED\_CLIENTS.

Any vendor record marked with an archive flag of 'Y' is deleted from the following tables where the vendor ID is equal to the S\_VENDOR\_ARCHIVES.ID

LEVEL 6:

F\_PURCHASE\_ORDER\_LINE\_ITEMS  
where (fpoi\_fpo\_po\_number, fpoi\_invoice\_date, fpoi\_invoice\_number ) in  
(select fpo\_po\_number, invoice\_date, invoice\_number  
from f\_purchase\_order\_invoices  
where ifi\_serial\_number in  
(select serial\_number  
from i\_food\_instruments  
where ven\_id = vid));

LEVEL 5:

F\_PURCHASE\_ORDER\_INVOICES  
where ifi\_serial\_number in  
(select serial\_number  
from i\_food\_instruments  
where ven\_id = vid);

I\_FI\_REJECT\_REASONS  
where ifi\_serial\_number in  
(select serial\_number  
from i\_food\_instruments  
where ven\_id = vid);

I\_FI\_FORMULAS  
where ifi\_serial\_number in  
(select serial\_number  
from i\_food\_instruments  
where ven\_id = vid);

LEVEL 4:

V\_CMP\_PAYMENT\_SCHEDULES

## LEVEL 3:

V\_FL\_PRICES  
V\_SANCTIONS  
V\_ACTIVITY\_FINDINGS  
V\_COMPLIANCE\_CASE\_CLIENTS  
    where vcc\_case\_id in  
    (select case\_id  
      from v\_compliance\_cases  
      where ven\_id = vid);  
V\_VEN\_WS\_FOOD\_GROUPS  
V\_DENIAL\_REASONS  
V\_AUTHORIZATION\_MILESTONES  
V\_PRICES  
V\_REQUIRED\_FOLLOWUPS  
V\_SUSPENSIONS  
V\_DISQUALIFICATIONS  
V\_CIVIL\_MONEY\_PENALTIES  
V\_APPEAL\_STEPS

## LEVEL 2:

I\_FOOD\_INSTRUCTIONS  
V\_APPEALS  
V\_MON\_ACTIVITIES  
V\_VENDOR\_ACCOUNTS  
V\_SURVEYS  
V\_VENDOR\_PHONES  
V\_EDUCATIONS  
V\_HOURS\_OF\_OPERATIONS  
V\_COMPLIANCE\_CASES  
V\_VENDOR\_FSP\_VIOLATIONS  
V\_RISK\_LEVEL\_HISTORIES  
V\_COLLECTIONS  
V\_VENDOR\_WHOLESALEERS  
V\_LA\_CLINICS  
V\_AUTHORIZATIONS  
V\_COMPLAINTS  
V\_COMMUNICATIONS  
V\_OVERALL\_RANKS  
V\_FELONIES  
V\_MULTI\_WIC  
V\_WIC\_VIOLATIONS  
V\_PS\_RESPONSES

## LEVEL 1:

v\_vendors

The S\_VENDOR\_ARCHIVES table is then updated by setting the ARCHIVE\_FLAG to 'N'

**1.29.1.4** Outputs

F\_PURCHASE\_ORDER\_LINE\_ITEMS  
A\_APPT\_TOPIC\_MATERIALS  
A\_APPT\_TOPICS  
A\_APPT\_APPTITEMS  
C\_CLIENT\_SVC\_NE\_MATERIALS  
C\_B\_N\_HH\_REASONS\_BFENDS  
F\_WAIT\_LIST\_CONTACTS  
C\_DIET\_NUTRIENTS  
C\_I\_C\_HH\_REASONS\_BFENDS  
C\_INCOMES  
A\_APPOINTMENTS  
C\_CLIENT\_NE\_TOPICS  
C\_WOMAN\_MEDICALS  
C\_BLOODWORK\_DATA  
C\_CERT\_TERM\_REASONS  
C\_B\_N\_HEALTHS  
C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
F\_WAIT\_LISTS  
C\_INFANT\_CHILD\_MEDICALS  
C\_DIETARY\_ASSESSMENTS  
C\_HEALTH\_RISK\_FACTORS  
C\_I\_C\_HEALTHS  
C\_ECONOMIC\_UNIT\_MEMBERS  
C\_PEER\_RBFENDS  
C\_CONTACTS  
C\_P\_COUNS\_NOTES  
I\_BF\_VCHR\_ITEMS  
I\_BF\_PROMO\_VCHRS  
C\_CLIENT\_SERVICES  
C\_CERTIFICATIONS  
C\_TRANSFER\_HISTORIES  
C\_IMMUNIZATIONS  
C\_CLIENT\_REFERRALS  
C\_CLIENT\_COMMUNICATIONS  
C\_RESOLUTIONS  
C\_CLIENT\_NOTES  
C\_CLIENT\_PROGS  
C\_CLIENT\_GROUPS  
C\_INCOME\_HISTORIES  
C\_INFANT\_DATA  
C\_CLIENT\_GOALS  
C\_CERT\_PEER\_COUNSELS  
F\_PURCHASE\_ORDER\_INVOICES  
I\_FI\_REJECT\_REASONS  
I\_FI\_FORMULAS  
S\_CLIENT\_ARCHIVES  
S\_VENDOR\_ARCHIVES  
V\_PAYMENT\_SCHEDULES  
I\_FOOD\_INSTRUMENTS  
V\_SANCTIONS

V\_ACTIVITY\_FINDINGS  
I\_BF\_VCHR\_ITEMS  
V\_FI\_PRICES  
V\_COMPLIANCE\_CASE\_CLIENTS  
V\_APPEALS  
V\_MON\_ACTIVITIES  
I\_BF\_PROMO\_VCHRS  
V\_VEN\_WS\_FOOD\_GROUPS  
V\_DENIAL\_REASONS  
V\_AUTHORIZATION\_MILESTONES  
V\_PRICES  
V\_SURVEYS  
V\_ACCOUNTS  
V\_VENDOR\_PHONES  
V\_EDUCATIONS  
V\_HOURS\_OF\_OPERATIONS  
V\_COMPLIANCE\_CASES  
I\_BF\_PROMO\_PAYMENTS  
V\_VENDOR\_HEALTHS  
V\_VENDOR\_WHOLESALEERS  
V\_LA\_CLINICS  
V\_AUTHORIZATIONS  
V\_COMPLAINTS  
V\_COMMUNICATIONS  
V\_OVERALL\_RANKS

## 1.29.2 EC4\_RETRIEVE.SQL

### *Overview*

Run the following scripts (in system Administration):

- \* EC4\_TRUNC\_ARCH\_CLI.SQL (reference System Administration DTSDSection 6 - 1.1.4.1)
- \* EC4\_INS\_RETR\_CLI.SQL
- \* EC4\_TRUNC\_ARCH\_VEN.SQL (reference System Administration DTSDSection 6 - 1.1.4.1)
- \* EC4\_INS\_RETR\_VEN.SQL

### 1.29.2.1 Inputs

N/A

### 1.29.2.2 Selection

N/A

### 1.29.2.3 Processing

The following scripts are run:

EC4\_TRUNC\_ARCH\_CLI.SQL  
EC4\_INS\_RETR\_CLI.SQL  
EC4\_TRUNC\_ARCH\_VEN.SQL  
EC4\_INS\_RETR\_VEN.SQL

### 1.29.2.4 Outputs

%AGCY%EC4\_RETRIEVE.LOG

### 1.29.3 EC4\_INS\_RETR\_CLI.SQL

#### *Overview*

Performs insert from Client tables into all temporary tables for retrieval of archived information. These tables are temporarily storing this information for transfer back into the AIM Certification tables. The temporary tables have a prefix of 'R' which represents that the tables are storing archived information that is being retrieved

r\_c\_clients  
r\_c\_bloodwork\_data  
r\_c\_w\_hh\_reasons\_bfends  
r\_c\_certifications  
r\_c\_cert\_peer\_counsels  
r\_c\_cert\_term\_reasons  
r\_c\_client\_communications  
r\_c\_client\_goals  
r\_c\_client\_ne\_topics  
r\_c\_client\_notes  
r\_c\_client\_progs  
r\_c\_client\_referrals  
r\_c\_client\_services  
r\_c\_client\_svc\_ne\_materials  
r\_c\_contacts  
r\_c\_dietary\_assessments  
r\_c\_diet\_nutrients  
r\_c\_economic\_unit\_members  
r\_c\_family\_economic\_units  
r\_c\_family\_phones  
r\_c\_fam\_referrals  
r\_c\_feu\_communications  
r\_c\_food\_pack\_scripts  
r\_c\_health\_risk\_factors  
r\_c\_immunizations  
r\_c\_incomes  
r\_c\_income\_histories  
r\_c\_infant\_child\_medicals  
r\_c\_infant\_data  
r\_c\_i\_c\_healths  
r\_c\_i\_c\_hh\_reasons\_bfends  
r\_c\_peer\_rbfends  
r\_c\_p\_couns\_notes  
r\_c\_w\_healths  
r\_c\_resolutions  
r\_c\_transfer\_histories  
r\_c\_woman\_medicals  
r\_s\_client\_archives

**1.29.3.1 Inputs**

```
C_CLIENTS CC
S_CLIENT_ARCHIVES SCA
```

**1.29.3.2 Selection**

```
WHERE sca.client_id = cc.client_id
      AND SCA.ARCHIVE_FLAG = 'R'
      AND (TRUNC(sca.date_modified) >
           (SELECT good_proc_date FROM e_eod_controls) )
      ORDER BY cc.client_id;
```

This where clause selects where S\_CLIENT\_ARCHIVES.CLIENT\_ID equals the C\_CLIENTS.CLIENT\_ID and the ARCHIVE\_FLAG indicates that information has been archived and is being retrieved ('R') and the date modified is greater than the last successful procedure date. The ARCHIVE\_FLAG is set equal to 'R' by first setting the flag to 'Y' which indicates that a record is to be archived. After the information has been deleted and archived by the system the ARCHIVE\_FLAG is set to 'N' which is then set to 'R' when a retrieval request is made.

**1.29.3.3 Processing**

```
IF search_clients%FOUND THEN
```

```
INSERT INTO (list of output tables, each field in table)
```

**1.29.3.4 Outputs**

```
r_c_clients
r_c_bloodwork_data
r_c_w_healths
r_c_b_n_hh_reasons_bfends
r_c_certifications
r_c_cert_peer_counsels
r_c_cert_term_reasons
r_c_client_communications
r_c_client_goals
r_c_client_ne_topics
r_c_client_notes
r_c_client_progs
r_c_client_referrals
r_c_client_services
```

r\_c\_client\_svc\_ne\_materials  
r\_c\_contacts  
r\_c\_dietary\_assessments  
r\_c\_diet\_nutrients  
r\_c\_economic\_unit\_members  
r\_c\_family\_economic\_units  
r\_c\_family\_phones  
r\_c\_fam\_referrals  
r\_c\_feu\_communications  
r\_c\_food\_pack\_scripts  
r\_c\_health\_risk\_factors  
r\_c\_immunizations  
r\_c\_incomes  
r\_c\_income\_histories  
r\_c\_infant\_child\_medicals  
r\_c\_infant\_data  
r\_c\_i\_c\_healths  
r\_c\_i\_c\_hh\_reasons\_bfends  
r\_c\_peer\_rbfends  
r\_c\_p\_couns\_notes  
r\_c\_w\_healths  
r\_c\_resolutions  
r\_c\_transfer\_histories  
r\_c\_woman\_medicals  
r\_s\_client\_archives

#### 1.29.4 EC4\_INS\_RETR\_VEN.SQL

##### *Overview*

Performs insert from Vendor tables into all temporary tables.

r\_v\_vendors  
r\_v\_accounts  
r\_v\_activity\_findings  
r\_v\_appeals  
r\_v\_authorizations  
r\_v\_auth\_milestones  
r\_v\_communications  
r\_v\_complaints  
r\_v\_compliance\_cases  
r\_v\_denial\_reasons  
r\_v\_educations  
r\_v\_fi\_prices  
r\_v\_hours\_of\_operations  
r\_v\_la\_clinics  
r\_v\_mon\_activities  
r\_v\_overall\_ranks

r\_v\_owners  
r\_v\_owner\_phones  
r\_v\_payment\_schedules  
r\_v\_prices  
r\_v\_sanctions  
r\_v\_surveys  
r\_v\_vendor\_fsps  
r\_v\_vendor\_healths  
r\_v\_vendor\_phones  
r\_v\_vendor\_wholesalers  
r\_v\_ven\_ws\_food\_groups  
r\_s\_vendor\_archives

#### 1.29.4.1 Inputs

V\_VENDORS VV  
S\_VENDOR\_ARCHIVES SVV

#### 1.29.4.2 Selection

```
WHERE vv.id = svv.id
      AND SVV.ARCHIVE_FLAG = 'R'
      AND (TRUNC(svv.date_modified) >
           (SELECT good_proc_date FROM e_eod_controls))
ORDER BY vv.id
```

This where clause selects where V\_VENDORS.ID equals the S\_VENDOR\_ARCHIVES.ID and the ARCHIVE\_FLAG indicates that information has been archived and is being retrieved ('R') and the date modified is greater than the last successful procedure date. The ARCHIVE\_FLAG is set equal to 'R' by first setting the flag to 'Y' which indicates that a record is to be archived. After the information has been deleted and archived by the system the ARCHIVE\_FLAG is set to 'N' which is then set to 'R' when a retrieval request is made.

#### 1.29.4.3 Processing

```
IF search_vendors%FOUND THEN

INSERT INTO (list of output tables)
```

#### 1.29.4.4 Outputs

r\_v\_vendors  
r\_v\_accounts

r\_v\_activity\_findings  
r\_v\_appeals  
r\_v\_authorizations  
r\_v\_auth\_milestones  
r\_v\_communications  
r\_v\_complaints  
r\_v\_compliance\_cases  
r\_v\_denial\_reasons  
r\_v\_educations  
r\_v\_fi\_prices  
r\_v\_hours\_of\_operations  
r\_v\_la\_clinics  
r\_v\_mon\_activities  
r\_v\_overall\_ranks  
r\_v\_owners  
r\_v\_owner\_phones  
r\_v\_payment\_schedules  
r\_v\_prices  
r\_v\_sanctions  
r\_v\_surveys  
r\_v\_vendor\_fsps  
r\_v\_vendor\_healths  
r\_v\_vendor\_phones  
r\_v\_vendor\_wholesalers  
r\_v\_ven\_ws\_food\_groups  
r\_s\_vendor\_archives

### **1.30** Print out status logs

#### **1.30.1** EC5.BAT

##### Overview

This batch file initiates the printing of Central log files.

##### Processing

Called by EC1.BAT.

The script first reads in the EC4.INI file to initialize and set up temporary environmental variables.

It then calls the EC\_PRINT.BAT batch file which combines Central processing logs for output.

#### **1.30.2** EC\_PRINT.BAT

##### *Overview*

This batch file produces master log files by agency for the following log files:

- \* Truncate
- \* Import
- \* Delete
- \* Insert
- \* Update
- \* End of day import
- \* End of day insert
- \* End of day retrieve
- \* Export

##### **1.30.2.1** Inputs

N/A

##### **1.30.2.2** Selection

Called by EC5.BAT

### 1.30.2.3 Processing

The system sets the environmental variables for the particular Local Agency reading them from the EC4.INI file and places them in a temporary file to be used for retrieving scripts.

Prints out combined log files for the following files:

1. Combines the EC.LOG with the EC\_HIS.LOG by writing it to the end of the EC\_HIS.LOG file.
2. Writes the EC1\_SQL.LOG, EC2\_SQL.LOG, EC4\_BANK\_SQL.LOG, EC4\_SQL.LOG, EC4\_TRNC.LOG and %\_EC2B\_SQL.LOG files to the end of the CTRL\_SQL.LOG (new file each day) and SQL.LOG (history file of CTRL\_SQL.LOG).
3. Then creates the files AGCY\_%.LOG (new file each day) and AGCY\_%\_HIS.LOG (history file of AGCY\_%.LOG) that log the truncate, import, delete - insert - update - delete, EOD import, EOD insert, EOD retrieve, and export information. The following logs for each Local Agency are written to the end of the AGCY\_%.LOG and AGCY\_%\_HIS.LOG (the wildcard represents the Local Agencies ID):
  - The truncate section: %\_TRUNC.LOG
  - The import section: %\_IMP.LOG
  - The delete - insert - update - delete section: %\_EC2\_AGENCY.LOG
  - The EOD import section: %EOD\_IMP.LOG
  - The EOD insert section: %\_EC4\_TAB.LOG
  - The EOD retrieve section: %\_EC4\_RETRIEVE.LOG
  - The export section: %EXP.LOG

The final combined files are EC\_HIS.LOG, AGCY\_CTRL.LOG, CTRL\_SQL.LOG, SQL.LOG, AGCY\_%\_HIS.LOG and AGCY\_%.LOG. Examples of these log files can be found in Appendix A.

### 1.30.2.4 Outputs

Agcy\_%\_his.log. See Appendix A for example of the layout.

### 1.31 Consolidate information from central

#### 1.31.1 EA3.BAT

##### Overview

This batch file runs at the Local Agencies after Central processing has been completed.

##### Processing

The system sets the environmental variables for the particular Local Agency reading them from the EA1.INI file and places them in a temporary file.

This script retrieves the return DMP file from central by running EA3\_FTP\_RUN.BAT, then searches for a completion flag (%.don) from the particular Agency to begin loading the Agencies data from the DMP file.

The agency is then imported by starting the following SQL scripts:

- EA\_TRGOFF.SQL which disables all triggers and constraints.
- EA3\_TRNC.SQL which initializes the tables.

After importing is completed EA3\_PRE.SQL is started which updates organizational units staff member id to null if it's not in the current agency.

The following SQL scripts are then initiated to delete, insert and update the Agency:

- EA3\_DELTRIG.SQL
- EA3\_IN\_TAB.SQL
- EA3\_TAB\_UP.SQL
- EA3\_SYNC\_LA\_ASSIGNMENT.SQL
- EA3\_TRGON.SQL

Once the Agency has been updated EA3.SQL and EA3\_RETRIEVE.SQL are executed.

EA3.SQL runs the following SQL scripts:

- EA3\_TERM\_TR\_CLIENTS.SQL
- EA3\_PROC\_FI.SQL
- EA3\_PRG\_ARCHV.SQL

EA3\_RETRIEVE.SQL runs the following scripts:

- EA3\_INS\_RETR\_CLI.SQL
- EA3\_INS\_RETR\_VEN.SQL

The script then executes the batch file EA\_PRINT.BAT.

The script finishes by executing EA3\_DBA.SQL to monitor DBA activities.

### 1.31.2 EA\_TRGOFF.SQL

#### Overview

Sets database triggers off, so that table contents can be modified without restrictions.

#### 1.31.2.1 Inputs

N/A

#### 1.31.2.2 Selection

N/A

#### 1.31.2.3 Processing

Alter the following tables with 'DISABLE ALL TRIGGERS'

F\_ANNUAL\_FACTORS  
I\_FOOD\_INSTRUMENT\_TYPES  
C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
C\_CLIENT\_SERVICES

Alter the following tables with 'DISABLE CONSTRAINT'

I\_FOOD\_INSTRUMENT\_TYPES with constraint IFIT\_IFIT\_FK  
I\_FOOD\_PACKAGE\_FI\_TYPES with constraint IFPFT\_IFPFT\_FK  
F\_MANUFACTURERS with constraint MAN\_FMC\_FK  
F\_MFR\_CONTACTS with constraint FMC\_MAN\_FK  
C\_RESOLUTIONS with constraint CR1\_CC\_FK

#### 1.31.2.4 Outputs

N/A

### 1.31.3 EA3\_TRNC.SQL

#### Overview

This script empties out the a\_\* and r\_\* tables before they are filled with data sent from the Central Processor to the Agencies.

### 1.31.3.1 Inputs

The following is a list of tables that are truncated by this script:

A\_STATE\_CLIENTS  
A\_C\_STATE\_CLIENTS  
A\_F\_POVERTY\_BASES  
A\_EOD\_CONTROLS  
A\_A\_ACTIVITIES  
A\_A\_APPT\_ITEMS  
A\_AAS\_CLASS\_CATEGORIES  
A\_AAS\_ITEMS  
A\_A\_ATTEND\_STATUSES  
A\_A\_OFFICE\_CLOSED  
A\_A\_SERVICES  
A\_I\_AGE\_RANGES  
A\_C\_ANSWERS  
A\_C\_ANSWER\_TYPES  
A\_C\_BLOODWORK\_TYPES  
A\_C\_BREAST\_PUMP\_REASONS  
A\_C\_BREAST\_PUMP\_TYPES  
A\_C\_CATEGORIES  
A\_C\_CAT\_BLOOD\_FACTORS  
A\_C\_CAT\_GOALS  
A\_C\_CAT\_NUTR\_EDS  
A\_C\_CAT\_REFERRALS  
A\_C\_COMMUNICATION\_TYPES  
A\_C\_CP\_MESSAGES  
A\_C\_DIAGNOSES  
A\_C\_DIETARY\_REQUIREMENTS  
A\_C\_DIET\_NUTRIENT\_TYPES  
A\_C\_DISABILITIES  
A\_C\_EDUCATION\_LEVELS  
A\_C\_ELEVATIONS  
A\_C\_ETHNIC\_GROUPS  
A\_C\_GOALS  
A\_C\_HC\_PAYEES  
A\_C\_HH\_QUESTIONS  
A\_C\_HH\_RESPONSES  
A\_C\_INCOME\_INTERVALS  
A\_C\_INCOME\_LEVELS  
A\_C\_INCOME\_SOURCES  
A\_C\_INCOME\_VERIFICATIONS  
A\_C\_INFANT\_STATUSES  
A\_C\_LANGUAGES  
A\_C\_MARITAL\_STATUSES  
A\_C\_NCHS\_CLASSIFICATIONS  
A\_C\_NCHS\_TYPES  
A\_C\_NCHS\_DATA  
A\_C\_NO\_CONTACT\_REASONS

A\_C\_NUTR\_ED\_MATERIALS  
A\_C\_NUTR\_ED\_TOPICS  
A\_C\_PICKUP\_INTERVALS  
A\_C\_PILOT\_QUESTIONS  
A\_C\_PILOT\_STUDIES  
A\_C\_PRIORITIES  
A\_C\_PROOF\_ADDRESSES  
A\_C\_PROOF\_IDENTITIES  
A\_C\_PS\_QUESTIONS  
A\_C\_RACES  
A\_C\_REASONS\_BF\_ENDED  
A\_C\_RISK\_FACTORS  
A\_C\_RISK\_FACTOR\_DIAGNOSES  
A\_C\_RISK\_FACTOR\_TYPES  
A\_C\_RF\_GOALS  
A\_C\_RF\_NUTR\_EDS  
A\_C\_RF\_REFERRALS  
A\_C\_RF\_SERVICES  
A\_C\_HH\_RESPONSE\_RFS  
A\_C\_BMI\_DATA  
A\_C\_BMI\_ANTHROPOMETRIC  
A\_C\_BMI\_WEIGHT\_GAIN  
A\_C\_SCHEDULE\_DAYS  
A\_C\_SMOKING\_CHANGES  
A\_C\_SMOKING\_STAGES  
A\_C\_SOURCES\_HEALTH\_CARE  
A\_C\_SYMPTOMS  
A\_C\_TERM\_REASONS  
A\_C\_TOPICS  
A\_C\_VOTER\_REGISTRATIONS  
A\_C\_CAT\_BLOODWORKS  
A\_C\_DESIREABLE\_WEIGHTS  
A\_C\_IMMUNIZATIONS\_NOT\_ASSESSED  
A\_F\_CONTROLS  
A\_F\_CASELOAD\_TYPES  
A\_F\_FUND\_SOURCES  
A\_F\_POVERTY\_LEVELS  
A\_F\_WAIT\_LIST\_RESPONSES  
A\_I\_BANK\_DISPOSITIONS  
A\_I\_CATEGORY\_GROUPS  
A\_I\_PACKAGES  
A\_I\_PRODUCTS  
A\_I\_REJECT\_REASONS  
A\_I\_CONTAINERS  
A\_I\_DISPOSITIONS  
A\_I\_UNITS\_OF\_MEASURE  
A\_I\_VOID\_REASONS  
A\_I\_FOOD\_GROUPS  
A\_I\_FOODS  
A\_I\_FOOD\_DISTRIBUTIONS  
A\_I\_MAXIMUM\_FOODS

A\_I\_FOOD\_PACKAGES  
A\_I\_CATEGORY\_GROUP\_PKGS  
A\_I\_FOOD\_PACKAGE\_FI\_TYPES  
A\_I\_FOOD\_PACKAGE\_FOODS  
A\_I\_PACKAGE\_DISTRIBUTIONS  
A\_I\_WS\_FOOD\_GROUPS  
A\_I\_FOOD\_INSTRUMENT\_TYPES  
A\_I\_FOOD\_INSTRUMENT\_FOODS  
A\_C\_FOOD\_PKG\_RISK\_FACTORS  
A\_O\_OUTREACH\_ORG\_TYPES  
A\_O\_OUTREACH\_COMM\_TYPES  
A\_O\_OUTREACH\_ORGANIZATIONS  
A\_O\_OUTREACH\_ORG\_PHONES  
A\_O\_OUTREACH\_COMMS  
A\_O\_PROGRAMS  
A\_O\_PROGRAM\_DATES  
A\_O\_TITLE\_CATEGORIES  
A\_O\_STAFF\_TITLES  
A\_S\_CITIES  
A\_S\_CONTACT\_METHODS  
A\_S\_CONTACT\_TITLES  
A\_S\_COUNTIES  
A\_S\_PHONE\_TYPES  
A\_S\_STATES  
A\_S\_ZIPS  
A\_V\_ACTIVITY\_TYPES  
A\_V\_APPLICATION\_MILESTONES  
A\_V\_APPLICATION\_TYPES  
A\_V\_BANK\_BRANCHES  
A\_V\_CASE\_STATUSES  
A\_V\_COLLECTION\_TYPES  
A\_V\_COMMUNICATION\_TYPES  
A\_V\_COMPLAINT\_SOURCE\_TYPES  
A\_V\_COMPLAINT\_STATUSES  
A\_V\_COMPLAINT\_SUBJECTS  
A\_V\_COMPLIANCE\_CASE\_DESIGNATNS  
A\_V\_COMPLIANCE\_CASE\_TYPES  
A\_V\_DELIVERY\_TYPES  
A\_V\_DENIAL\_REASONS  
A\_V\_DISQUAL\_REASONS  
A\_V\_FINDING\_CODES  
A\_V\_FSP\_REGIONAL\_OFFICES  
A\_V\_FSP\_VIOLATION\_CODES  
A\_V\_LEGAL\_FIRMS  
A\_V\_LEGAL\_REPRESENTATIVES  
A\_V\_MILESTONE\_TYPES  
A\_V\_OWNER\_TYPES  
A\_V\_PEER\_GROUPS  
A\_V\_RISK\_LEVELS  
A\_V\_SANCTION\_TYPES  
A\_V\_STATUSES

A\_V\_SUSPENSION\_REASONS  
A\_V\_VIOLATION\_ACTION  
A\_V\_WHOLESALEERS  
A\_V\_WIC\_CODES  
A\_V\_OWNERS  
A\_V\_VENDORS  
A\_O\_STAFF\_MEMBERS  
A\_O\_ORGANIZATIONAL\_UNITS  
A\_O\_ANNUAL\_WIC\_COST\_SUMMARIES  
A\_F\_ANNUAL\_FACTORS  
A\_F\_BUDGETS  
A\_F\_CASE\_ASSIGNMENTS  
A\_F\_CASELOADS  
A\_F\_CASELOAD\_RESTRICTIONS  
A\_F\_MFR\_CONTACTS  
A\_F\_MANUFACTURERS  
A\_I\_FI\_INVENTORIES  
A\_I\_STOCK\_INVENTORIES  
A\_I\_FOOD\_INSTRUMENTS  
A\_I\_FI\_REJECT\_REASONS  
A\_C\_RESOLUTIONS  
A\_C\_RESOLVED\_CLIENTS  
A\_S\_GEO\_LOCATIONS;  
A\_S\_PRINTER\_ADDRESSES;  
A\_DEL\_STATEMENTS  
R\_S\_CLIENT\_ARCHIVES  
PROMPT table containing clients who have been retrieved from the archives  
R\_ARCHIVED\_CLIENTS  
PROMPT table containing clients who have been archived at central

### **1.31.3.2 Selection**

All records are deleted from the tables listed.

### **1.31.3.3 Processing**

Performs an Oracle TRUNCATE on each of the tables, removing all records in the tables.

### **1.31.3.4 Outputs**

Same as Inputs (Section 1.32.3.1.)

### 1.31.4 EA3\_PRE.SQL

#### *Overview*

This script updates organizational units staff member ID to null if it's not in the current agency.

#### 1.31.4.1 Input

A\_O\_ORGANIZATIONAL\_UNITS

#### 1.31.4.2 Selection

Updates staff member's ID to NULL if it does not belong to the current agency:

For the a\_o\_organizational\_units table:

WHERE OU\_SEQ\_ID is not the Server's local agency sequence number  
AND SEQ\_ID is not the Server's local agency sequence number

The Server's local agency is defined as: SELECT ENV('AGENCY\_SEQ') FROM DUAL

#### 1.31.4.3 Processing

UPDATE A\_O\_ORGANIZATIONAL\_UNITS SET SM\_STAFF\_MEMBER\_ID = NULL

#### 1.31.4.4 Output

A\_O\_ORGANIZATIONAL\_UNITS

### 1.31.5 EA3\_DELTRIG.SQL

#### *Overview*

This script deletes data from the local agency database that had been deleted from the central database since the last end of day run.

#### 1.31.5.1 Inputs

## A\_DEL\_STATEMENTS

### 1.31.5.2 Selection

for a\_del\_statements:

```
WHERE target_proc_date IS NULL
ORDER BY origin_agcy, del_seq
```

### 1.31.5.3 Processing

Loop through all selected records on table a\_del\_statements:

```
write out the a_del_statements.del_statement to a log
```

```
set local variables:
```

```
cursor_var := DBMS_SQL.OPEN_CURSOR;
delete_statement := a_del_statements.del_statement;
```

```
send the command to SQL:
```

```
DBMS_SQL.PARSE(cursor_var, DELETE_STATEMENT, dbms_sql.v7);
```

```
capture the return value and close the SQL:
```

```
ret_value := DBMS_SQL.EXECUTE(cursor_var);
DBMS_SQL.CLOSE_CURSOR(cursor_var);
```

```
End loop;
```

```
update the a_del_statements, setting the target_proc_date to the current date where the target_proc_date is null
```

### 1.31.5.4 Outputs

```
a_del_statements
```

```
message to print the delete statements processed:
```

```
EA3_DELTRIG.LOG
```

### 1.31.6 EA3\_IN\_TAB.SQL

#### *Overview*

This script inserts records into AIM tables from End of Day admin tables.

A\_ACTIVITIES  
AAS\_CLASS\_CATEGORIES  
AAS\_ITEMS  
A\_APPT\_ITEMS  
A\_ATTEND\_STATUSES  
A\_OFFICE\_CLOSED  
A\_SERVICES  
I\_AGE\_RANGES  
C\_ANSWERS  
C\_ANSWER\_TYPES  
C\_BLOODWORK\_TYPES  
C\_BREAST\_PUMP\_REASONS  
C\_BREAST\_PUMP\_TYPES  
C\_CATEGORIES  
C\_CAT\_BLOOD\_FACTORS  
C\_CAT\_GOALS  
C\_CAT\_NUTR\_EDS  
C\_CAT\_REFERRALS  
C\_COMMUNICATION\_TYPES  
C\_CP\_MESSAGES  
C\_DIAGNOSES  
C\_DIETARY\_REQUIREMENTS  
C\_DIET\_NUTRIENT\_TYPES  
C\_DISABILITIES  
C\_EDUCATION\_LEVELS  
C\_ELEVATIONS  
C\_ETHNIC\_GROUPS  
C\_GOALS  
C\_HC\_PAYEES  
C\_HH\_QUESTIONS  
C\_HH\_RESPONSES  
C\_INCOME\_INTERVALS  
C\_INCOME\_LEVELS  
F\_POVERTY\_BASES  
C\_INCOME\_SOURCES  
C\_INCOME\_VERIFICATIONS  
C\_INFANT\_STATUSES  
C\_LANGUAGES  
C\_MARITAL\_STATUSES  
C\_NCHS\_CLASSIFICATIONS  
C\_NCHS\_TYPES  
C\_NCHS\_DATA

C\_NO\_CONTACT\_REASONS  
C\_NUTR\_ED\_MATERIALS  
C\_NUTR\_ED\_TOPICS  
C\_PICKUP\_INTERVALS  
C\_PILOT\_QUESTIONS  
C\_PILOT\_STUDIES  
C\_PRIORITIES  
C\_PROOF\_ADDRESSES  
C\_PROOF\_IDENTITIES  
C\_PS\_QUESTIONS  
C\_RACES  
C\_REASONS\_BF\_ENDED  
C\_RISK\_FACTORS  
C\_RISK\_FACTOR\_DIAGNOSES  
C\_RISK\_FACTOR\_TYPES  
C\_RF\_GOALS  
C\_RF\_NUTR\_EDS  
C\_RF\_REFERRALS  
C\_RF\_SERVICES  
C\_HH\_RESPONSE\_RFS  
C\_BMI\_DATA  
C\_BMI\_ANTHROPOMETRIC  
C\_BMI\_WEIGHT\_GAIN  
C\_SCHEDULE\_DAYS  
C\_SMOKING\_CHANGES  
C\_SMOKING\_STAGES  
C\_SOURCES\_HEALTH\_CARE  
C\_SYMPTOMS  
C\_TERM\_REASONS  
C\_TOPICS  
C\_VOTER\_REGISTRATIONS  
C\_CAT\_BLOODWORKS  
C\_DESIREABLE\_WEIGHTS  
C\_IMMUNIZATIONS\_NOT\_ASSESSED  
F\_CONTROLS  
F\_CASELOAD\_TYPES  
F\_FUND\_SOURCES  
F\_POVERTY\_LEVELS  
F\_WAIT\_LIST\_RESPONSES  
I\_BANK\_DISPOSITIONS  
I\_CATEGORY\_GROUPS  
I\_PACKAGES  
I\_PRODUCTS  
I\_REJECT\_REASONS  
I\_CONTAINERS  
I\_DISPOSITIONS  
I\_UNITS\_OF\_MEASURE  
I\_VOID\_REASONS  
I\_FOOD\_GROUPS  
I\_FOODS  
I\_FOOD\_DISTRIBUTIONS

I\_MAXIMUM\_FOODS  
I\_FOOD\_PACKAGES  
I\_CATEGORY\_GROUP\_PKGS  
I\_FOOD\_PACKAGE\_FOODS  
I\_PACKAGE\_DISTRIBUTIONS  
I\_WS\_FOOD\_GROUPS  
I\_FOOD\_INSTRUMENT\_TYPES  
I\_FOOD\_PACKAGE\_FI\_TYPES  
I\_FOOD\_INSTRUMENT\_FOODS  
C\_FOOD\_PKG\_RISK\_FACTORS  
O\_OUTREACH\_ORG\_TYPES  
O\_OUTREACH\_COMM\_TYPES  
O\_PROGRAMS  
O\_PROGRAM\_DATES  
O\_TITLE\_CATEGORIES  
O\_STAFF\_TITLES  
S\_CITIES  
S\_CONTACT\_METHODS  
S\_CONTACT\_TITLES  
S\_COUNTIES  
S\_PHONE\_TYPES  
S\_STATES  
S\_ZIPS  
S\_GEO\_LOCATIONS  
V\_ACTIVITY\_TYPES  
V\_APPLICATION\_MILESTONES  
V\_APPLICATION\_TYPES  
V\_BANK\_BRANCHES  
V\_CASE\_STATUSES  
V\_COLLECTION\_TYPES  
V\_COMMUNICATION\_TYPES  
V\_COMPLAINT\_SOURCE\_TYPES  
V\_COMPLAINT\_STATUSES  
V\_COMPLAINT\_SUBJECTS  
V\_COMPLIANCE\_CASE\_DESIGNATIONS  
V\_COMPLIANCE\_CASE\_TYPES  
V\_DELIVERY\_TYPES  
V\_DENIAL\_REASONS  
V\_DISQUAL\_REASONS  
V\_FINDING\_CODES  
V\_FSP\_REGIONAL\_OFFICES  
V\_FSP\_VIOLATION\_CODES  
V\_LEGAL\_FIRMS  
V\_LEGAL\_REPRESENTATIVES  
V\_MILESTONE\_TYPES  
V\_OWNER\_TYPES  
V\_PEER\_GROUPS  
V\_RISK\_LEVELS  
V\_SANCTION\_TYPES  
V\_STATUSES  
V\_SUSPENSION\_REASONS

V\_VIOLATION\_ACTION  
V\_WHOLESALEERS  
V\_WIC\_CODES  
V\_OWNERS  
V\_VENDORS  
O\_STAFF\_MEMBERS  
PROMPT EA3 updating :A\_O\_ORGANIZATIONAL\_UNITS.SM\_STAFF\_MEMBER\_ID TO  
NULL  
O\_ORGANIZATIONAL\_UNITS  
O\_ANNUAL\_WIC\_COST\_SUMMARIES  
F\_ANNUAL\_FACTORS  
F\_BUDGETS  
F\_CASE\_ASSIGNMENTS  
F\_CASELOADS  
F\_CASELOAD\_RESTRICTIONS  
F\_MFR\_CONTACTS  
F\_MANUFACTURERS  
I\_FI\_INVENTORIES  
I\_STOCK\_INVENTORIES  
C\_RESOLVED\_CLIENTS  
C\_RESOLUTIONS  
I\_FI\_REJECT\_REASONS  
S\_PRINTER\_ADDRESSES  
DEL\_STATEMENTS

#### 1.31.6.1 Inputs

Records are inserted from the following tables:

A\_EOD\_CONTROLS  
A\_A\_ACTIVITIES  
A\_AAS\_CLASS\_CATEGORIES  
A\_AAS\_ITEMS  
A\_A\_APPT\_ITEMS  
A\_A\_ATTEND\_STATUSES  
A\_A\_OFFICE\_CLOSED  
A\_A\_SERVICES  
A\_I\_AGE\_RANGES  
A\_C\_ANSWERS  
A\_C\_ANSWER\_TYPES  
A\_C\_BLOODWORK\_TYPES  
A\_C\_BREAST\_PUMP\_REASONS  
A\_C\_BREAST\_PUMP\_TYPES  
A\_C\_CATEGORIES  
A\_C\_CAT\_BLOOD\_FACTORS  
A\_C\_CAT\_GOALS  
A\_C\_CAT\_NUTR\_EDS  
A\_C\_CAT\_REFERRALS  
A\_C\_COMMUNICATION\_TYPES

A\_C\_CP\_MESSAGES  
A\_C\_DIAGNOSES  
A\_C\_DIETARY\_REQUIREMENTS  
A\_C\_DIET\_NUTRIENT\_TYPES  
A\_C\_DISABILITIES  
A\_C\_EDUCATION\_LEVELS  
A\_C\_ELEVATIONS  
A\_C\_ETHNIC\_GROUPS  
A\_C\_GOALS  
A\_C\_HC\_PAYEES  
A\_C\_HH\_QUESTIONS  
A\_C\_HH\_RESPONSES  
A\_C\_INCOME\_INTERVALS  
A\_C\_INCOME\_LEVELS  
A\_F\_POVERTY\_BASES  
A\_C\_INCOME\_SOURCES  
A\_C\_INCOME\_VERIFICATIONS  
A\_C\_INFANT\_STATUSES  
A\_C\_LANGUAGES  
A\_C\_MARITAL\_STATUSES  
A\_C\_NCHS\_CLASSIFICATIONS  
A\_C\_NCHS\_TYPES  
A\_C\_NCHS\_DATA  
A\_C\_NO\_CONTACT\_REASONS  
A\_C\_NUTR\_ED\_MATERIALS  
A\_C\_NUTR\_ED\_TOPICS  
A\_C\_PICKUP\_INTERVALS  
A\_C\_PILOT\_QUESTIONS  
A\_C\_PILOT\_STUDIES  
A\_C\_PRIORITIES  
A\_C\_PROOF\_ADDRESSES  
A\_C\_PROOF\_IDENTITIES  
A\_C\_PS\_QUESTIONS  
A\_C\_RACES  
A\_C\_REASONS\_BF\_ENDED  
A\_C\_RESOLUTIONS  
A\_C\_RESOLVED\_CLIENTS  
A\_C\_RISK\_FACTORS  
A\_C\_RISK\_FACTOR\_DIAGNOSES  
A\_C\_RISK\_FACTOR\_TYPES  
A\_C\_RF\_GOALS  
A\_C\_RF\_NUTR\_EDS  
A\_C\_RF\_REFERRALS  
A\_C\_RF\_SERVICES  
A\_C\_HH\_RESPONSE\_RFS  
A\_C\_BMI\_DATA  
A\_C\_BMI\_ANTHROPOMETRIC  
A\_C\_BMI\_WEIGHT\_GAIN  
A\_C\_SCHEDULE\_DAYS  
A\_C\_SMOKING\_CHANGES  
A\_C\_SMOKING\_STAGES

A\_C\_SOURCES\_HEALTH\_CARE  
A\_C\_SYMPTOMS  
A\_C\_TERM\_REASONS  
A\_C\_TOPICS  
A\_C\_VOTER\_REGISTRATIONS  
A\_C\_CAT\_BLOODWORKS  
A\_C\_DESIREABLE\_WEIGHTS  
PROMPT EA1 inserting :A\_C\_IMMUNIZATIONS\_NOT\_ASSESSED  
A\_F\_CONTROLS  
A\_F\_CASELOAD\_TYPES  
A\_F\_FUND\_SOURCES  
A\_F\_POVERTY\_LEVELS  
A\_F\_WAIT\_LIST\_RESPONSES  
A\_I\_BANK\_DISPOSITIONS  
A\_I\_CATEGORY\_GROUPS  
A\_I\_PACKAGES  
A\_I\_PRODUCTS  
A\_I\_REJECT\_REASONS  
A\_I\_CONTAINERS  
A\_I\_DISPOSITIONS  
A\_I\_UNITS\_OF\_MEASURE  
A\_I\_VOID\_REASONS  
A\_I\_FOOD\_GROUPS  
A\_I\_FOODS  
A\_I\_FOOD\_DISTRIBUTIONS  
A\_I\_MAXIMUM\_FOODS  
A\_I\_FOOD\_PACKAGES  
A\_I\_CATEGORY\_GROUP\_PKGS  
A\_I\_FOOD\_PACKAGE\_FI\_TYPES  
A\_I\_FOOD\_PACKAGE\_FOODS  
A\_I\_PACKAGE\_DISTRIBUTIONS  
A\_I\_WS\_FOOD\_GROUPS  
A\_I\_FOOD\_INSTRUMENT\_TYPES  
A\_I\_FOOD\_INSTRUMENT\_FOODS  
A\_C\_FOOD\_PKG\_RISK\_FACTORS  
A\_O\_OUTREACH\_ORG\_TYPES  
A\_O\_OUTREACH\_COMM\_TYPES  
--A\_O\_OUTREACH\_ORGANIZATIONS  
--A\_O\_OUTREACH\_ORG\_PHONES  
--A\_O\_OUTREACH\_COMMS  
A\_O\_PROGRAMS  
A\_O\_PROGRAM\_DATES  
A\_O\_TITLE\_CATEGORIES  
A\_O\_STAFF\_TITLES  
A\_S\_CITIES  
A\_S\_CONTACT\_METHODS  
A\_S\_CONTACT\_TITLES  
A\_S\_COUNTIES  
A\_S\_PHONE\_TYPES  
A\_S\_STATES  
A\_S\_ZIPS

A\_S\_GEO\_LOCATIONS  
A\_V\_ACTIVITY\_TYPES  
A\_V\_APPLICATION\_MILESTONES  
A\_V\_APPLICATION\_TYPES  
A\_V\_BANK\_BRANCHES  
A\_V\_CASE\_STATUSES  
A\_V\_COLLECTION\_TYPES  
A\_V\_COMMUNICATION\_TYPES  
A\_V\_COMPLAINT\_SOURCE\_TYPES  
A\_V\_COMPLAINT\_STATUSES  
A\_V\_COMPLAINT\_SUBJECTS  
A\_V\_COMPLIANCE\_CASE\_DESIGNATNS  
A\_V\_COMPLIANCE\_CASE\_TYPES  
A\_V\_DELIVERY\_TYPES  
A\_V\_DENIAL\_REASONS  
A\_V\_DISQUAL\_REASONS  
A\_V\_FINDING\_CODES  
A\_V\_FSP\_REGIONAL\_OFFICES  
A\_V\_FSP\_VIOLATION\_CODES  
A\_V\_LEGAL\_FIRMS  
A\_V\_LEGAL\_REPRESENTATIVES  
A\_V\_MILESTONE\_TYPES  
A\_V\_OWNER\_TYPES  
A\_V\_PEER\_GROUPS  
A\_V\_RISK\_LEVELS  
A\_V\_SANCTION\_TYPES  
A\_V\_STATUSES  
A\_V\_SUSPENSION\_REASONS  
A\_V\_VIOLATION\_ACTION  
A\_V\_WHOLESALEERS  
A\_V\_WIC\_CODES  
A\_V\_OWNERS  
A\_V\_VENDORS  
--A\_O\_STAFF\_MEMBERS  
A\_O\_ORGANIZATIONAL\_UNITS  
A\_O\_ANNUAL\_WIC\_COST\_SUMMARIES  
A\_F\_ANNUAL\_FACTORS  
A\_F\_BUDGETS  
A\_F\_CASE\_ASSIGNMENTS  
A\_F\_CASELOADS  
A\_F\_CASELOAD\_RESTRICTIONS  
A\_F\_MFR\_CONTACTS  
A\_F\_MANUFACTURERS  
A\_I\_FI\_INVENTORIES  
A\_I\_STOCK\_INVENTORIES  
A\_I\_FOOD\_INSTRUMENTS  
A\_I\_FI\_REJECT\_REASONS  
S\_PRINTER\_ADDRESSES  
A\_DEL\_STATEMENTS

**1.31.6.2 Selection**

WHERE TRUNC (date\_created) > (date of last successful EOD))

**1.31.6.3 Processing**

Inserts rows into the agency server's tables by selecting the records from the corresponding a\_\* table:

```

Insert into aim.table
  (...columns...)
  (select ...columns...
    from eodadm.a_table
    where the date_created is after the last successful end of day run)

```

where table is the name of a table, such as a\_attend\_statuses  
and columns is the matching names of the columns in the tables.

For example:

```

Insert into aim.a_attend_statuses
  (attend_status_code, description, date_created, created_by, date_modified, modified_by,
  note)
  (Select attend_status_code, description, date_created, created_by, date_modified,
  modified_by, note
  from eodadm.a_a_attend_statuses
  where trunc(date_created) > (select good_proc_date from eodadm.a_eod_controls) )

```

**1.31.6.4 Outputs**

A\_ACTIVITIES  
AAS\_CLASS\_CATEGORIES  
AAS\_ITEMS  
A\_APPT\_ITEMS  
A\_ATTEND\_STATUSES  
A\_OFFICE\_CLOSED  
A\_SERVICES  
I\_AGE\_RANGES  
C\_ANSWERS  
C\_ANSWER\_TYPES  
C\_BLOODWORK\_TYPES  
C\_BREAST\_PUMP\_REASONS  
C\_BREAST\_PUMP\_TYPES  
C\_CATEGORIES  
C\_CAT\_BLOOD\_FACTORS

C\_CAT\_GOALS  
C\_CAT\_NUTR\_EDS  
C\_CAT\_REFERRALS  
C\_COMMUNICATION\_TYPES  
C\_CP\_MESSAGES  
C\_DIAGNOSES  
C\_DIETARY\_REQUIREMENTS  
C\_DIET\_NUTRIENT\_TYPES  
C\_DISABILITIES  
C\_EDUCATION\_LEVELS  
C\_ELEVATIONS  
C\_ETHNIC\_GROUPS  
C\_GOALS  
C\_HC\_PAYEES  
C\_HH\_QUESTIONS  
C\_HH\_RESPONSES  
C\_INCOME\_INTERVALS  
C\_INCOME\_LEVELS  
F\_POVERTY\_BASES  
C\_INCOME\_SOURCES  
C\_INCOME\_VERIFICATIONS  
C\_INFANT\_STATUSES  
C\_LANGUAGES  
C\_MARITAL\_STATUSES  
C\_NCHS\_CLASSIFICATIONS  
C\_NCHS\_TYPES  
C\_NCHS\_DATA  
C\_NO\_CONTACT\_REASONS  
C\_NUTR\_ED\_MATERIALS  
C\_NUTR\_ED\_TOPICS  
C\_PICKUP\_INTERVALS  
C\_PILOT\_QUESTIONS  
C\_PILOT\_STUDIES  
C\_PRIORITIES  
C\_PROOF\_ADDRESSES  
C\_PROOF\_IDENTITIES  
C\_PS\_QUESTIONS  
C\_RACES  
C\_REASONS\_BF\_ENDED  
C\_RISK\_FACTORS  
C\_RISK\_FACTOR\_DIAGNOSES  
C\_RISK\_FACTOR\_TYPES  
C\_RF\_GOALS  
C\_RF\_NUTR\_EDS  
C\_RF\_REFERRALS  
C\_RF\_SERVICES  
C\_HH\_RESPONSE\_RFS  
C\_BMI\_DATA  
C\_BMI\_ANTHROPOMETRIC  
C\_BMI\_WEIGHT\_GAIN  
C\_SCHEDULE\_DAYS

C\_SMOKING\_CHANGES  
C\_SMOKING\_STAGES  
C\_SOURCES\_HEALTH\_CARE  
C\_SYMPTOMS  
C\_TERM\_REASONS  
C\_TOPICS  
C\_VOTER\_REGISTRATIONS  
C\_CAT\_BLOODWORKS  
C\_DESIREABLE\_WEIGHTS  
C\_IMMUNIZATIONS\_NOT\_ASSESSED  
F\_CONTROLS  
F\_CASELOAD\_TYPES  
F\_FUND\_SOURCES  
F\_POVERTY\_LEVELS  
F\_WAIT\_LIST\_RESPONSES  
I\_BANK\_DISPOSITIONS  
I\_CATEGORY\_GROUPS  
I\_PACKAGES  
I\_PRODUCTS  
I\_REJECT\_REASONS  
I\_CONTAINERS  
I\_DISPOSITIONS  
I\_UNITS\_OF\_MEASURE  
I\_VOID\_REASONS  
I\_FOOD\_GROUPS  
I\_FOODS  
I\_FOOD\_DISTRIBUTIONS  
I\_MAXIMUM\_FOODS  
I\_FOOD\_PACKAGES  
I\_CATEGORY\_GROUP\_PKGS  
I\_FOOD\_PACKAGE\_FOODS  
I\_PACKAGE\_DISTRIBUTIONS  
I\_WS\_FOOD\_GROUPS  
I\_FOOD\_INSTRUMENT\_TYPES  
I\_FOOD\_PACKAGE\_FI\_TYPES  
I\_FOOD\_INSTRUMENT\_FOODS  
C\_FOOD\_PKG\_RISK\_FACTORS  
O\_OUTREACH\_ORG\_TYPES  
O\_OUTREACH\_COMM\_TYPES  
O\_PROGRAMS  
O\_PROGRAM\_DATES  
O\_TITLE\_CATEGORIES  
O\_STAFF\_TITLES  
S\_CITIES  
S\_CONTACT\_METHODS  
S\_CONTACT\_TITLES  
S\_COUNTIES  
S\_PHONE\_TYPES  
S\_STATES  
S\_ZIPS  
S\_GEO\_LOCATIONS

V\_ACTIVITY\_TYPES  
V\_APPLICATION\_MILESTONES  
V\_APPLICATION\_TYPES  
V\_BANK\_BRANCHES  
V\_CASE\_STATUSES  
V\_COLLECTION\_TYPES  
V\_COMMUNICATION\_TYPES  
V\_COMPLAINT\_SOURCE\_TYPES  
V\_COMPLAINT\_STATUSES  
V\_COMPLAINT\_SUBJECTS  
V\_COMPLIANCE\_CASE\_DESIGNATIONS  
V\_COMPLIANCE\_CASE\_TYPES  
V\_DELIVERY\_TYPES  
V\_DENIAL\_REASONS  
V\_DISQUAL\_REASONS  
V\_FINDING\_CODES  
V\_FSP\_REGIONAL\_OFFICES  
V\_FSP\_VIOLATION\_CODES  
V\_LEGAL\_FIRMS  
V\_LEGAL\_REPRESENTATIVES  
V\_MILESTONE\_TYPES  
V\_OWNER\_TYPES  
V\_PEER\_GROUPS  
V\_RISK\_LEVELS  
V\_SANCTION\_TYPES  
V\_STATUSES  
V\_SUSPENSION\_REASONS  
V\_VIOLATION\_ACTION  
V\_WHOLESALERS  
V\_WIC\_CODES  
V\_OWNERS  
V\_VENDORS  
O\_STAFF\_MEMBERS  
PROMPT EA3 updating :A\_O\_ORGANIZATIONAL\_UNITS.SM\_STAFF\_MEMBER\_ID TO NULL  
O\_ORGANIZATIONAL\_UNITS  
O\_ANNUAL\_WIC\_COST\_SUMMARIES  
F\_ANNUAL\_FACTORS  
F\_BUDGETS  
F\_CASE\_ASSIGNMENTS  
F\_CASELOADS  
F\_CASELOAD\_RESTRICTIONS  
F\_MFR\_CONTACTS  
F\_MANUFACTURERS  
I\_FI\_INVENTORIES  
I\_STOCK\_INVENTORIES  
C\_RESOLVED\_CLIENTS  
C\_RESOLUTIONS  
I\_FI\_REJECT\_REASONS  
S\_PRINTER\_ADDRESSES  
DEL\_STATEMENTS

### 1.31.7 EA3\_TAB\_UP.SQL

#### *Overview*

This script performs updates to the following tables:

A\_ACTIVITIES  
AAS\_CLASS\_CATEGORIES  
AAS\_ITEMS  
A\_APPT\_ITEMS  
A\_ATTEND\_STATUSES  
A\_OFFICE\_CLOSED  
A\_SERVICES  
I\_AGE\_RANGES  
C\_ANSWERS  
C\_ANSWER\_TYPES  
C\_BLOODWORK\_TYPES  
C\_BREAST\_PUMP\_REASONS  
C\_BREAST\_PUMP\_TYPES  
C\_CATEGORIES  
C\_CAT\_BLOOD\_FACTORS  
C\_CAT\_GOALS  
C\_CAT\_NUTR\_EDS  
C\_CAT\_REFERRALS  
C\_COMMUNICATION\_TYPES  
C\_CP\_MESSAGES  
C\_DIAGNOSES  
C\_DIETARY\_REQUIREMENTS  
C\_DIET\_NUTRIENT\_TYPES  
C\_DISABILITIES  
C\_EDUCATION\_LEVELS  
C\_ELEVATIONS  
C\_ETHNIC\_GROUPS  
C\_GOALS  
C\_HC\_PAYEES  
C\_HH\_QUESTIONS  
C\_HH\_RESPONSES  
C\_INCOME\_INTERVALS  
C\_INCOME\_LEVELS  
F\_POVERTY\_BASES  
C\_INCOME\_SOURCES  
C\_INCOME\_VERIFICATIONS  
C\_INFANT\_STATUSES  
C\_LANGUAGES  
C\_MARITAL\_STATUSES  
C\_NCHS\_CLASSIFICATIONS  
C\_NCHS\_TYPES  
C\_NCHS\_DATA

C\_NO\_CONTACT\_REASONS  
C\_NUTR\_ED\_MATERIALS  
C\_NUTR\_ED\_TOPICS  
C\_PICKUP\_INTERVALS  
C\_PILOT\_QUESTIONS  
C\_PILOT\_STUDIES  
C\_PRIORITIES  
C\_PROOF\_ADDRESSES  
C\_PROOF\_IDENTITIES  
C\_PS\_QUESTIONS  
C\_RACES  
C\_REASONS\_BF\_ENDED  
C\_RISK\_FACTORS  
C\_RISK\_FACTOR\_DIAGNOSES  
C\_RISK\_FACTOR\_TYPES  
C\_RF\_GOALS  
C\_RF\_NUTR\_EDS  
C\_RF\_REFERRALS  
C\_RF\_SERVICES  
C\_HH\_RESPONSE\_RFS  
C\_BMI\_DATA  
C\_BMI\_ANTHROPOMETRIC  
C\_BMI\_WEIGHT\_GAIN  
C\_SCHEDULE\_DAYS  
C\_SMOKING\_CHANGES  
C\_SMOKING\_STAGES  
C\_SOURCES\_HEALTH\_CARE  
C\_SYMPTOMS  
C\_TERM\_REASONS  
C\_TOPICS  
C\_VOTER\_REGISTRATIONS  
C\_CAT\_BLOODWORKS  
C\_DESIREABLE\_WEIGHTS  
C\_IMMUNIZATIONS\_NOT\_ASSESSED  
F\_CONTROLS  
F\_CASELOAD\_TYPES  
F\_FUND\_SOURCES  
F\_POVERTY\_LEVELS  
F\_WAIT\_LIST\_RESPONSES  
I\_BANK\_DISPOSITIONS  
I\_CATEGORY\_GROUPS  
I\_PACKAGES  
I\_PRODUCTS  
I\_REJECT\_REASONS  
I\_CONTAINERS  
I\_DISPOSITIONS  
I\_UNITS\_OF\_MEASURE  
I\_VOID\_REASONS  
I\_FOOD\_GROUPS  
I\_FOODS  
I\_FOOD\_DISTRIBUTIONS

I\_MAXIMUM\_FOODS  
I\_FOOD\_PACKAGES  
I\_CATEGORY\_GROUP\_PKGS  
I\_FOOD\_PACKAGE\_FOODS  
I\_PACKAGE\_DISTRIBUTIONS  
I\_WS\_FOOD\_GROUPS  
I\_FOOD\_INSTRUMENT\_TYPES  
I\_FOOD\_PACKAGE\_FI\_TYPES  
I\_FOOD\_INSTRUMENT\_FOODS  
C\_FOOD\_PKG\_RISK\_FACTORS  
O\_OUTREACH\_ORG\_TYPES  
O\_OUTREACH\_COMM\_TYPES  
O\_PROGRAMS  
O\_PROGRAM\_DATES  
O\_TITLE\_CATEGORIES  
O\_STAFF\_TITLES  
S\_CITIES  
S\_CONTACT\_METHODS  
S\_CONTACT\_TITLES  
S\_COUNTIES  
S\_PHONE\_TYPES  
S\_STATES  
S\_ZIPS  
S\_GEO\_LOCATIONS  
V\_ACTIVITY\_TYPES  
V\_APPLICATION\_MILESTONES  
V\_APPLICATION\_TYPES  
V\_BANK\_BRANCHES  
V\_CASE\_STATUSES  
V\_COLLECTION\_TYPES  
V\_COMMUNICATION\_TYPES  
V\_COMPLAINT\_SOURCE\_TYPES  
V\_COMPLAINT\_STATUSES  
V\_COMPLAINT\_SUBJECTS  
V\_COMPLIANCE\_CASE\_DESIGNATIONS  
V\_COMPLIANCE\_CASE\_TYPES  
V\_DELIVERY\_TYPES  
V\_DENIAL\_REASONS  
V\_DISQUAL\_REASONS  
V\_FINDING\_CODES  
V\_FSP\_REGIONAL\_OFFICES  
V\_FSP\_VIOLATION\_CODES  
V\_LEGAL\_FIRMS  
V\_LEGAL\_REPRESENTATIVES  
V\_MILESTONE\_TYPES  
V\_OWNER\_TYPES  
V\_PEER\_GROUPS  
V\_RISK\_LEVELS  
V\_SANCTION\_TYPES  
V\_STATUSES  
V\_SUSPENSION\_REASONS

V\_VIOLATION\_ACTION  
V\_WHOLESALEERS  
V\_WIC\_CODES  
V\_OWNERS  
V\_VENDORS  
O\_STAFF\_MEMBERS  
PROMPT EA3 updating :A\_O\_ORGANIZATIONAL\_UNITS.SM\_STAFF\_MEMBER\_ID TO  
NULL  
O\_ORGANIZATIONAL\_UNITS  
O\_ANNUAL\_WIC\_COST\_SUMMARIES  
F\_ANNUAL\_FACTORS  
F\_BUDGETS  
F\_CASE\_ASSIGNMENTS  
F\_CASELOADS  
F\_CASELOAD\_RESTRICTIONS  
F\_MFR\_CONTACTS  
F\_MANUFACTURERS  
I\_FI\_INVENTORIES  
I\_STOCK\_INVENTORIES  
C\_RESOLVED\_CLIENTS  
C\_RESOLUTIONS  
I\_FI\_REJECT\_REASONS  
S\_PRINTER\_ADDRESSES  
DEL\_STATEMENTS

#### 1.31.7.1 Inputs

The following temporary tables are used to make the updates:

A\_EOD\_CONTROLS  
A\_A\_ACTIVITIES  
A\_AAS\_CLASS\_CATEGORIES  
A\_AAS\_ITEMS  
A\_A\_APPT\_ITEMS  
A\_A\_ATTEND\_STATUSES  
A\_A\_OFFICE\_CLOSED  
A\_A\_SERVICES  
A\_I\_AGE\_RANGES  
A\_C\_ANSWERS  
A\_C\_ANSWER\_TYPES  
A\_C\_BLOODWORK\_TYPES  
A\_C\_BREAST\_PUMP\_REASONS  
A\_C\_BREAST\_PUMP\_TYPES  
A\_C\_CATEGORIES  
A\_C\_CAT\_BLOOD\_FACTORS  
A\_C\_CAT\_GOALS  
A\_C\_CAT\_NUTR\_EDS  
A\_C\_CAT\_REFERRALS

A\_C\_COMMUNICATION\_TYPES  
A\_C\_CP\_MESSAGES  
A\_C\_DIAGNOSES  
A\_C\_DIETARY\_REQUIREMENTS  
A\_C\_DIET\_NUTRIENT\_TYPES  
A\_C\_DISABILITIES  
A\_C\_EDUCATION\_LEVELS  
A\_C\_ELEVATIONS  
A\_C\_ETHNIC\_GROUPS  
A\_C\_GOALS  
A\_C\_HC\_PAYEES  
A\_C\_HH\_QUESTIONS  
A\_C\_HH\_RESPONSES  
A\_C\_INCOME\_INTERVALS  
A\_C\_INCOME\_LEVELS  
A\_F\_POVERTY\_BASES  
A\_C\_INCOME\_SOURCES  
A\_C\_INCOME\_VERIFICATIONS  
A\_C\_INFANT\_STATUSES  
A\_C\_LANGUAGES  
A\_C\_MARITAL\_STATUSES  
A\_C\_NCHS\_CLASSIFICATIONS  
A\_C\_NCHS\_TYPES  
A\_C\_NCHS\_DATA  
A\_C\_NO\_CONTACT\_REASONS  
A\_C\_NUTR\_ED\_MATERIALS  
A\_C\_NUTR\_ED\_TOPICS  
A\_C\_PICKUP\_INTERVALS  
A\_C\_PILOT\_QUESTIONS  
A\_C\_PILOT\_STUDIES  
A\_C\_PRIORITIES  
A\_C\_PROOF\_ADDRESSES  
A\_C\_PROOF\_IDENTITIES  
A\_C\_PS\_QUESTIONS  
A\_C\_RACES  
A\_C\_REASONS\_BF\_ENDED  
A\_C\_RESOLUTIONS  
A\_C\_RESOLVED\_CLIENTS  
A\_C\_RISK\_FACTORS  
A\_C\_RISK\_FACTOR\_DIAGNOSES  
A\_C\_RISK\_FACTOR\_TYPES  
A\_C\_RF\_GOALS  
A\_C\_RF\_NUTR\_EDS  
A\_C\_RF\_REFERRALS  
A\_C\_RF\_SERVICES  
A\_C\_HH\_RESPONSE\_RFS  
A\_C\_BMI\_DATA  
A\_C\_BMI\_ANTHROPOMETRIC  
A\_C\_BMI\_WEIGHT\_GAIN  
A\_C\_SCHEDULE\_DAYS  
A\_C\_SMOKING\_CHANGES

A\_C\_SMOKING\_STAGES  
A\_C\_SOURCES\_HEALTH\_CARE  
A\_C\_SYMPTOMS  
A\_C\_TERM\_REASONS  
A\_C\_TOPICS  
A\_C\_VOTER\_REGISTRATIONS  
A\_C\_CAT\_BLOODWORKS  
A\_C\_DESIREABLE\_WEIGHTS  
PROMPT EA1 inserting :A\_C\_IMMUNIZATIONS\_NOT\_ASSESSED  
A\_F\_CONTROLS  
A\_F\_CASELOAD\_TYPES  
A\_F\_FUND\_SOURCES  
A\_F\_POVERTY\_LEVELS  
A\_F\_WAIT\_LIST\_RESPONSES  
A\_I\_BANK\_DISPOSITIONS  
A\_I\_CATEGORY\_GROUPS  
A\_I\_PACKAGES  
A\_I\_PRODUCTS  
A\_I\_REJECT\_REASONS  
A\_I\_CONTAINERS  
A\_I\_DISPOSITIONS  
A\_I\_UNITS\_OF\_MEASURE  
A\_I\_VOID\_REASONS  
A\_I\_FOOD\_GROUPS  
A\_I\_FOODS  
A\_I\_FOOD\_DISTRIBUTIONS  
A\_I\_MAXIMUM\_FOODS  
A\_I\_FOOD\_PACKAGES  
A\_I\_CATEGORY\_GROUP\_PKGS  
A\_I\_FOOD\_PACKAGE\_FL\_TYPES  
A\_I\_FOOD\_PACKAGE\_FOODS  
A\_I\_PACKAGE\_DISTRIBUTIONS  
A\_I\_WS\_FOOD\_GROUPS  
A\_I\_FOOD\_INSTRUMENT\_TYPES  
A\_I\_FOOD\_INSTRUMENT\_FOODS  
A\_C\_FOOD\_PKG\_RISK\_FACTORS  
A\_O\_OUTREACH\_ORG\_TYPES  
A\_O\_OUTREACH\_COMM\_TYPES  
--A\_O\_OUTREACH\_ORGANIZATIONS  
--A\_O\_OUTREACH\_ORG\_PHONES  
--A\_O\_OUTREACH\_COMMS  
A\_O\_PROGRAMS  
A\_O\_PROGRAM\_DATES  
A\_O\_TITLE\_CATEGORIES  
A\_O\_STAFF\_TITLES  
A\_S\_CITIES  
A\_S\_CONTACT\_METHODS  
A\_S\_CONTACT\_TITLES  
A\_S\_COUNTIES  
A\_S\_PHONE\_TYPES  
A\_S\_STATES

A\_S\_ZIPS  
A\_S\_GEO\_LOCATIONS  
A\_V\_ACTIVITY\_TYPES  
A\_V\_APPLICATION\_MILESTONES  
A\_V\_APPLICATION\_TYPES  
A\_V\_BANK\_BRANCHES  
A\_V\_CASE\_STATUSES  
A\_V\_COLLECTION\_TYPES  
A\_V\_COMMUNICATION\_TYPES  
A\_V\_COMPLAINT\_SOURCE\_TYPES  
A\_V\_COMPLAINT\_STATUSES  
A\_V\_COMPLAINT\_SUBJECTS  
A\_V\_COMPLIANCE\_CASE\_DESIGNATNS  
A\_V\_COMPLIANCE\_CASE\_TYPES  
A\_V\_DELIVERY\_TYPES  
A\_V\_DENIAL\_REASONS  
A\_V\_DISQUAL\_REASONS  
A\_V\_FINDING\_CODES  
A\_V\_FSP\_REGIONAL\_OFFICES  
A\_V\_FSP\_VIOLATION\_CODES  
A\_V\_LEGAL\_FIRMS  
A\_V\_LEGAL\_REPRESENTATIVES  
A\_V\_MILESTONE\_TYPES  
A\_V\_OWNER\_TYPES  
A\_V\_PEER\_GROUPS  
A\_V\_RISK\_LEVELS  
A\_V\_SANCTION\_TYPES  
A\_V\_STATUSES  
A\_V\_SUSPENSION\_REASONS  
A\_V\_VIOLATION\_ACTION  
A\_V\_WHOLESALEERS  
A\_V\_WIC\_CODES  
A\_V\_OWNERS  
A\_V\_VENDORS  
--A\_O\_STAFF\_MEMBERS  
A\_O\_ORGANIZATIONAL\_UNITS  
A\_O\_ANNUAL\_WIC\_COST\_SUMMARIES  
A\_F\_ANNUAL\_FACTORS  
A\_F\_BUDGETS  
A\_F\_CASE\_ASSIGNMENTS  
A\_F\_CASELOADS  
A\_F\_CASELOAD\_RESTRICTIONS  
A\_F\_MFR\_CONTACTS  
A\_F\_MANUFACTURERS  
A\_I\_FI\_INVENTORIES  
A\_I\_STOCK\_INVENTORIES  
A\_I\_FOOD\_INSTRUMENTS  
A\_I\_FI\_REJECT\_REASONS  
S\_PRINTER\_ADDRESSES  
A\_DEL\_STATEMENTS

**1.31.7.2 Selection**

WHERE date\_modified > (SELECT good\_proc\_date FROM EODADM.a\_eod\_controls))

**1.31.7.3 Processing**

Update existing records on the tables listed in the Inputs section (above). Selects records from the eodadm.a\_\* tables that have their date\_modified after the last successful EOD run, matches them to a record on the agency tables, and updates the agency tables with the eodadm.a\_\* record contents:

```
update table
  set (...columns...) =
    (select ...columns...
     from eodadm.a_table
     where date_modified > (last EOD)
     and table.pk = eodadm.a_table.pk )
  where (pk) in
    (select pk from eodadm.a_table where date_modified > last EOD)
```

where table is the name of a table, such as c\_goals  
 columns is the matching names of the columns in the table  
 pk is the primary key to a table

For example:

```
Update c_goals
  set (description,date_created,created_by,date_modified,modified_by,note) = (Select
  description,date_created,created_by,date_modified,modified_by,note
  from eodadm.a_c_goals
  where date_modified > (last EOD)
  and c_goals.goal_code = eodadm.a_c_goals.goal_code )
  where (c_goals.goal_code) in
    (select goal_code from eodadm.a_c_goals where trunc(date_modified) > (last
EOD))
```

**1.31.7.4 Outputs**

A\_ACTIVITIES  
 AAS\_CLASS\_CATEGORIES  
 AAS\_ITEMS  
 A\_APPT\_ITEMS  
 A\_ATTEND\_STATUSES  
 A\_OFFICE\_CLOSED  
 A\_SERVICES

I\_AGE\_RANGES  
C\_ANSWERS  
C\_ANSWER\_TYPES  
C\_BLOODWORK\_TYPES  
C\_BREAST\_PUMP\_REASONS  
C\_BREAST\_PUMP\_TYPES  
C\_CATEGORIES  
C\_CAT\_BLOOD\_FACTORS  
C\_CAT\_GOALS  
C\_CAT\_NUTR\_EDS  
C\_CAT\_REFERRALS  
C\_COMMUNICATION\_TYPES  
C\_CP\_MESSAGES  
C\_DIAGNOSES  
C\_DIETARY\_REQUIREMENTS  
C\_DIET\_NUTRIENT\_TYPES  
C\_DISABILITIES  
C\_EDUCATION\_LEVELS  
C\_ELEVATIONS  
C\_ETHNIC\_GROUPS  
C\_GOALS  
C\_HC\_PAYEES  
C\_HH\_QUESTIONS  
C\_HH\_RESPONSES  
C\_INCOME\_INTERVALS  
C\_INCOME\_LEVELS  
F\_POVERTY\_BASES  
C\_INCOME\_SOURCES  
C\_INCOME\_VERIFICATIONS  
C\_INFANT\_STATUSES  
C\_LANGUAGES  
C\_MARITAL\_STATUSES  
C\_NCHS\_CLASSIFICATIONS  
C\_NCHS\_TYPES  
C\_NCHS\_DATA  
C\_NO\_CONTACT\_REASONS  
C\_NUTR\_ED\_MATERIALS  
C\_NUTR\_ED\_TOPICS  
C\_PICKUP\_INTERVALS  
C\_PILOT\_QUESTIONS  
C\_PILOT\_STUDIES  
C\_PRIORITIES  
C\_PROOF\_ADDRESSES  
C\_PROOF\_IDENTITIES  
C\_PS\_QUESTIONS  
C\_RACES  
C\_REASONS\_BF\_ENDED  
C\_RISK\_FACTORS  
C\_RISK\_FACTOR\_DIAGNOSES  
C\_RISK\_FACTOR\_TYPES  
C\_RF\_GOALS

C\_RF\_NUTR\_EDS  
C\_RF\_REFERRALS  
C\_RF\_SERVICES  
C\_HH\_RESPONSE\_RFS  
C\_BMI\_DATA  
C\_BMI\_ANTHROPOMETRIC  
C\_BMI\_WEIGHT\_GAIN  
C\_SCHEDULE\_DAYS  
C\_SMOKING\_CHANGES  
C\_SMOKING\_STAGES  
C\_SOURCES\_HEALTH\_CARE  
C\_SYMPTOMS  
C\_TERM\_REASONS  
C\_TOPICS  
C\_VOTER\_REGISTRATIONS  
C\_CAT\_BLOODWORKS  
C\_DESIREABLE\_WEIGHTS  
C\_IMMUNIZATIONS\_NOT\_ASSESSED  
F\_CONTROLS  
F\_CASELOAD\_TYPES  
F\_FUND\_SOURCES  
F\_POVERTY\_LEVELS  
F\_WAIT\_LIST\_RESPONSES  
I\_BANK\_DISPOSITIONS  
I\_CATEGORY\_GROUPS  
I\_PACKAGES  
I\_PRODUCTS  
I\_REJECT\_REASONS  
I\_CONTAINERS  
I\_DISPOSITIONS  
I\_UNITS\_OF\_MEASURE  
I\_VOID\_REASONS  
I\_FOOD\_GROUPS  
I\_FOODS  
I\_FOOD\_DISTRIBUTIONS  
I\_MAXIMUM\_FOODS  
I\_FOOD\_PACKAGES  
I\_CATEGORY\_GROUP\_PKGS  
I\_FOOD\_PACKAGE\_FOODS  
I\_PACKAGE\_DISTRIBUTIONS  
I\_WS\_FOOD\_GROUPS  
I\_FOOD\_INSTRUMENT\_TYPES  
I\_FOOD\_PACKAGE\_FI\_TYPES  
I\_FOOD\_INSTRUMENT\_FOODS  
C\_FOOD\_PKG\_RISK\_FACTORS  
O\_OUTREACH\_ORG\_TYPES  
O\_OUTREACH\_COMM\_TYPES  
O\_PROGRAMS  
O\_PROGRAM\_DATES  
O\_TITLE\_CATEGORIES  
O\_STAFF\_TITLES

S\_CITIES  
S\_CONTACT\_METHODS  
S\_CONTACT\_TITLES  
S\_COUNTIES  
S\_PHONE\_TYPES  
S\_STATES  
S\_ZIPS  
S\_GEO\_LOCATIONS  
V\_ACTIVITY\_TYPES  
V\_APPLICATION\_MILESTONES  
V\_APPLICATION\_TYPES  
V\_BANK\_BRANCHES  
V\_CASE\_STATUSES  
V\_COLLECTION\_TYPES  
V\_COMMUNICATION\_TYPES  
V\_COMPLAINT\_SOURCE\_TYPES  
V\_COMPLAINT\_STATUSES  
V\_COMPLAINT\_SUBJECTS  
V\_COMPLIANCE\_CASE\_DESIGNATIONS  
V\_COMPLIANCE\_CASE\_TYPES  
V\_DELIVERY\_TYPES  
V\_DENIAL\_REASONS  
V\_DISQUAL\_REASONS  
V\_FINDING\_CODES  
V\_FSP\_REGIONAL\_OFFICES  
V\_FSP\_VIOLATION\_CODES  
V\_LEGAL\_FIRMS  
V\_LEGAL\_REPRESENTATIVES  
V\_MILESTONE\_TYPES  
V\_OWNER\_TYPES  
V\_PEER\_GROUPS  
V\_RISK\_LEVELS  
V\_SANCTION\_TYPES  
V\_STATUSES  
V\_SUSPENSION\_REASONS  
V\_VIOLATION\_ACTION  
V\_WHOLESALEERS  
V\_WIC\_CODES  
V\_OWNERS  
V\_VENDORS  
O\_STAFF\_MEMBERS  
PROMPT EA3 updating :A\_O\_ORGANIZATIONAL\_UNITS.SM\_STAFF\_MEMBER\_ID TO NULL  
O\_ORGANIZATIONAL\_UNITS  
O\_ANNUAL\_WIC\_COST\_SUMMARIES  
F\_ANNUAL\_FACTORS  
F\_BUDGETS  
F\_CASE\_ASSIGNMENTS  
F\_CASELOADS  
F\_CASELOAD\_RESTRICTIONS  
F\_MFR\_CONTACTS  
F\_MANUFACTURERS

I\_FI\_INVENTORIES  
I\_STOCK\_INVENTORIES  
C\_RESOLVED\_CLIENTS  
C\_RESOLUTIONS  
I\_FI\_REJECT\_REASONS  
S\_PRINTER\_ADDRESSES  
DEL\_STATEMENTS

### 1.31.8 EA\_TRGON.SQL

#### Overview

Sets database triggers on, so that triggers and foreign keys are re-enabled.

#### 1.31.8.1 Inputs

N/A

#### 1.31.8.2 Selection

N/A

#### 1.31.8.3 Processing

Alter the following tables with 'ENABLE ALL TRIGGERS'

F\_ANNUAL\_FACTORS  
I\_FOOD\_INSTRUMENT\_TYPES  
C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
C\_CLIENT\_SERVICES

Alter the following tables with 'ENABLE CONSTRAINT'

I\_FOOD\_INSTRUMENT\_TYPES with constraint IFIT\_IFIT\_FK  
I\_FOOD\_PACKAGE\_FI\_TYPES with constraint IFPFT\_IFPFT\_FK  
F\_MANUFACTURERS with constraint MAN\_FMC\_FK  
F\_MFR\_CONTACTS with constraint FMC\_MAN\_FK  
C\_RESOLUTIONS with constraint CR1\_CC\_FK

#### 1.31.8.4 Outputs

N/A

**1.32** Clinic serial number replenish**1.32.1** EC4\_CL\_SER\_REP.SQL**Overview**

During the end of day process the system automatically replenishes serial numbers for the clinics with the number of serial numbers at 10% or less of the number defined by the static factors, “small clinic FI’s”, “med clinic FI’s”, and “large clinic FI’s”.

**1.32.1.1** Inputs

I\_FI\_INVENTORIES  
 F\_CONTROLS  
 O\_ORGANIZATIONAL\_UNITS

**1.32.1.2** Selection

MAX(I\_FI\_INVENTORIES.END\_NO) for each clinic where  
 (I\_FI\_INVENTORIES.END\_NO - I\_FI\_INVENTORIES.LAST\_FI\_NO\_USED ≤ (.1) \*  
 F\_CONTROLS.SMALL\_CLINIC\_FIS and O\_ORGANIZATIONAL\_UNITS.ORG\_SIZE =  
 ‘SMALL’) OR  
 (I\_FI\_INVENTORIES.END\_NO - I\_FI\_INVENTORIES.LAST\_FI\_NO\_USED ≤ (.1) \*  
 F\_CONTROLS.MED\_CLINIC\_FIS and O\_ORGANIZATIONAL\_UNITS.ORG\_SIZE =  
 ‘MEDIUM’) OR  
 (I\_FI\_INVENTORIES.END\_NO - I\_FI\_INVENTORIES.LAST\_FI\_NO\_USED ≤ (.1) \*  
 F\_CONTROLS.LARGE\_CLINIC\_FIS and O\_ORGANIZATIONAL\_UNITS.ORG\_SIZE =  
 ‘LARGE’)

**1.32.1.3** Processing

For each I\_FI\_INVENTORIES record meeting the selection criteria  
 Loop

```

If O_ORGANIZATIONAL_UNITS.ORG_SIZE = ‘SMALL’
    number_of_fis = F_CONTROLS.SMALL_CLINIC_FIS
Elsif O_ORGANIZATIONAL_UNITS.ORG_SIZE = ‘MEDIUM’
    number_of_fis = F_CONTROLS.MED_CLINIC_FIS
Else
    number_of_fis = F_CONTROLS.LARGE_CLINIC_FIS
End if
  
```

Insert into I\_FI\_INVENTORIES at the central database setting the columns as follows:

START_NO	= MAX(I_FI_INVENTORIES.END_NO) + 1
END_NO	= START_NO + number_of_fis
NO_ISSUED	= number_of_fis
LAST_FI_NO_USED	= START_NO - 1

Copy this I\_FI\_INVENTORIES record to the L/A database of the clinic

End Loop

#### 1.32.1.4 Outputs

I\_FI\_INVENTORIES

### 1.33 Update caseload assignment information from the clinics

#### 1.33.1 EA3\_SYNC\_LA\_ASSIGNMENT.SQL

##### *Overview*

This script is used at local agencies for end of day processing to synchronize the caseload assignment data that has been manually re-allocated by the staff. This script needs to be executed before the end of day administrative tables are deleted, as it uses the end of day administrative f\_case\_assignment table to insert and/or update data into f\_case\_assignments at the local agency.

##### 1.33.1.1 Inputs

f\_case\_assignments  
a\_f\_case\_assignments

##### 1.33.1.2 Processing

For all F\_CASE\_ASSIGNMENTS records at the local agencies where the primary keys are equal to those in EODADM.A\_F\_CASE\_ASSIGNMENTS (a temporary table holding Caseload Assignments that have changed at the state level)

Update F\_CASE\_ASSIGNMENTS

Set Assigned, Alloc Factor, Date Created, Created By, Date Modified, and Modified By to their equivalents in EODADM.A\_F\_CASE\_ASSIGNMENTS.

For new case assignments insert new record in F\_CASE\_ASSIGNMENTS table.

##### 1.33.1.3 Outputs

Log file of which assignments have changed. Log file is EA3\_SYNC\_LA\_ASSIGNMENT.LOG

f\_case\_assignments

#### 1.33.2 EA3.SQL

##### *Overview*

Runs SQL scripts for EA3.BAT.

**1.33.2.1** Inputs

eod\_controls  
env\_variables

**1.33.2.2** Selection

env\_variables  
    where code = 'EOD\_BY'

**1.33.2.3** Processing

update eod\_controls, setting TO\_DATE to the system date  
update env\_variables, setting env\_value to 'EA3' where code = 'EOD\_BY'

run SQL script EA3\_PROC\_FI  
run SQL script EA3\_PRG\_ARCHV  
run SQL script <AGENCY\_CODE>.SQL

**1.33.2.4** Outputs

end\_controls  
env\_variables

Log file: EA3\_SQL.LOG

**1.34** Send redemption/rejection information to agencies**1.34.1** EA3\_PROC\_FI.SQL**Overview**

Process the Food instrument's received from Central. Update the food instrument data of the local agency with the food instrument's received from the central server.

**1.34.1.1** Inputs

a\_i\_food\_instruments  
a\_i\_fi\_reject\_reasons  
i\_food\_instruments

**1.34.1.2** Selection

For i\_food\_instruments:

WHERE serial\_number = eodadm.a\_i\_food\_instruments.serial\_number

**1.34.1.3** Processing

If a corresponding i\_food\_instruments record is not found for the a\_i\_food\_instruments,  
then write error message

Elseif i\_food\_instruments.date\_modified = a\_i\_food\_instruments.date\_modified  
and i\_food\_instruments.date\_created = a\_i\_food\_instruments.date\_created (records are  
same)  
then write error message.

Else Updates food instrument table from the food instrument information sent from the Central database:

```
UPDATE i_food_instruments
SET   ou_seq_id=eodadm.a_i_food_instruments.ou_seq_id,
      revalidation_code=eodadm.a_i_food_instruments.revalidation_code,
      compliance_buy_flag=eodadm.a_i_food_instruments.compliance_buy_flag,
      ifit_fi_type_code=eodadm.a_i_food_instruments.ifit_fi_type_code,
      idis_disposition_code=eodadm.a_i_food_instruments.idis_disposition_code,
      vcc_case_id=eodadm.a_i_food_instruments.vcc_case_id,
      ifi_serial_number=eodadm.a_i_food_instruments.ifi_serial_number,
      bd_bank_disposition_code=eodadm.a_i_food_instruments.bd_bank_disposition_code,
      cfpp_effective_date=eodadm.a_i_food_instruments.cfpp_effective_date,
      cfpp_cc_client_id=eodadm.a_i_food_instruments.cfpp_cc_client_id,
```

```

ma_aty_act_type_code=eodadm.a_i_food_instruments.ma_aty_act_type_code,
ma_seq_nbr=eodadm.a_i_food_instruments.ma_seq_nbr,
ivr_void_reason_code=eodadm.a_i_food_instruments.ivr_void_reason_code,
cfpp_ifp_food_package_id=eodadm.a_i_food_instruments.cfpp_ifp_food_package_id,
ven_id=eodadm.a_i_food_instruments.ven_id,
ma_ven_id=eodadm.a_i_food_instruments.ma_ven_id,
cc_client_id=eodadm.a_i_food_instruments.cc_client_id,
sm_staff_member_id=eodadm.a_i_food_instruments.sm_staff_member_id,
fpo_po_number=eodadm.a_i_food_instruments.fpo_po_number,
stale_date=eodadm.a_i_food_instruments.stale_date,
issue_method=eodadm.a_i_food_instruments.issue_method,
est_reb_value=eodadm.a_i_food_instruments.est_reb_value,
first_date_to_use=eodadm.a_i_food_instruments.first_date_to_use,
last_date_to_use=eodadm.a_i_food_instruments.last_date_to_use,
maximum_amt=eodadm.a_i_food_instruments.maximum_amt,
redemption_amt=eodadm.a_i_food_instruments.redemption_amt,
cleared_date=eodadm.a_i_food_instruments.cleared_date,
reject_date=eodadm.a_i_food_instruments.reject_date,
requested_amt=eodadm.a_i_food_instruments.requested_amt,
approved_amt=eodadm.a_i_food_instruments.approved_amt,
approval_date=eodadm.a_i_food_instruments.approval_date,
issue_date=eodadm.a_i_food_instruments.issue_date,
void_date=eodadm.a_i_food_instruments.void_date,
obligated_amt=eodadm.a_i_food_instruments.obligated_amt,
formula_return_flag=eodadm.a_i_food_instruments.formula_return_flag,
allocated_amt=eodadm.a_i_food_instruments.allocated_amt,
missing_issuance_flag=eodadm.a_i_food_instruments.missing_issuance_flag,
process_date=eodadm.a_i_food_instruments.process_date,
date_modified=eodadm.a_i_food_instruments.date_modified,
modified_by=eodadm.a_i_food_instruments.modified_by,
note=eodadm.a_i_food_instruments.note
received_by_state=eodadm.a_i_food_instruments.received_by_state
cat_category_code=eodadm.a_i_food_instruments.cat_category_code
VCC_VEN_ID=eodadm.a_i_food_instruments.VCC_VEN_ID
CP2_ID=eodadm.a_i_food_instruments.CP2_ID
buy_number=eodadm.a_i_food_instruments.buy_number
issue_month=eodadm.a_i_food_instruments.issue_month
WHERE serial_number = eodadm.a_i_food_instruments.serial_number

```

If this FI is revalidated from Central then delete the corresponding reject reasons records from I\_FI\_REJECT\_REASONS:

```

IF (eodadm.a_i_food_instruments.reject_date IS NULL) AND
  (ifi_rec.reject_date IS NOT NULL) THEN
  BEGIN
    DELETE FROM i_fi_reject_reasons
    WHERE ifi_serial_number = eodadm.a_i_food_instruments.serial_number
  
```

If this FI is rejected, then create respective records in the I\_FI\_REJECT\_REASONS:

```

IF (eodadm.a_i_food_instruments.reject_date IS NOT NULL) THEN
  BEGIN

```

```

DELETE FROM i_fi_reject_reasons
WHERE ifi_serial_number = eodadm.a_i_food_instruments.serial_number

INSERT INTO i_fi_reject_reasons(
    ifi_serial_number,
    irr_reject_reason_code,
    date_created,
    created_by,
    date_modified,
    modified_by)
(SELECT
    ifi_serial_number,
    irr_reject_reason_code,
    date_created,
    created_by,
    date_modified,
    modified_by
FROM
    eodadm.a_i_fi_reject_reasons
WHERE ifi_serial_number = eodadm.a_i_food_instruments.serial_number)

```

Delete the record from the Central-to-Agency transfer file, as it is no longer required:

```

DELETE FROM eodadm.a_i_food_instruments
WHERE serial_number=eodadm.a_i_food_instruments.serial_number;

```

Also delete reject reasons record from the Central-to-Agency transfer file:

```

DELETE FROM eodadm.a_i_fi_reject_reasons
WHERE ifi_serial_number = eodadm.a_i_food_instruments.serial_number

```

#### 1.34.1.4 Outputs

If the check is not found at the local agency then flash a message:

```

IF check record is not found THEN
    success := 'N';
    Output('Problem with check : ' || eodadm.a_i_food_instruments.serial_number);
    Output('Problem finding the record here, Cannot be processed further');
ELSE

```

If the check is found, but nothing has been modified (duplicate), then discard:

```

    IF TRUNC(eodadm.a_i_food_instruments.process_date) = TRUNC(ifi_rec.process_date)
    AND TRUNC(eodadm.a_i_food_instruments.date_modified) = TRUNC(ifi_rec.date_modified)
THEN success := 'N';

```

```

    OUTPUT('Duplicate Check : ' || eodadm.a_i_food_instruments.serial_number);

```

```

ELSE

```

```

    EXCEPTION

```

```

    WHEN OTHERS THEN

```

```

        OUTPUT('Serial Number : ' || eodadm.a_i_food_instruments.serial_number);

```

```
OUTPUT('Problem in Updating the Check');
    success := 'N';

EXCEPTION
WHEN NO_DATA_FOUND THEN
    NULL;
WHEN OTHERS THEN
    success := 'N';
    OUTPUT('Serial Number : ' ||
eodadm.a_i_food_instruments.serial_number);
    OUTPUT('Problem in deleting the FI REJECT REASONS records');

EXCEPTION
WHEN NO_DATA_FOUND THEN
    NULL;
WHEN OTHERS THEN
    success := 'N';
    OUTPUT('Serial Number : ' ||
eodadm.a_i_food_instruments.serial_number);
    OUTPUT('Problem in inserting the FI REJECT REASONS records');
```

i\_food\_instruments  
i\_fi\_reject\_reasons

### 1.35 Transmit archival clients and update records at agencies

#### 1.35.1 EA3\_PRG\_ARCHV.SQL

##### *Overview*

Purge archived client records from c\_clients where a record exists in r\_archived\_clients for that client ID. The deletes have been ordered from the smallest level child up to the parent (c\_clients). Archived vendors do not get purged here as that process is handled through delete triggers.

##### 1.35.1.1 Inputs

r\_archived\_clients

##### 1.35.1.2 Selection

All records in the r\_archived\_clients table.

##### 1.35.1.3 Processing

Purges archived client records from the following tables where the client\_ID matches the client\_ID in r\_archived\_clients.

LEVEL 5:

A\_APPT\_TOPIC\_MATERIALS

LEVEL 4:

A\_APPT\_TOPICS

A\_APPT\_APPT\_ITEMS

AAS\_APPT\_CLIENTS

C\_CLIENT\_SVC\_NE\_MATERIALS

F\_WAIT\_LIST\_CONTACTS

C\_DIET\_NUTRIENTS

C\_INCOMES

LEVEL 3:

A\_APPOINTMENTS  
C\_I\_C\_HH\_REASONS\_BFENDS  
C\_W\_HH\_REASONS\_BFENDS  
C\_FOOD\_BOX\_DIST  
C\_INCOME\_HISTORIES  
C\_CLIENT\_NE\_TOPICS  
C\_WOMAN\_MEDICALS  
C\_BLOODWORK\_DATA  
C\_CERT\_TERM\_REASONS  
C\_B\_N\_HEALTHS  
C\_FOOD\_PACKAGE\_PRESCRIPTIONS  
F\_WAIT\_LISTS  
C\_P\_HEALTHS  
C\_INFANT\_CHILD\_MEDICALS  
C\_DIETARY\_ASSESSMENTS  
C\_HEALTH\_RISK\_FACTORS  
C\_I\_C\_HEALTHS  
C\_ECONOMIC\_UNIT\_MEMBERS  
C\_PEER\_RBFENDS  
C\_CONTACTS  
C\_P\_COUNS\_NOTES  
I\_BF\_VCHR\_ITEMS

## LEVEL 2:

I\_BF\_PROMO\_VCHRS  
V\_COMPLIANCE\_CASE\_CLIENTS  
C\_CLIENT\_SERVICES  
C\_CERTIFICATIONS  
C\_TRANSFER\_HISTORIES  
C\_IMMUNIZATIONS  
C\_CLIENT\_REFERRALS  
C\_CLIENT\_COMMUNICATIONS  
C\_RESOLUTIONS  
C\_CLIENT\_NOTES  
C\_CLIENT\_PROGS  
C\_CLIENT\_GROUPS  
C\_INCOME\_HISTORIES  
C\_INFANT\_DATA  
C\_CLIENT\_GOALS  
C\_CERT\_PEER\_COUNSELS  
C\_WOMAN\_MEDICALS  
C\_W\_HEALTH  
C\_INFANT\_CHILD\_MEDICALS  
C\_I\_C\_HEALTHS  
C\_BLOODWORK\_DATA  
C\_PREV\_FAMILIES  
C\_PREV\_NAMES  
C\_ALLERGY\_FOODS  
C\_MORE\_ETHNIC\_GROUPS

## LEVEL 1:

AAS\_APPT\_CLIENTS  
AAS\_APPT\_ITEMS  
AAS\_APPOINTMENTS  
AAS\_CLASS\_FAMILIES  
C\_CLIENTS  
C\_INCOMES  
C\_ECONOMIC\_UNIT\_MEMBERS  
C\_INCOMES\_HISTORIES  
DEL\_STATEMENTS

**1.35.1.4** Outputs

N/A

**1.35.2 EA3\_RETRIEVE.SQL*****Overview***

Run scripts EA3\_INS\_RETR\_CLI.SQL and EA3\_INS\_RETR\_VEN.SQL to retrieve clients and vendors who were archived and then requested to be put back on the system.

**1.35.2.1 Inputs**

N/A

**1.35.2.2 Selection**

N/A

**1.35.2.3 Processing**

N/A

Run scripts EA3\_INS\_RETR\_CLI.SQL and EA3\_INS\_RETR\_VEN.SQL

**1.35.2.4 Outputs**

### 1.35.3 EA3\_INS\_RETR\_CLI.SQL

#### *Overview*

This script performs inserts from the temporary tables into all client tables.

r\_c\_clients  
r\_c\_bloodwork\_data  
r\_c\_w\_healths  
r\_c\_w\_hh\_reasons\_bfends  
r\_c\_certifications  
r\_c\_cert\_peer\_counsels  
r\_c\_cert\_term\_reasons  
r\_c\_client\_communications  
r\_c\_client\_goals  
r\_c\_client\_ne\_topics  
r\_c\_client\_notes  
r\_c\_client\_progs  
r\_c\_client\_referrals  
r\_c\_client\_services  
r\_c\_client\_svc\_ne\_materials  
r\_c\_contacts  
r\_c\_dietary\_assessments  
r\_c\_diet\_nutrients  
r\_c\_economic\_unit\_members  
r\_c\_family\_economic\_units  
r\_c\_family\_phones  
r\_c\_fam\_referrals  
r\_c\_feu\_communications  
r\_c\_health\_risk\_factors  
r\_c\_incomes  
r\_c\_income\_histories  
r\_c\_infant\_child\_medicals  
r\_c\_infant\_data  
r\_c\_i\_c\_healths  
r\_c\_i\_c\_hh\_reasons\_bfends  
r\_c\_peer\_rbfends  
r\_c\_p\_couns\_notes  
r\_c\_p\_healths  
r\_c\_resolutions  
r\_c\_transfer\_histories  
r\_c\_woman\_medicals  
r\_s\_client\_archives  
r\_c\_smoking\_histories  
r\_c\_prev\_families  
r\_c\_more\_ethnic\_groups  
r\_c\_prev\_names  
r\_c\_BF\_promo\_issuances

r\_c\_allergy\_foods  
r\_c\_PS\_responses  
r\_c\_client\_BP\_issuances  
r\_c\_resolved\_clients  
r\_c\_cert\_nutr\_eds  
r\_c\_food\_box\_dists  
r\_c\_food\_package\_prescriptions

### 1.35.3.1 Inputs

r\_s\_client\_archives

### 1.35.3.2 Selection

where local\_agency in  
(select env ('AGENCY\_SEQ') from dual)

### 1.35.3.3 Processing

This calls database function ins\_arch\_cli which recreates all of the client's information. It returns a "0" if successful, a "1" if the client already exists, a "2" if no information exists, and a "3" for other errors.

#### 1.35.3.4 Outputs

```
if ret_val = 0 then
    commit;
    OUTPUT('Successfully Retrieved Client '||to_char(cl_id));
else rollback;

if ret_val = 1 then
    OUTPUT('Client '||to_char(cl_id)||' already exists, Client not Processed' );
elseif ret_val = 2 then
    OUTPUT('Client '||to_char(cl_id)||' does not exist in Temporary Tables, Client not
    Processed' );

    elseif ret_val = 3 then
        OUTPUT('Client '||to_char(cl_id)||' generated Oracle Error, Client not Processed'
        );
end if
```

#### 1.35.4 EA3\_INS\_RETR\_VEN.SQL

##### *Overview*

This script performs inserts from the temporary tables into all vendor tables.

##### 1.35.4.1 Inputs

r\_s\_vendor\_archives

##### 1.35.4.2 Selection

N/A

##### 1.35.4.3 Processing

This calls database function ins\_arch\_ven which recreates all of the vendor's information. It returns a "0" if successful, a "1" if the vendor already exists, a "2" if no information exists, and a "3" for other errors.

##### 1.35.4.4 Outputs

```
if ret_val = 0 then
    commit;
```

```
        OUTPUT('Successfully Retrieved Vendor '||to_char(ven_id));
else
    rollback;

if ret_val = 1 then
    OUTPUT('Vendor '||to_char(ven_id)||' already exists, Vendor not Processed' );

    elsif ret_val = 2 then
        OUTPUT('Vendor '||to_char(ven_id)||' does not exist in Temporary Tables, Vendor not
        Processed' );

        elsif ret_val = 3 then
            OUTPUT('Vendor '||to_char(ven_id)||' generated Oracle Error, Vendor not
            Processed' );
end if;
```

## 1.36 Print out status logs of the process

### 1.36.1 EA\_PRINT.BAT

#### *Overview*

This batch file produces log files by agency for the following processes:

- \* Insert (new / updated data going to central)
- \* Export
- \* Truncate
- \* Import
- \* Delete
- \* Insert (new data from central)
- \* Update

#### 1.36.1.1 Inputs

N/A

#### 1.36.1.2 Selection

Called by EA3.BAT

#### 1.36.1.3 Processing

The system sets the environmental variables for the particular Local Agency reading them from the EA1.INI file and places them in a temporary file to be used for retrieving scripts.

Creates combined log files for the following files:

1. Combines the EA.LOG with the EA\_HIS.LOG by writing it to the end of the EA\_HIS.LOG file.
2. Writes the EA1\_SQL.LOG, EA3\_SQL.LOG, EA3\_RETRIEVE.LOG, and EA4\_SQL.LOG files to the end of the AGCY\_SQL.LOG.
3. Then creates files AGCY\_CTRL.LOG (new file written each day) AGCY\_%.LOG (contains the history of agcy\_ctrl.log updated each day) that log the insert and export information that is dumped from the Agency to Central. The following logs are written to the end of the AGCY\_CTRL.LOG and AGCY\_%.LOG (the wildcard represents the Local Agencies ID):
  - The insert section: EA1\_TAB\_UP.LOG
  - The export section: EXP.LOG
  - The EOD export section: EODEXP.LOG

4. Creates the files CTRL\_AGCY.LOG (new file written each day) and AGCY\_%.LOG (contains the history of agcy\_ctrl.log updated each day) that log the truncate, import, delete, insert, and update information that is loaded from Central to the Local Agency. The following logs are written to the end of the CTRL\_AGCY.LOG and AGCY\_%.LOG (the wildcard represents the Local Agencies ID):

- The truncate section: EA3\_TRNC.LOG
- The import section: IMP.LOG
- The delete section: EA3\_DELTRIG.LOG
- The insert section: EA3\_IN\_TAB.LOG
- The update section: EA3\_TAB\_UP.LOG

The final combined files are EA\_HIS.LOG, AGCY\_CTRL.LOG, CTRL\_AGCY.LOG, and AGCY\_%.LOG. Examples of these log files can be found in Appendix A.

#### **1.36.1.4** Outputs

Combined output of agcy\_ctrl.log, agcy\_%.log, and ctrl\_agcy.log.

## 1.37 RUN\_CASELD.CMD

### *Overview*

This command file begins monthly caseload processing at the State level.

### 1.37.1 Processing

This script runs at the start of every month on the central FTP Server. It calls another script named: INSERT\_MONTHLY\_CASELOADS which populates caseload information for the current month on the State database.

### 1.37.2 INSERT\_MONTHLY\_CASELOADS.SQL

#### *Overview*

This SQL script populates the caseload information for the current month on the State database and updates the caseload information for the past 2 months for the voided checks. It executes caseload population package's NEW\_FCTD procedure to insert new caseloads for the current month and UPD\_FCTD procedure to update the past two month's caseloads.

At the start of the month this script will first populate the F\_CASELOAD\_TYPE\_DETAILS table with the Client's ID, Category Code, Language Code, Ethnic Group Code, Priority ID, Clinic Sequence ID, the current Fiscal Month and Year, the Client's Income, Family Size, Adjunctively Eligible Flag, Migrant Flag, Refugee Flag, State Funding Flag, and WIC/CSF Program. If the Client received a current certification record, the Certification Flag is set to 'Y'. If the Client was issued an unvoided Food Instrument where I\_FOOD\_INSTRUMENTS.FIRST\_DATE\_TO\_USE was in the current month, or if the client has a C\_CLIENTS.CAT\_CATEGORY\_CODE = 'IEN' and the Client's mother is in C\_CLIENTS.CC\_CLIENT\_ID and an I\_FOOD\_INSTRUMENTS.CC\_CLIENT\_ID exists for the Mother's ID with the First Date to Use, then the Participant Flag is set to 'Y'.

On every EOD run, new F\_CASELOAD\_TYPE\_DETAILS records are populated for all Clients who have already been issued checks for the new month, as well as any Exclusively Breastfeeding Infants who still have a mother on WIC. When Food Instruments are created for existing Clients, the F\_CASELOAD\_TYPE\_DETAILS.PARTICIPANT\_FLAG is updated for all applicable records.

It will also generate the F\_CASELOADS.CLIENT\_COUNT by incrementing this value by 1 for each record in the F\_CASELOAD\_TYPE\_DETAILS table for the Category Code, Language Code, Ethnic Group Code, Priority ID, State Funding Flag, Clinic Sequence ID, the current Fiscal Month and Year, and WIC/CSF Program where F\_CASELOAD\_TYPES.DESCRPTION = 'ENROLLEE'. The count is repeated for

clients where the F\_CASELOAD\_TYPE\_DETAILS.MIGRANT\_FLAG = 'Y', updating the F\_CASELOADS table where F\_CASELOAD\_TYPES.DESCRPTION = 'MIGRANT'. The same process is then followed for Clients where the F\_CASELOAD\_TYPE\_DETAILS.PARTICIPANT\_FLAG = 'Y', updating the F\_CASELOADS table where F\_CASELOAD\_TYPES.DESCRPTION = 'PARTICIPANT'.

### 1.37.3 Outputs

F\_CASELOADS  
F\_CASELOAD\_TYPE\_DETAILS