```sql
DROP VIEW ADD_ORS.MATCH_METRICS;

CREATE OR REPLACE FORCE VIEW ADD_ORS.MATCH_METRICS
(METRIC_NAME, METRIC_COUNT)
AS
SELECT 'a) Count Acccounts' AS METRIC_NAME, CAST(COUNT(*) AS varchar2(48)) AS METRIC_COUNT
FROM C_HM_ACCOUNTS
UNION
SELECT 'b) Count Tokens' AS METRIC_NAME, CAST(COUNT(*) AS varchar2(48)) AS METRIC_COUNT FROM
 C_HM_ACCOUNTS_STRP
UNION
SELECT 'c) Count EMI' AS METRIC_NAME, CAST(COUNT(*) AS varchar2(48)) AS METRIC_COUNT FROM
C_HM_ACCOUNTS_EMI
UNION
SELECT 'd) EMI Source ' || EMI.SOURCE_NAME AS METRIC_NAME, CAST(COUNT(EMI.SOURCE_NAME )AS
varchar2(48)) AS METRIC_COUNT  FROM C_HM_ACCOUNTS_EMI EMI GROUP BY EMI.SOURCE_NAME
UNION
SELECT 'e) Count EMO' AS METRIC_NAME, CAST(COUNT(*) AS varchar2(48)) AS METRIC_COUNT FROM
C_HM_ACCOUNTS_EMO
UNION
SELECT 'f) EMI Matched' AS METRIC_NAME, CAST(COUNT(EMI.SOURCE_KEY) AS varchar2(48)) AS
METRIC_COUNT FROM C_HM_ACCOUNTS_EMI EMI WHERE EMI.SOURCE_KEY IN(SELECT EMO.SOURCE_KEY FROM
C_HM_ACCOUNTS_EMO EMO)
UNION
SELECT 'g) EMI not Matched' AS METRIC_NAME, CAST(COUNT(EMI.SOURCE_KEY) AS varchar2(48)) AS
METRIC_COUNT FROM C_HM_ACCOUNTS_EMI EMI WHERE EMI.SOURCE_KEY NOT IN(SELECT EMO.SOURCE_KEY
FROM C_HM_ACCOUNTS_EMO EMO)
UNION
SELECT 'h) EMO Avg. Matched' AS METRIC_NAME, CAST(AVG(COUNT(EMO.SOURCE_KEY)) AS varchar2(8))
 AS METRIC_COUNT FROM C_HM_ACCOUNTS_EMO EMO GROUP BY EMO.SOURCE_KEY
UNION
SELECT 'i) Rule ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME  AS METRIC_NAME, CAST(COUNT(EMO
.SOURCE_NAME || ' : ' || EMO.RULE_NO) AS varchar2(48)) AS METRIC_COUNT FROM
C_HM_ACCOUNTS_EMO EMO GROUP BY 'i) Rule ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME
UNION
SELECT 'i) Tot. Rule No. ' || EMO.RULE_NO AS METRIC_NAME,  CAST(COUNT(EMO.RULE_NO) AS
varchar2(48)) AS METRIC_COUNT FROM C_HM_ACCOUNTS_EMO EMO GROUP BY EMO.RULE_NO
UNION
SELECT 'i) Rule HIT ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME  AS METRIC_NAME, CAST(COUNT
(EMO.SOURCE_NAME || ' : ' || EMO.RULE_NO) AS varchar2(48)) AS METRIC_COUNT FROM
C_HM_ACCOUNTS_EMO EMO
WHERE EMO.RULE_NO IN (5,6,7,8)
GROUP BY 'i) Rule HIT ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME, EMO.AUTOMERGE_IND HAVING
 EMO.AUTOMERGE_IND = 1
UNION
SELECT 'i) Rule NM ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME  AS METRIC_NAME, CAST(COUNT(
EMO.SOURCE_NAME || ' : ' || EMO.RULE_NO) AS varchar2(48)) AS METRIC_COUNT FROM
C_HM_ACCOUNTS_EMO EMO
WHERE EMO.RULE_NO IN (5,6,7,8)
GROUP BY 'i) Rule NM ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME, EMO.AUTOMERGE_IND HAVING
EMO.AUTOMERGE_IND = 2
UNION
SELECT 'j) Population' AS METRIC_NAME, POP.POPULATION_NAME AS METRIC_COUNT FROM
C_REPOS_SSA_POPULATION POP WHERE POP.ENABLED_IND = 1
UNION
SELECT 'k) Level ' || CGX.BO_CLASS_CODE AS METRIC_NAME, CAST(COUNT(CGX.BO_CLASS_CODE)  AS
varchar2(48)) AS METRIC_COUNT FROM C_HM_ACCOUNTS CGX, C_HM_ACCOUNTS_EMO EMO
WHERE
CGX.ROWID_OBJECT = EMO.ROWID_OBJECT_MATCHED
GROUP BY
CGX.BO_CLASS_CODE
UNION
```

```
SELECT '1) Job ' || c_repos_job_metric_type.METRIC_TYPE_DESC AS METRIC_NAME, CAST(
c_repos_job_metric.METRIC_VALUE AS varchar2(8)) AS METRIC_COUNT FROM (c_repos_job_control
INNER JOIN c_repos_job_metric ON c_repos_job_control.ROWID_JOB = c_repos_job_metric.
ROWID_JOB) INNER JOIN c_repos_job_metric_type ON c_repos_job_metric.METRIC_TYPE_CODE =
c_repos_job_metric_type.METRIC_TYPE_CODE
WHERE C_REPOS_JOB_CONTROL.ROWID_JOB = (SELECT JC.ROWID_JOB FROM C_REPOS_JOB_CONTROL JC WHERE
 END_RUN_DATE = (SELECT MAX(END_RUN_DATE) FROM C_REPOS_JOB_CONTROL));

```

```
CREATE OR REPLACE PACKAGE BODY ADD_ORS.ADD_UE AS
--
FUNCTION show_version
    RETURN cmxlb.cmx_med_str
AS
addue_version cmxlb.cmx_med_str := 'Version 1.4 (2014-03-13 16:02:00)';
BEGIN
    RETURN addue_version;
END;
--
FUNCTION GET_PARAMETER(in_name VARCHAR2, in_dft VARCHAR2) RETURN VARCHAR2 AS
     result VARCHAR2(255);
BEGIN
     SELECT VALUE INTO result FROM C_PARAMETERS WHERE NAME = in_name;
     return result;
EXCEPTION
   WHEN OTHERS THEN
          RETURN in_dft;
END;
--
PROCEDURE chk_acct_ids
(
    acct_rowid_object      varchar2,
    acct_sys_name          varchar2,
    acct_sys_id            varchar2,
    acct_cd                varchar2,
    src_nf_ok              boolean,
    out_rowid_object OUT   varchar2,
    out_pkey_src_obj OUT   varchar2,
    out_err_flg      OUT   char,
    out_err_msg      OUT   varchar2
)
AS
sql_stmt varchar2(2000);
t_src_sys_name  varchar2(10);
t_src_sys_id    varchar2(40);
t_rowid_object  varchar2(14);
t_count         integer;
BEGIN
   IF acct_rowid_object is not null THEN
   BEGIN
      -- Check if the rowid_object is valid
     sql_stmt := 'select src_sys_name, src_sys_id from c_hm_accounts where rowid_object =
''' || acct_rowid_object || '''';
      --dbms_output.put_line('sql_stmt = ' || sql_stmt);
      EXECUTE IMMEDIATE sql_stmt into t_src_sys_name, t_src_sys_id;
      --dbms_output.put_line('account found');
      -- Rowid found
      out_rowid_object := acct_rowid_object;
      -- Version 1.4 - Always use account rowid_object to build pkey source object
      out_pkey_src_obj := 'BATCH' || ':' || acct_rowid_object;
      out_err_flg := 'N';
      out_err_msg := '';
      EXCEPTION
        WHEN NO_DATA_FOUND THEN
          --dbms_output.put_line('account not found');
           out_rowid_object := NULL;
           out_pkey_src_obj := NULL;
           out_err_flg := 'Y';
           out_err_msg := 'Invalid account rowid object: ' || acct_rowid_object;
   END;
   ELSIF (acct_sys_name is NOT NULL and acct_sys_id is NOT NULL) THEN
   BEGIN
      -- Check if there is an account with the provided source keys
      sql_stmt := 'select rowid_object from c_hm_accounts where src_sys_name = :1 and src_sys_id = :2';
      dbms_output.put_line('Executing:' || sql_stmt);
      dbms_output.put_line('Using src_sys_name = ' || acct_sys_name || ' src_sys Id: ' || acct_sys_id);
      EXECUTE IMMEDIATE sql_stmt into t_rowid_object using acct_sys_name, acct_sys_id;
      -- Rowid found
      out_rowid_object := t_rowid_object;
      out_pkey_src_obj := acct_sys_name || ':' || acct_sys_id;
      out_err_flg := 'N';
      out_err_msg := '';
      EXCEPTION
        WHEN NO_DATA_FOUND THEN
          IF src_nf_ok THEN
            -- No existing record with this key combo, treat it as a new account
             dbms_output.put_line('No data found, src_nf_ok is true');
             out_rowid_object := NULL;
             out_pkey_src_obj := acct_sys_name || ':' || acct_sys_id;
             out_err_flg := 'N';
             out_err_msg := '';
          ELSE
             dbms_output.put_line('No data found, src_nf_ok is false');
             out_rowid_object := NULL;
             out_pkey_src_obj := NULL;
             out_err_flg := 'Y';
             out_err_msg := 'Account source key not found: ' || acct_sys_id || ':' || acct_rowid_object;
          END IF;
   END;
   ELSIF acct_cd IS NOT NULL THEN
   BEGIN
      -- Check if there are multiple accounts with this account code
      sql_stmt := 'select count(*) from c_hm_accounts where account_cd = :a1';
      EXECUTE IMMEDIATE sql_stmt into t_count using acct_cd;
      --
      -- Bugfix for 3.11. Moved the single row select statment down so it is executed *after* checking that the record count=1
      --
      IF t_count = 0 THEN
          IF src_nf_ok THEN
           -- No existing record with this account code, treat it as a new account
           out_rowid_object := NULL;
           out_pkey_src_obj := 'BATCH' || ':' || acct_cd;
           out_err_flg := 'N';
           out_err_msg := '';
          ELSE
             out_rowid_object := NULL;
             out_pkey_src_obj := NULL;
             out_err_flg := 'Y';
             out_err_msg := 'No account found for account code: ' || acct_cd;
          END IF;
      ELSIF t_count = 1 THEN
        -- Get the id of the account with the provided account cd
        sql_stmt := 'select rowid_object from c_hm_accounts where account_cd = :a1';
        EXECUTE IMMEDIATE sql_stmt into t_rowid_object using acct_cd;
        out_rowid_object := t_rowid_object;
        out_pkey_src_obj := 'BATCH' || ':' || acct_cd;
        out_err_flg := 'N';
        out_err_msg := '';
      ELSE
```

```
          out_rowid_object := NULL;
          out_pkey_src_obj := NULL;
          out_err_flg := 'Y';
          out_err_msg := 'More than one row found for account Code: ' || acct_cd;
        END IF;
        EXCEPTION
          WHEN NO_DATA_FOUND THEN
            IF src_nf_ok THEN
             -- No existing record with this account code, treat it as a new account
             out_rowid_object := NULL;
             out_pkey_src_obj := 'BATCH' || ':' || acct_cd;
             out_err_flg := 'N';
             out_err_msg := '';
            ELSE
               out_rowid_object := NULL;
               out_pkey_src_obj := NULL;
               out_err_flg := 'Y';
               out_err_msg := 'No account found for account code: ' || acct_cd;
            END IF;
     END;
     ELSE
       -- No account identifiers provided
       out_rowid_object := NULL;
       out_pkey_src_obj := NULL;
       out_err_flg := 'Y';
       out_err_msg := 'At least one account identifier should be provided';
     END IF;
  --
  END chk_acct_ids;
  --
  --
  PROCEDURE chk_ptac_ids
  (
      ptac_rowid_object      varchar2,
      ptac_sys_name          varchar2,
      ptac_sys_id            varchar2,
      out_rowid_object OUT   varchar2,
      out_err_flg      OUT   char,
      out_err_msg      OUT   varchar2
  )
  AS
  sql_stmt varchar2(2000);
  t_src_sys_name  varchar2(10);
  t_src_sys_id    varchar2(40);
  t_rowid_object  varchar2(14);
  t_count         integer;
  BEGIN
     IF ptac_rowid_object is not null THEN
     BEGIN
        -- Check if the rowid_object is valid
        sql_stmt := 'select rowid_object from c_bo_party_acct where rowid_object = :rid';
        EXECUTE IMMEDIATE sql_stmt into t_rowid_object using ptac_rowid_object;
        -- Rowid found
        out_rowid_object := ptac_rowid_object;
        out_err_flg := 'N';
        out_err_msg := '';
        EXCEPTION
          WHEN NO_DATA_FOUND THEN
             out_rowid_object := NULL;
             out_err_flg := 'Y';
             out_err_msg := 'Invalid primary customer rowid object: ' || ptac_rowid_object;
     END;
```

```
   ELSIF (ptac_sys_name is NOT NULL and ptac_sys_id is NOT NULL) THEN
   BEGIN
      -- Check if there is an account with the provided source keys
      sql_stmt := 'select rowid_object from c_bo_party_acct_xref where rowid_system = ''' || ptac_sys_name ||
                           ''' and pkey_src_object = ''' || ptac_sys_id || '''';
      --dbms_output.put_line('Executing:' || sql_stmt);
      --dbms_output.put_line('Using rowid_system = ' || ptac_sys_name || ' Sys Id: ' || ptac_sys_id);
      --EXECUTE IMMEDIATE sql_stmt into t_rowid_object using ptac_sys_name, ptac_sys_id;
      EXECUTE IMMEDIATE sql_stmt into t_rowid_object;
      -- Rowid found
      out_rowid_object := t_rowid_object;
      out_err_flg := 'N';
      out_err_msg := '';
      EXCEPTION
        WHEN NO_DATA_FOUND THEN
           -- No existing PTAC record with this key combo, error
           out_rowid_object := NULL;
           out_err_flg := 'Y';
           out_err_msg := 'No customer record found for provided source keys: ' || ptac_sys_name || ':' || ptac_sys_id;
   END;
   ELSE
     -- No PTAC identifiers provided
     out_rowid_object := NULL;
     out_err_flg := 'N';
     out_err_msg := '';
   END IF;
END chk_ptac_ids;


PROCEDURE post_landing
(
    in_rowid_job            cmxlb.cmx_rowid,
    in_ldg_table_name       cmxlb.cmx_table_name,
    in_stg_table_name       cmxlb.cmx_table_name,
    in_prl_table_name       cmxlb.cmx_table_name,
    out_error_msg      OUT  cmxlb.cmx_message,
    out_return_code    OUT  int
)
AS
t_acc_rowid_object varchar2(14);
t_acc_rowid_object2 varchar2(14);
t_ptac_rowid_object varchar2(14);
t_pkey_src_obj varchar2(100);
t_pkey_src_obj2 varchar2(100);
t_err_flg char(1);
t_err_msg varchar2(2000);
--
TYPE ref_cursor is REF CURSOR;
m_cursor ref_cursor;
--
cursor c_accts is
select acc_rowid_object, account_cd, src_sys_name, src_sys_id, ptac_rowid_object, ptac_src_name, ptac_src_key
from   c_ldg_hm_accounts
for update;
--
cursor c_accts_rel is
select acc_rowid_object_1, account_cd_1, src_sys_name_1, src_sys_id_1,
       acc_rowid_object_2, account_cd_2, src_sys_name_2, src_sys_id_2
```

```
from   c_ldg_accounts_rel
for update;
--
cursor c_accts_ptac_rel is
select acc_rowid_object, account_cd, src_sys_name, src_sys_id, ptac_rowid_object,
ptac_src_name, ptac_src_key
from   C_LDG_HM_ACCT_PTY_AC_REL
for update;
--
--
BEGIN
    cmxlog.debug ('Inside ADD_UE.post_landing');
    IF in_ldg_table_name = 'C_LDG_HM_ACCOUNTS' AND in_stg_table_name =
'C_STG_HM_ACCTS_BATCH' THEN
    BEGIN
      cmxlog.debug ('ADDUE: Landing table name is ' || in_ldg_table_name || ' Staging table
name is ' || in_stg_table_name);
      FOR r_accts in c_accts LOOP
        --Check the account identifiers
        chk_acct_ids (r_accts.acc_rowid_object, r_accts.src_sys_name, r_accts.src_sys_id,
r_accts.account_cd, TRUE, t_acc_rowid_object, t_pkey_src_obj, t_err_flg, t_err_msg);
        IF t_err_flg = 'N' THEN
            chk_ptac_ids (r_accts.ptac_rowid_object, r_accts.ptac_src_name, r_accts.
ptac_src_key, t_ptac_rowid_object, t_err_flg, t_err_msg);
        END IF;
        UPDATE C_LDG_HM_ACCOUNTS
        SET  cmp_acc_rowid_object = t_acc_rowid_object,
             cmp_pkey_src_obj = t_pkey_src_obj,
             cmp_ptac_rowid_object = t_ptac_rowid_object,
             error_flag = t_err_flg,
             error_msg  = t_err_msg
        WHERE CURRENT OF c_accts;
      END LOOP;
      COMMIT;
    END;
    ELSIF in_ldg_table_name = 'C_LDG_ACCOUNTS_REL' AND in_stg_table_name =
'C_STG_HM_ACCTS_REL_BATCH' THEN
    BEGIN
      cmxlog.debug ('ADDUE: Landing table name is ' || in_ldg_table_name || ' Staging table
name is ' || in_stg_table_name);
      FOR r_accts_rel in c_accts_rel LOOP
        --Check the account identifiers
        chk_acct_ids (r_accts_rel.acc_rowid_object_1, r_accts_rel.src_sys_name_1,
r_accts_rel.src_sys_id_1, r_accts_rel.account_cd_1, FALSE, t_acc_rowid_object,
t_pkey_src_obj, t_err_flg, t_err_msg);
        IF t_err_flg = 'N' THEN
            chk_acct_ids (r_accts_rel.acc_rowid_object_2, r_accts_rel.src_sys_name_2,
r_accts_rel.src_sys_id_2, r_accts_rel.account_cd_2, FALSE, t_acc_rowid_object2,
t_pkey_src_obj2, t_err_flg, t_err_msg);
        END IF;
        UPDATE C_LDG_ACCOUNTS_REL
        SET  cmp_acc_rowid_object_1 = t_acc_rowid_object,
             cmp_pkey_src_obj = t_pkey_src_obj || ':' || t_pkey_src_obj2,
             cmp_acc_rowid_object_2 = t_acc_rowid_object2,
             error_flag = t_err_flg,
             error_msg  = t_err_msg
        WHERE CURRENT OF c_accts_rel;
      END LOOP;
      COMMIT;
    END;
    ELSIF in_ldg_table_name = 'C_LDG_HM_ACCT_PTY_AC_REL' AND in_stg_table_name =
'C_STG_HM_ACCT_PTAC_BATCH' THEN
```

```
    BEGIN
      cmxlog.debug ('ADDUE: Landing table name is ' || in_ldg_table_name || ' Staging table name is ' || in_stg_table_name);
      FOR r_accts_ptac_rel in c_accts_ptac_rel LOOP
        --Check the account identifiers
        chk_acct_ids (r_accts_ptac_rel.acc_rowid_object, r_accts_ptac_rel.src_sys_name, r_accts_ptac_rel.src_sys_id, r_accts_ptac_rel.account_cd, FALSE, t_acc_rowid_object, t_pkey_src_obj, t_err_flg, t_err_msg);
        --
        IF t_err_flg = 'N' THEN
            chk_ptac_ids (r_accts_ptac_rel.ptac_rowid_object, r_accts_ptac_rel.ptac_src_name, r_accts_ptac_rel.ptac_src_key, t_ptac_rowid_object, t_err_flg, t_err_msg);
        END IF;
        --
        --
        IF r_accts_ptac_rel.ptac_src_name is NOT NULL and r_accts_ptac_rel.ptac_src_key IS NOT NULL THEN
            t_pkey_src_obj2 := t_pkey_src_obj || ':' || r_accts_ptac_rel.ptac_src_name || ':' || r_accts_ptac_rel.ptac_src_key;
        ELSE
            t_pkey_src_obj2 := t_pkey_src_obj || ':' || t_ptac_rowid_object;
        END IF;
        --
        --
        UPDATE C_LDG_HM_ACCT_PTY_AC_REL
        SET  cmp_acc_rowid_object = t_acc_rowid_object,
             cmp_pkey_src_obj = t_pkey_src_obj2,
             cmp_ptac_rowid_object = t_ptac_rowid_object,
             error_flag = t_err_flg,
             error_msg  = t_err_msg
        WHERE CURRENT OF c_accts_ptac_rel;
      END LOOP;
      COMMIT;
    END;
    ELSE
      -- Do nothing
      CMXlog.debug ('ADDUE Post Landing - no action taken');
    END IF;
END;
PROCEDURE post_stage_concat
(
    in_party_acct_id varchar2,
    out_txn_div_display    OUT  varchar2
)
AS
--
BEGIN
    with
    stgbo as
    (
    select party_acct_id, TXN_DIV_CD from C_BO_PTAC_TXN_DIV WHERE party_acct_id = in_party_acct_id
    union
    select party_acct_id, TXN_DIV_CD from C_STG_PTAC_TXN_DIV WHERE PARTY_ACCT_ID = in_party_acct_id
    )
    select distinct TRIM(A||' '||B||' '||C||' '||D) into out_txn_div_display from (select party_acct_id, TXN_DIV_CD from stgbo ) pivot( max(TXN_DIV_CD) for TXN_DIV_CD in ( 'ADC' A, 'ADD' B, 'AMD' C,  'APOC' D)) WHERE PARTY_ACCT_ID = in_party_acct_id;
END post_stage_concat;
PROCEDURE post_stage
(
    in_rowid_job            cmxlb.cmx_rowid,
    in_ldg_table_name       cmxlb.cmx_table_name,
    in_stg_table_name       cmxlb.cmx_table_name,
    out_error_msg      OUT  cmxlb.cmx_message,
    out_return_code    OUT  int
)
AS
sql_stmt varchar2(2000);
t_party_acct_id varchar2(14);
t_txn_div_cd varchar2(20);
t_txn_div_display varchar2(50);
commit_count NUMBER := 0;
commit_inc NUMBER := 1000;
--
CURSOR C_PTAC_TXN IS
SELECT PARTY_ACCT_ID, TXN_DIV_CD, TXN_DIV_DISPLAY
FROM   C_STG_PTAC_TXN_DIV;
--
BEGIN
--
    commit_inc := to_number(GET_PARAMETER('post_stage_commit', commit_inc));
    IF in_ldg_table_name = 'C_LDG_PTAC_TXN_DIV' AND in_stg_table_name = 'C_STG_PTAC_TXN_DIV' THEN
    --    20130225 SCK Update the stage txn_div_display col with a denormalized string derived from an aggregate of both staging and base object
        cmxlog.debug ('ADDUE: Landing table name is ' || in_ldg_table_name || ' Staging table name is ' || in_stg_table_name);
        BEGIN
              FOR R_PTAC_TXN in C_PTAC_TXN LOOP
                    post_stage_concat(R_PTAC_TXN.PARTY_ACCT_ID, t_txn_div_display);
                    UPDATE C_STG_PTAC_TXN_DIV
                    SET txn_div_display = t_txn_div_display, create_date = sysdate WHERE TXN_DIV_CD = R_PTAC_TXN.TXN_DIV_CD AND
                    PARTY_ACCT_ID = R_PTAC_TXN.PARTY_ACCT_ID;  -- CURRENT OF C_PTAC_TXN;
                    commit_count := commit_count + commit_inc;
                    IF MOD(commit_count, 1000) = 0 THEN
                        cmxlog.debug ('ADDUE: post_stage_concat is: ' || commit_count || ':' || R_PTAC_TXN.PARTY_ACCT_ID || ' : ' || t_txn_div_display);
                        COMMIT;
                    END IF;
              END LOOP;
              COMMIT;
        END;
    ELSE
      CMXlog.debug ('ADDUE Post Stage - no action taken');
    END IF;
END post_stage;
END ADD_UE;
/
```

```
CREATE OR REPLACE PACKAGE ADD_ORS.ADD_BATCHJOBS AS
/******************************************************************************
   NAME:       ADD_BATCHJOBS
   PURPOSE:
   REVISIONS:
   Ver        Date        Author           Description
   ---------  ----------  ---------------  ------------------------------------
   2.0        4/24/2011   SCK     Added procedure to execute batch group
******************************************************************************/
    SUBTYPE cmx_rowid IS CHAR( 14 );
    SUBTYPE cmx_med_str IS VARCHAR2( 2000 );
    SUBTYPE cmx_big_str IS VARCHAR2( 8000 );
    SUBTYPE cmx_table_name IS VARCHAR2( 30 );
    SUBTYPE cmx_column_name IS VARCHAR2( 30 );
    SUBTYPE cmx_data_value IS VARCHAR2( 1000 );
    SUBTYPE cmx_object_name IS VARCHAR2( 128 );
    SUBTYPE cmx_object_desc IS VARCHAR2( 255 );
    SUBTYPE cmx_system_name IS VARCHAR2( 50 );
    SUBTYPE cmx_status_msg IS VARCHAR2( 1024 );
    SUBTYPE cmx_date_string IS VARCHAR2( 100 );
    SUBTYPE cmx_user_name IS VARCHAR2( 50 );
    SUBTYPE cmx_single_char IS VARCHAR2( 1 );
    SUBTYPE cmx_job_id IS CHAR( 14 );
    SUBTYPE cmx_error_code IS INT;
    SUBTYPE cmx_error_message IS VARCHAR2( 4000 );

    -- constants for job status, c_repos_job_status_type
    SUCCESS                        INT := 0;
    WARNING                        INT := 1;
    PROCESSING                     INT := 2;
    FAILED                         INT := 99;
    INCOMPLETE                     INT := 4;
    CLEANSE_JOB_FAILED             EXCEPTION;
    LOAD_JOB_FAILED                EXCEPTION;
    MATCHMERGE_JOB_FAILED          EXCEPTION;
    DDL_ERROR                      EXCEPTION;
    SQL_DDL_ERROR                  EXCEPTION;
    JOB_NOT_VALID                  EXCEPTION;
    NO_INDEX_START_TABLE_NAME       EXCEPTION;
    BAD_PARAMETER                  EXCEPTION;
    GLOBAL_INITIALIZATION_FAILED   EXCEPTION;
    READ_CONFIG_FILE_FAILED        EXCEPTION;
    GENERIC_ERROR                  EXCEPTION;

    out_error_code                 cmx_error_code;
    out_error_message              cmx_error_message;

--  Global variables
    job_print_out                  BOOLEAN := true;    -- print messages to console/log file?
    job_print_debug                BOOLEAN := true;    -- print messages to the siperian
debug log

--  Procedure to print to console or siperian debug log
   PROCEDURE x_print (in_msg VARCHAR2);


--- Generic procedure to initialize all global variables --
   PROCEDURE global_initialize(
        in_host_name                                VARCHAR2,
        in_tns_name                                 VARCHAR2,
        in_db_user                                  VARCHAR2,
        out_error_code                    OUT       cmx_error_code,
        out_error_message               OUT       cmx_error_message );

    -- Control Procedure to execute Stage, Load, Match and Merge jobs --
    PROCEDURE exec_mrm_process(
        in_base_table_name  IN  VARCHAR2,
        in_stg_table_name   IN  VARCHAR2,
        in_job_type         IN  VARCHAR2,
        in_host_name        IN  VARCHAR2,
        in_tns_name         IN  VARCHAR2,
        in_db_user          IN  VARCHAR2,
        out_error_code      OUT cmx_error_code,
        out_error_message   OUT cmx_error_message );

--- Generic procedure to run Stage jobs ---
    PROCEDURE run_cleanse(
        in_table_name                             VARCHAR2,
        out_error_code                  OUT       cmx_error_code,
        out_error_message               OUT       cmx_error_message );

--- Generic procedure to run Load jobs ---
    PROCEDURE run_load(
        in_table_name                             VARCHAR2,
        in_force_update_ind                       INT,
        out_error_code                  OUT       cmx_error_code,
        out_error_message               OUT       cmx_error_message );

--- Generic procedure to run Match and Merge jobs ---
    PROCEDURE run_matchmerge(
        in_table_name                             VARCHAR2,
        in_match_set                              VARCHAR2,
        out_error_code                  OUT       cmx_error_code,
        out_error_message               OUT       cmx_error_message );

    PROCEDURE process_load(
        in_table_name                             VARCHAR2,
        out_error_code                  OUT       cmx_error_code,
        out_error_message               OUT       cmx_error_message );

    PROCEDURE process_stage(
        in_table_name                             VARCHAR2,
        out_error_code                  OUT       cmx_error_code,
        out_error_message               OUT       cmx_error_message );

    PROCEDURE process_matchmerge(
        in_table_name                             VARCHAR2,
        out_error_code                  OUT       cmx_error_code,
        out_error_message               OUT       cmx_error_message );

    PROCEDURE execute_batchgroup(
        in_mrm_server                             VARCHAR2,
        in_batch_group                            VARCHAR2,
        in_autosys_id                             VARCHAR2,
        in_autosys_password                       VARCHAR2,
        out_error_code                  OUT       cmx_error_code,
        out_error_message               OUT       cmx_error_message );

END;
/
```

```
CREATE OR REPLACE PACKAGE BODY ADD_ORS.ADD_TASK_ASSIGN AS
addtask_pkb_version cmxlb.cmx_med_str := 'Version 2.0 (2013-03-13 16:32:08)';
addtask_dsi_name cmxlb.cmx_med_str := 'ADDApp';                                    -- Name of our app
addtask_sa_name  cmxlb.cmx_med_str := 'TransactionalCustomer';                     -- Name of the SA that will be attached to the task
addtask_tbl_name  cmxlb.cmx_med_str := 'C_BO_PARTY_ACCT';                         -- Name of the base object table that will be attached to the task data
addtask_tsk_type  cmxlb.cmx_med_str := 'AssignHier';                                   -- Task type to use when creating task
addtask_x_email   cmxlb.cmx_med_str := 'ADD_Data_Stewards_Support@abbott.com';       -- USE the C_PARAMETER Table  Email address to send the exception report to
addtask_x_sender  cmxlb.cmx_med_str := 'NewCustomerAssignment@abbott.com';
                          -- Email address of the sender
l_mesg varchar2(200);
--
--Version History:
--Version 1.0 (2012-26-06 15:52:00) initial version;
--Version 2.0 (2013-03-13 16:32:08) SCK    Generate tasks and assign to users based on Area and ISO Country Code
FUNCTION show_version
    RETURN cmxlb.cmx_med_str
AS
BEGIN
    RETURN addtask_pkb_version;
END;
--
--
PROCEDURE get_zip
(
    in_party_acct_id varchar2,
    out_zip_postal_cd    OUT  varchar2
)
AS
--
BEGIN

SELECT distinct NVL(AD.ZIP_POSTAL_CD, NULL) into out_zip_postal_cd FROM C_BO_PTAC_ADDR x, C_BO_ADDRESS ad
where
AD.ROWID_OBJECT = X.ADDR_ID and
X.PARTY_ACCT_ID = in_party_acct_id;
IF out_zip_postal_cd IS NOT NULL THEN
    out_zip_postal_cd := ':' || out_zip_postal_cd;
END IF;

END get_zip;
PROCEDURE existing_add_tam
(
    in_party_acct_id varchar2,
    existing_relationships    OUT  NUMBER
)
AS
BEGIN
SELECT COUNT(PA.ROWID_OBJECT) INTO existing_relationships FROM C_BO_PARTY_ACCT PA, C_HM_ACCOUNTS HM , C_HM_ACCTS_PTY_ACCT_REL HPR
WHERE PA.ROWID_OBJECT = HPR.ROWID_BO_PARTY_ACCOUNT AND
HPR.ROWID_HM_ACCOUNTS = HM.ROWID_OBJECT AND
HPR.REL_TYPE_CODE = 'CG1 is parent of Customer' AND
HM.ADD_TAM = 'Y' AND
PA.ROWID_OBJECT = in_party_acct_id;

```

```
END existing_add_tam;
FUNCTION GET_PARAMETER(in_name VARCHAR2, in_dft VARCHAR2) RETURN VARCHAR2 AS
     result VARCHAR2(255);
BEGIN
     SELECT VALUE INTO result FROM C_PARAMETERS WHERE NAME = in_name;
     return result;
EXCEPTION
   WHEN OTHERS THEN
          RETURN in_dft;
END;

FUNCTION send_mail (in_sender varchar2, in_recepients varchar2, in_subject varchar2,
in_message varchar2)
    RETURN BOOLEAN
AS
     db_name           VARCHAR2(100) := '';
BEGIN
     select user || ' on ' || sys_context('USERENV', 'DB_NAME') into db_name from dual;

    UTL_MAIL.SEND (
       sender => in_sender,
       recipients => in_recepients,
       subject => in_subject || ' ' || db_name,
       message => in_message);
       return TRUE;
EXCEPTION
   WHEN OTHERS THEN
            dbms_output.put_line('Error sending email: ' || sqlcode || '-' || sqlerrm);
            cmxut.debug_print (l_mesg);
            return FALSE;
END;
--
--
PROCEDURE exception_report (in_email varchar2)
AS
cursor c_ptac is
  select iso_alpha3_cd, count(*) cnt
  from   c_bo_ptac_txn_div dv, c_bo_party_acct pa
  where
  PA.ROWID_OBJECT = DV.PARTY_ACCT_ID and
        dv.cust_match_status = 'E' and pa.bo_class_code='Transactional Customer'
   group by iso_alpha3_cd
   order by 1;
--
l_count integer := 0;
l_body  varchar2(6000);
l_subject varchar2(100) := 'Exception Report - unassigned countries for ' || to_char(sysdate
, 'MM/DD/YYYY');
l_recepients varchar2(255);
BEGIN
    l_body                := 'Country   Unassigned New' || chr(10);
    l_body:= l_body ||  '  Code        Customers     ' || chr(10);
    l_body := l_body ||  '--------    --------------------' || chr(10);
    --
    FOR r_ptac in c_ptac LOOP
        l_body := l_body ||rpad(' ', 3) || r_ptac.iso_alpha3_cd || rpad(' ', 15) || r_ptac.
cnt || chr(10);
        l_count := l_count + r_ptac.cnt;
    END LOOP;
    IF l_count = 0 THEN
        -- No exceptions
        dbms_output.put_line('No exceptions were found');
    ELSE
        l_body := chr(10) || l_body || chr(10) || 'Total unassigned new customers: ' || l_count;
        dbms_output.put_line('Subject: ' || l_subject);
        dbms_output.put_line(l_body);
        l_recepients := GET_PARAMETER('TASK Exception Email', addtask_x_email);
        IF send_mail(addtask_x_sender, l_recepients, l_subject, l_body) THEN
           dbms_output.put_line('Email sent');
        ELSE
           dbms_output.put_line('Error sending email');
        END IF;
    END IF;
END;

FUNCTION get_user_for_country(in_cntry_cd varchar2)
    RETURN cmxlb.cmx_rowid
AS
l_rowid_cntry cmxlb.cmx_rowid;
l_rowid_user cmxlb.cmx_rowid;
l_user_name varchar2(50);
l_global_loc_cd C_LU_ISO_CNTRY.GLOBAL_LOC_CD%TYPE;
l_user_found BOOLEAN;
l_area_found BOOLEAN;
BEGIN
    -- Get the rowid and area for the country
    l_mesg := 'Getting rowid and area for country: ' || in_cntry_cd;
    dbms_output.put_line(l_mesg);
    BEGIN
         SELECT ROWID_OBJECT, GLOBAL_LOC_CD
         INTO     l_rowid_cntry, l_global_loc_cd
         FROM    C_LU_ISO_CNTRY
         WHERE
                iso_alpha3_cd=in_cntry_cd;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
              -- Invalid country!
              l_mesg := 'Country code not found! ' || in_cntry_cd;
              return NULL;
    END;
    -- Get the user associated with the country that has the earliest create date
    l_mesg := 'Looking for assignment for country: ' || in_cntry_cd || ' rowid country: ' || l_rowid_cntry;
    dbms_output.put_line(l_mesg);
    l_user_found := FALSE;
    BEGIN
        SELECT USER_NAME
        INTO     l_user_name
        FROM    (
                        SELECT A.USER_NAME FROM
                        C_BO_DSR_USER A, C_BO_DSR_USER_ISO_CNTRY B
                        WHERE
                               B.ROWID_ISO_CNTRY=l_rowid_cntry AND
                               B.ROWID_DSR_USER=A.ROWID_OBJECT AND
                               NVL(B.DELETED_IND,0)=0 AND
                               NVL(B.HUB_STATE_IND,1)=1 AND
                               NVL(A.HUB_STATE_IND,1)=1 AND
                               A.USER_ENABLED_IND=1
                         ORDER BY B.CREATE_DATE)
         WHERE ROWNUM < 2;
         l_user_found := TRUE;
         l_mesg := 'Country level assignment found, rowid_user: ' || l_rowid_user;
         dbms_output.put_line(l_mesg);
    EXCEPTION
         WHEN NO_DATA_FOUND THEN
               l_user_found := FALSE;
    END;
    --
    IF NOT l_user_found THEN
        -- No active user was found for the country - get the user that has the area responsibility
           l_mesg := 'Did not find assignment for country: ' || in_cntry_cd || ' checking area level ' || l_global_loc_cd;
           dbms_output.put_line(l_mesg);
           l_area_found := FALSE;
        --
       BEGIN
        SELECT USER_NAME
        INTO      l_user_name
        FROM     (
                        SELECT A.USER_NAME FROM
                        C_BO_DSR_USER A, C_BO_DSR_USER_ISO_CNTRY B
                        WHERE
                               B.ROWID_ISO_CNTRY IS NULL AND
                               B.GLOBAL_LOC_CD=l_global_loc_cd AND
                               B.ROWID_DSR_USER=A.ROWID_OBJECT AND
                               NVL(B.DELETED_IND,0)=0 AND
                               NVL(B.HUB_STATE_IND,1)=1 AND
                               NVL(A.HUB_STATE_IND,1)=1 AND
                               A.USER_ENABLED_IND=1
                         ORDER BY B.CREATE_DATE)
         WHERE ROWNUM < 2;
            l_area_found := TRUE;
            l_mesg := 'Area level assignment found, rowid_user: ' || l_rowid_user;
           dbms_output.put_line(l_mesg);
         EXCEPTION
            WHEN NO_DATA_FOUND THEN
                 l_area_found := FALSE;
         END;
       --
       --
    END IF;
    IF NOT (l_area_found or l_user_found) THEN
      BEGIN
           l_mesg := 'Did not find either country or area level assignment for country: ' || in_cntry_cd;
           dbms_output.put_line(l_mesg);
           cmxut.debug_print (l_mesg);
           return NULL;
      END;
    END IF;
   --
    BEGIN
         l_mesg := 'Looking up user: ' || l_user_name;
         dbms_output.put_line(l_mesg);
         cmxut.debug_print (l_mesg);
        SELECT ROWID_USER
        INTO      l_rowid_user
        FROM    C_REPOS_USER
        WHERE
               user_name = l_user_name;
         EXCEPTION
            WHEN NO_DATA_FOUND THEN
                l_mesg := 'User not found in C_REPOS_USER! : ' || l_user_name;
               dbms_output.put_line(l_mesg);
```

```
                cmxut.debug_print (l_mesg);
                return NULL;
      END;
     RETURN l_rowid_user;
END;
--
PROCEDURE add_assign_tasks(
    in_max_tasks IN INT,
    out_unassigned_task_count    OUT   INT,
    out_assigned_task_count       OUT   INT,
    out_error_msg                     OUT   cmxlb.cmx_message,
    out_return_code                   OUT   INT)
AS
--    20130313    SCK    Generate tasks and assign to users based on Area and ISO Country Code
lock_applied BOOLEAN;
l_rowid_sa cmxlb.cmx_rowid;
l_rowid_tbl cmxlb.cmx_rowid;
l_rowid_user cmxlb.cmx_rowid;
l_task_title cmxlb.cmx_big_str;
l_rowid_task cmxlb.cmx_rowid;
l_match_status char(1);
zip_postal_cd varchar2(36);
dueDateDay NUMBER;
existing_relationships NUMBER;
--
        CURSOR c_txn
        IS
            SELECT TX.TXN_DIV_CD,
                TX.CUST_MATCH_STATUS,
                TX.ROWID_OBJECT,
                TX.PARTY_ACCT_ID,
                TX.HUB_STATE_IND,
                PA.CUST_NBR_PK,
                PA.CUST_NM,
                PA.ISO_ALPHA3_CD
            FROM C_BO_PTAC_TXN_DIV TX, C_BO_PARTY_ACCT PA
            WHERE TX.PARTY_ACCT_ID = PA.ROWID_OBJECT AND
                TX.CUST_MATCH_STATUS IN ('N',  'E') AND
                TX.TXN_DIV_CD = 'ADD' AND
                NVL(TX.HUB_STATE_IND,1) = 1
            FOR UPDATE ;
        --
        BEGIN
                dueDateDay := TO_NUMBER (GET_PARAMETER ('DUE DATE DAYS', '7'));
                l_mesg := 'due days: ' || dueDateDay;
                BEGIN
                    SELECT rowid_subject_area
                    INTO l_rowid_sa
                    FROM c_repos_subject_area
                    WHERE dsi_name = addtask_dsi_name AND name = addtask_sa_name;
                END;
                 l_mesg := l_mesg || ' sa: ' || TRIM(l_rowid_sa);
                BEGIN
                SELECT rowid_table
                    INTO l_rowid_tbl
                    FROM c_repos_table
                    WHERE table_name = addtask_tbl_name;
                END;
                 l_mesg := l_mesg || ' table: ' || TRIM(l_rowid_tbl);
               dbms_output.put_line(l_mesg);
                cmxut.debug_print (l_mesg);
                FOR r_txn in c_txn LOOP
                    existing_add_tam(r_txn.PARTY_ACCT_ID, existing_relationships);
                     l_mesg := 'tam_relates: ' || existing_relationships;
                    IF existing_relationships = 0 THEN
                         l_rowid_user := get_user_for_country(r_txn.iso_alpha3_cd);
                          l_mesg := l_mesg || ' user_iso: ' || TRIM(l_rowid_user) || '_' || r_txn.iso_alpha3_cd;
                         IF l_rowid_user IS NULL THEN
                            l_match_status := 'E';
                         ELSE
                            get_zip(r_txn.PARTY_ACCT_ID, zip_postal_cd);
                            BEGIN
                                 l_task_title := r_txn.cust_nbr_pk || ':' || r_txn.cust_nm || ':' || r_txn.iso_alpha3_cd || zip_postal_cd;
                                 l_mesg := 'Creating task with title: ' || l_task_title || ' for rowid_user: ' || l_rowid_user;
                                dbms_output.put_line(l_mesg);
                                cmxut.debug_print (l_mesg);
                                 l_rowid_task := c_repos_task_seq.nextval;
                                 --
                                 l_mesg := 'Inserting into c_repos_task_assignment';
                                dbms_output.put_line(l_mesg);
                                 cmxut.debug_print (l_mesg);
                                 INSERT INTO c_repos_task_assignment (rowid_task, rowid_user, rowid_subject_area, task_type, task_comment, due_date, creator, updated_by, status, title, rowid_workflow_process)
                                 VALUES (l_rowid_task, l_rowid_user, l_rowid_sa, addtask_tsk_type, 'A new customer has been introduced into MDM and may require Hierarchy maintenance (Row ID: ' || r_txn.PARTY_ACCT_ID || ')', trunc(sysdate+dueDateDay), user, user, 'OPEN', l_task_title, l_rowid_task);
                            END;
                           -- Create task data
                           BEGIN
                                 l_mesg := 'Inserting into c_repos_task_data';
                                dbms_output.put_line(l_mesg);
                                 cmxut.debug_print (l_mesg);
                                INSERT INTO C_REPOS_TASK_DATA (ROWID_OBJECT, ROWID_TABLE, ROWID_TASK, ROWID_TASK_DATA)
                                VALUES (r_txn.PARTY_ACCT_ID, l_rowid_tbl, l_rowid_task, c_repos_task_data_seq.nextval);
                           END;
                            l_match_status := 'A';
                         END IF;

                    ELSE
                        l_match_status := 'A';
                    END IF;
                     l_mesg := l_mesg || ' c_m_status: ' || l_match_status;
                    dbms_output.put_line(l_mesg);
                    cmxut.debug_print (l_mesg);
                    --
                     UPDATE C_BO_PTAC_TXN_DIV
                    SET CUST_MATCH_STATUS = l_match_status
                    WHERE ROWID_OBJECT = r_txn.rowid_object;
                    --
                    UPDATE C_BO_PTAC_TXN_DIV_XREF
                    SET CUST_MATCH_STATUS = l_match_status
                    WHERE ROWID_OBJECT = r_txn.rowid_object;
                END LOOP;
                COMMIT;
      exception_report(addtask_x_email);
EXCEPTION
```

```
        WHEN OTHERS THEN
               dbms_output.put_line(l_mesg);
                cmxut.debug_print (l_mesg);
               dbms_output.put_line(sqlcode || '-' || sqlerrm);
                cmxut.debug_print (sqlcode || '-' || sqlerrm);
                out_error_msg := 'ERROR ' || l_mesg || '::' || sqlcode || '-' || sqlerrm;
                out_return_code:=-10222;
        END;
END ADD_TASK_ASSIGN;
/
```