

```

1 DROP VIEW ADD_ORO.MATCH_METRICS;
2
3 CREATE OR REPLACE FORCE VIEW ADD_ORO.MATCH_METRICS
4 (METRIC_NAME, METRIC_COUNT)
5 AS
6 SELECT 'a) Count Acccounts' AS METRIC_NAME, CAST(COUNT(*) AS varchar2(48)) AS METRIC_COUNT
7 FROM C_HM_ACCOUNTS
8 UNION
9 SELECT 'b) Count Tokens' AS METRIC_NAME, CAST(COUNT(*) AS varchar2(48)) AS METRIC_COUNT FROM
10 C_HM_ACCOUNTS_STRP
11 UNION
12 SELECT 'c) Count EMI' AS METRIC_NAME, CAST(COUNT(*) AS varchar2(48)) AS METRIC_COUNT FROM
13 C_HM_ACCOUNTS_EMI
14 UNION
15 SELECT 'd) EMI Source ' || EMI.SOURCE_NAME AS METRIC_NAME, CAST(COUNT(EMI.SOURCE_NAME )AS
16 varchar2(48)) AS METRIC_COUNT FROM C_HM_ACCOUNTS_EMI EMI GROUP BY EMI.SOURCE_NAME
17 UNION
18 SELECT 'e) Count EMO' AS METRIC_NAME, CAST(COUNT(*) AS varchar2(48)) AS METRIC_COUNT FROM
19 C_HM_ACCOUNTS_EMO
20 UNION
21 SELECT 'f) EMI Matched' AS METRIC_NAME, CAST(COUNT(EMI.SOURCE_KEY) AS varchar2(48)) AS
22 METRIC_COUNT FROM C_HM_ACCOUNTS_EMI EMI WHERE EMI.SOURCE_KEY IN(SELECT EMO.SOURCE_KEY FROM
23 C_HM_ACCOUNTS_EMO EMO)
24 UNION
25 SELECT 'g) EMI not Matched' AS METRIC_NAME, CAST(COUNT(EMI.SOURCE_KEY) AS varchar2(48)) AS
26 METRIC_COUNT FROM C_HM_ACCOUNTS_EMI EMI WHERE EMI.SOURCE_KEY NOT IN(SELECT EMO.SOURCE_KEY
27 FROM C_HM_ACCOUNTS_EMO EMO)
28 UNION
29 SELECT 'h) EMO Avg. Matched' AS METRIC_NAME, CAST(AVG(COUNT(EMO.SOURCE_KEY)) AS varchar2(8))
30 AS METRIC_COUNT FROM C_HM_ACCOUNTS_EMO EMO GROUP BY EMO.SOURCE_KEY
31 UNION
32 SELECT 'i) Rule ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME AS METRIC_NAME, CAST(COUNT(EMO
33 .SOURCE_NAME || ' : ' || EMO.RULE_NO) AS varchar2(48)) AS METRIC_COUNT FROM
34 C_HM_ACCOUNTS_EMO EMO GROUP BY 'i) Rule ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME
35 UNION
36 SELECT 'i) Tot. Rule No. ' || EMO.RULE_NO AS METRIC_NAME, CAST(COUNT(EMO.RULE_NO) AS
37 varchar2(48)) AS METRIC_COUNT FROM C_HM_ACCOUNTS_EMO EMO GROUP BY EMO.RULE_NO
38 UNION
39 SELECT 'i) Rule HIT ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME AS METRIC_NAME, CAST(COUNT
40 (EMO.SOURCE_NAME || ' : ' || EMO.RULE_NO) AS varchar2(48)) AS METRIC_COUNT FROM
41 C_HM_ACCOUNTS_EMO EMO
42 WHERE EMO.RULE_NO IN (5,6,7,8)
43 GROUP BY 'i) Rule HIT ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME, EMO.AUTOMERGE_IND HAVING
44 EMO.AUTOMERGE_IND = 1
45 UNION
46 SELECT 'i) Rule NM ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME AS METRIC_NAME, CAST(COUNT(
47 EMO.SOURCE_NAME || ' : ' || EMO.RULE_NO) AS varchar2(48)) AS METRIC_COUNT FROM
48 C_HM_ACCOUNTS_EMO EMO
49 WHERE EMO.RULE_NO IN (5,6,7,8)
50 GROUP BY 'i) Rule NM ' || EMO.RULE_NO || ' : ' || EMO.SOURCE_NAME, EMO.AUTOMERGE_IND HAVING
51 EMO.AUTOMERGE_IND = 2
52 UNION
53 SELECT 'j) Population' AS METRIC_NAME, POP.POPULATION_NAME AS METRIC_COUNT FROM
54 C_REPOS_SSA_POPULATION POP WHERE POP.ENABLED_IND = 1
55 UNION
56 SELECT 'k) Level ' || CGX.BO_CLASS_CODE AS METRIC_NAME, CAST(COUNT(CGX.BO_CLASS_CODE) AS
57 varchar2(48)) AS METRIC_COUNT FROM C_HM_ACCOUNTS CGX, C_HM_ACCOUNTS_EMO EMO
58 WHERE
59 CGX.ROWID_OBJECT = EMO.ROWID_OBJECT_MATCHED
60 GROUP BY
61 CGX.BO_CLASS_CODE
62 UNION

```

```
42  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  
43  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));  
44
```

```

1 CREATE OR REPLACE PACKAGE BODY ADD_ORS.ADD_UE AS
2 --
3 FUNCTION show_version
4     RETURN cmxlb.cmx_med_str
5 AS
6 addue_version cmxlb.cmx_med_str := 'Version 1.4 (2014-03-13 16:02:00)';
7 BEGIN
8     RETURN addue_version;
9 END;
10 --
11 FUNCTION GET_PARAMETER(in_name VARCHAR2, in_dft VARCHAR2) RETURN VARCHAR2 AS
12     result VARCHAR2(255);
13 BEGIN
14     SELECT VALUE INTO result FROM C_PARAMETERS WHERE NAME = in_name;
15     return result;
16 EXCEPTION
17     WHEN OTHERS THEN
18         RETURN in_dft;
19 END;
20 --
21 PROCEDURE chk_acct_ids
22 (
23     acct_rowid_object      varchar2,
24     acct_sys_name          varchar2,
25     acct_sys_id            varchar2,
26     acct_cd                varchar2,
27     src_nf_ok              boolean,
28     out_rowid_object OUT   varchar2,
29     out_pkey_src_obj OUT   varchar2,
30     out_err_flg            OUT   char,
31     out_err_msg            OUT   varchar2
32 )
33 AS
34 sql_stmt varchar2(2000);
35 t_src_sys_name varchar2(10);
36 t_src_sys_id   varchar2(40);
37 t_rowid_object varchar2(14);
38 t_count        integer;
39 BEGIN
40     IF acct_rowid_object is not null THEN
41         BEGIN
42             -- Check if the rowid_object is valid
43             sql_stmt := 'select src_sys_name, src_sys_id from c_hm_accounts where rowid_object =
''' || acct_rowid_object || ''';
44             --dbms_output.put_line('sql_stmt = ' || sql_stmt);
45             EXECUTE IMMEDIATE sql_stmt into t_src_sys_name, t_src_sys_id;
46             --dbms_output.put_line('account found');
47             -- Rowid found
48             out_rowid_object := acct_rowid_object;
49             -- Version 1.4 - Always use account rowid_object to build pkey source object
50             out_pkey_src_obj := 'BATCH' || ':' || acct_rowid_object;
51             out_err_flg := 'N';
52             out_err_msg := '';
53             EXCEPTION
54                 WHEN NO_DATA_FOUND THEN
55                     --dbms_output.put_line('account not found');
56                     out_rowid_object := NULL;
57                     out_pkey_src_obj := NULL;
58                     out_err_flg := 'Y';
59                     out_err_msg := 'Invalid account rowid object: ' || acct_rowid_object;
60         END;
61     ELSIF (acct_sys_name is NOT NULL and acct_sys_id is NOT NULL) THEN

```

```

62     BEGIN
63         -- Check if there is an account with the provided source keys
64         sql_stmt := 'select rowid_object from c_hm_accounts where src_sys_name = :1 and
src_sys_id = :2';
65         dbms_output.put_line('Executing:' || sql_stmt);
66         dbms_output.put_line('Using src_sys_name = ' || acct_sys_name || ' src_sys Id: ' ||
acct_sys_id);
67         EXECUTE IMMEDIATE sql_stmt into t_rowid_object using acct_sys_name, acct_sys_id;
68         -- Rowid found
69         out_rowid_object := t_rowid_object;
70         out_pkey_src_obj := acct_sys_name || ':' || acct_sys_id;
71         out_err_flg := 'N';
72         out_err_msg := '';
73     EXCEPTION
74         WHEN NO_DATA_FOUND THEN
75             IF src_nf_ok THEN
76                 -- No existing record with this key combo, treat it as a new account
77                 dbms_output.put_line('No data found, src_nf_ok is true');
78                 out_rowid_object := NULL;
79                 out_pkey_src_obj := acct_sys_name || ':' || acct_sys_id;
80                 out_err_flg := 'N';
81                 out_err_msg := '';
82             ELSE
83                 dbms_output.put_line('No data found, src_nf_ok is false');
84                 out_rowid_object := NULL;
85                 out_pkey_src_obj := NULL;
86                 out_err_flg := 'Y';
87                 out_err_msg := 'Account source key not found: ' || acct_sys_id || ':' ||
acct_rowid_object;
88             END IF;
89     END;
90     ELSIF acct_cd IS NOT NULL THEN
91     BEGIN
92         -- Check if there are multiple accounts with this account code
93         sql_stmt := 'select count(*) from c_hm_accounts where account_cd = :a1';
94         EXECUTE IMMEDIATE sql_stmt into t_count using acct_cd;
95         --
96         -- Bugfix for 3.11. Moved the single row select statment down so it is executed
*after* checking that the record count=1
97         --
98         IF t_count = 0 THEN
99             IF src_nf_ok THEN
100                 -- No existing record with this account code, treat it as a new account
101                 out_rowid_object := NULL;
102                 out_pkey_src_obj := 'BATCH' || ':' || acct_cd;
103                 out_err_flg := 'N';
104                 out_err_msg := '';
105             ELSE
106                 out_rowid_object := NULL;
107                 out_pkey_src_obj := NULL;
108                 out_err_flg := 'Y';
109                 out_err_msg := 'No account found for account code: ' || acct_cd;
110             END IF;
111         ELSIF t_count = 1 THEN
112             -- Get the id of the account with the provided account cd
113             sql_stmt := 'select rowid_object from c_hm_accounts where account_cd = :a1';
114             EXECUTE IMMEDIATE sql_stmt into t_rowid_object using acct_cd;
115             out_rowid_object := t_rowid_object;
116             out_pkey_src_obj := 'BATCH' || ':' || acct_cd;
117             out_err_flg := 'N';
118             out_err_msg := '';
119         ELSE

```

```

120         out_rowid_object := NULL;
121         out_pkey_src_obj := NULL;
122         out_err_flg := 'Y';
123         out_err_msg := 'More than one row found for account Code: ' || acct_cd;
124     END IF;
125     EXCEPTION
126         WHEN NO_DATA_FOUND THEN
127             IF src_nf_ok THEN
128                 -- No existing record with this account code, treat it as a new account
129                 out_rowid_object := NULL;
130                 out_pkey_src_obj := 'BATCH' || ':' || acct_cd;
131                 out_err_flg := 'N';
132                 out_err_msg := '';
133             ELSE
134                 out_rowid_object := NULL;
135                 out_pkey_src_obj := NULL;
136                 out_err_flg := 'Y';
137                 out_err_msg := 'No account found for account code: ' || acct_cd;
138             END IF;
139     END;
140     ELSE
141         -- No account identifiers provided
142         out_rowid_object := NULL;
143         out_pkey_src_obj := NULL;
144         out_err_flg := 'Y';
145         out_err_msg := 'At least one account identifier should be provided';
146     END IF;
147     --
148     END chk_acct_ids;
149     --
150     --
151     PROCEDURE chk_ptac_ids
152     (
153         ptac_rowid_object      varchar2,
154         ptac_sys_name          varchar2,
155         ptac_sys_id            varchar2,
156         out_rowid_object OUT  varchar2,
157         out_err_flg          OUT  char,
158         out_err_msg          OUT  varchar2
159     )
160     AS
161     sql_stmt varchar2(2000);
162     t_src_sys_name  varchar2(10);
163     t_src_sys_id    varchar2(40);
164     t_rowid_object  varchar2(14);
165     t_count         integer;
166     BEGIN
167         IF ptac_rowid_object is not null THEN
168             BEGIN
169                 -- Check if the rowid_object is valid
170                 sql_stmt := 'select rowid_object from c_bo_party_acct where rowid_object = :rid';
171                 EXECUTE IMMEDIATE sql_stmt into t_rowid_object using ptac_rowid_object;
172                 -- Rowid found
173                 out_rowid_object := ptac_rowid_object;
174                 out_err_flg := 'N';
175                 out_err_msg := '';
176             EXCEPTION
177                 WHEN NO_DATA_FOUND THEN
178                     out_rowid_object := NULL;
179                     out_err_flg := 'Y';
180                     out_err_msg := 'Invalid primary customer rowid object: ' || ptac_rowid_object;
181             END;

```

```

182     ELSIF (ptac_sys_name is NOT NULL and ptac_sys_id is NOT NULL) THEN
183     BEGIN
184         -- Check if there is an account with the provided source keys
185         sql_stmt := 'select rowid_object from c_bo_party_acct_xref where rowid_system = ''' ||
ptac_sys_name ||
186                     ''' and pkey_src_object = ''' || ptac_sys_id || ''';
187         --dbms_output.put_line('Executing:' || sql_stmt);
188         --dbms_output.put_line('Using rowid_system = ' || ptac_sys_name || ' Sys Id: ' ||
ptac_sys_id);
189         --EXECUTE IMMEDIATE sql_stmt into t_rowid_object using ptac_sys_name, ptac_sys_id;
190         EXECUTE IMMEDIATE sql_stmt into t_rowid_object;
191         -- Rowid found
192         out_rowid_object := t_rowid_object;
193         out_err_flg := 'N';
194         out_err_msg := '';
195     EXCEPTION
196         WHEN NO_DATA_FOUND THEN
197             -- No existing PTAC record with this key combo, error
198             out_rowid_object := NULL;
199             out_err_flg := 'Y';
200             out_err_msg := 'No customer record found for provided source keys: ' ||
ptac_sys_name || ':' || ptac_sys_id;
201     END;
202     ELSE
203         -- No PTAC identifiers provided
204         out_rowid_object := NULL;
205         out_err_flg := 'N';
206         out_err_msg := '';
207     END IF;
208 END chk_ptac_ids;
209
210
211 PROCEDURE post_landing
212 (
213     in_rowid_job           cmxlb.cmx_rowid,
214     in_ldg_table_name      cmxlb.cmx_table_name,
215     in_stg_table_name      cmxlb.cmx_table_name,
216     in_prl_table_name      cmxlb.cmx_table_name,
217     out_error_msg          OUT cmxlb.cmx_message,
218     out_return_code        OUT int
219 )
220 AS
221 t_acc_rowid_object varchar2(14);
222 t_acc_rowid_object2 varchar2(14);
223 t_ptac_rowid_object varchar2(14);
224 t_pkey_src_obj varchar2(100);
225 t_pkey_src_obj2 varchar2(100);
226 t_err_flg char(1);
227 t_err_msg varchar2(2000);
228 --
229 TYPE ref_cursor is REF CURSOR;
230 m_cursor ref_cursor;
231 --
232 cursor c_accts is
233 select acc_rowid_object, account_cd, src_sys_name, src_sys_id, ptac_rowid_object,
ptac_src_name, ptac_src_key
234 from c_ldg_hm_accounts
235 for update;
236 --
237 cursor c_accts_rel is
238 select acc_rowid_object_1, account_cd_1, src_sys_name_1, src_sys_id_1,
239        acc_rowid_object_2, account_cd_2, src_sys_name_2, src_sys_id_2

```

```

240 from c_ldg_accounts_rel
241 for update;
242 --
243 cursor c_accts_ptac_rel is
244 select acc_rowid_object, account_cd, src_sys_name, src_sys_id, ptac_rowid_object,
ptac_src_name, ptac_src_key
245 from C_LDG_HM_ACCT_PTY_AC_REL
246 for update;
247 --
248 --
249 BEGIN
250     cmxlog.debug ('Inside ADD_UE.post_landing');
251     IF in_ldg_table_name = 'C_LDG_HM_ACCOUNTS' AND in_stg_table_name =
'C_STG_HM_ACCTS_BATCH' THEN
252         BEGIN
253             cmxlog.debug ('ADDUE: Landing table name is ' || in_ldg_table_name || ' Staging table
name is ' || in_stg_table_name);
254             FOR r_accts in c_accts LOOP
255                 --Check the account identifiers
256                 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);
257                 IF t_err_flg = 'N' THEN
258                     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);
259                 END IF;
260                 UPDATE C_LDG_HM_ACCOUNTS
261                 SET     cmp_acc_rowid_object = t_acc_rowid_object,
262                        cmp_pkey_src_obj = t_pkey_src_obj,
263                        cmp_ptac_rowid_object = t_ptac_rowid_object,
264                        error_flag = t_err_flg,
265                        error_msg = t_err_msg
266                 WHERE CURRENT OF c_accts;
267             END LOOP;
268             COMMIT;
269         END;
270     ELSIF in_ldg_table_name = 'C_LDG_ACCOUNTS_REL' AND in_stg_table_name =
'C_STG_HM_ACCTS_REL_BATCH' THEN
271         BEGIN
272             cmxlog.debug ('ADDUE: Landing table name is ' || in_ldg_table_name || ' Staging table
name is ' || in_stg_table_name);
273             FOR r_accts_rel in c_accts_rel LOOP
274                 --Check the account identifiers
275                 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);
276                 IF t_err_flg = 'N' THEN
277                     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);
278                 END IF;
279                 UPDATE C_LDG_ACCOUNTS_REL
280                 SET     cmp_acc_rowid_object_1 = t_acc_rowid_object,
281                        cmp_pkey_src_obj = t_pkey_src_obj || ':' || t_pkey_src_obj2,
282                        cmp_acc_rowid_object_2 = t_acc_rowid_object2,
283                        error_flag = t_err_flg,
284                        error_msg = t_err_msg
285                 WHERE CURRENT OF c_accts_rel;
286             END LOOP;
287             COMMIT;
288         END;
289     ELSIF in_ldg_table_name = 'C_LDG_HM_ACCT_PTY_AC_REL' AND in_stg_table_name =
'C_STG_HM_ACCT_PTAC_BATCH' THEN

```

```

290 BEGIN
291     cmxlog.debug ('ADDUE: Landing table name is ' || in_ldg_table_name || ' Staging table
name is ' || in_stg_table_name);
292     FOR r_accts_ptac_rel in c_accts_ptac_rel LOOP
293         --Check the account identifiers
294         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);
295         --
296         IF t_err_flg = 'N' THEN
297             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);
298         END IF;
299         --
300         --
301         IF r_accts_ptac_rel.ptac_src_name is NOT NULL and r_accts_ptac_rel.ptac_src_key IS
NOT NULL THEN
302             t_pkey_src_obj2 := t_pkey_src_obj || ':' || r_accts_ptac_rel.ptac_src_name ||
':' || r_accts_ptac_rel.ptac_src_key;
303         ELSE
304             t_pkey_src_obj2 := t_pkey_src_obj || ':' || t_ptac_rowid_object;
305         END IF;
306         --
307         --
308         UPDATE C_LDG_HM_ACCT_PTY_AC_REL
309         SET     cmp_acc_rowid_object = t_acc_rowid_object,
310               cmp_pkey_src_obj = t_pkey_src_obj2,
311               cmp_ptac_rowid_object = t_ptac_rowid_object,
312               error_flag = t_err_flg,
313               error_msg = t_err_msg
314         WHERE CURRENT OF c_accts_ptac_rel;
315     END LOOP;
316     COMMIT;
317 END;
318 ELSE
319     -- Do nothing
320     CMXlog.debug ('ADDUE Post Landing - no action taken');
321 END IF;
322 END;
323 PROCEDURE post_stage_concat
324 (
325     in_party_acct_id varchar2,
326     out_txn_div_display OUT varchar2
327 )
328 AS
329 --
330 BEGIN
331     with
332     stgbo as
333     (
334         select party_acct_id, TXN_DIV_CD from C_BO_PTAC_TXN_DIV WHERE party_acct_id =
in_party_acct_id
335     union
336         select party_acct_id, TXN_DIV_CD from C_STG_PTAC_TXN_DIV WHERE PARTY_ACCT_ID =
in_party_acct_id
337     )
338     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;
339 END post_stage_concat;
340 PROCEDURE post_stage
341 (

```



```

342         in_rowid_job                cmxlb.cmx_rowid,
343         in_ldg_table_name            cmxlb.cmx_table_name,
344         in_stg_table_name            cmxlb.cmx_table_name,
345         out_error_msg                OUT cmxlb.cmx_message,
346         out_return_code              OUT int
347     )
348 AS
349     sql_stmt varchar2(2000);
350     t_party_acct_id varchar2(14);
351     t_txn_div_cd varchar2(20);
352     t_txn_div_display varchar2(50);
353     commit_count NUMBER := 0;
354     commit_inc NUMBER := 1000;
355     --
356     CURSOR C_PTAC_TXN IS
357     SELECT PARTY_ACCT_ID, TXN_DIV_CD, TXN_DIV_DISPLAY
358     FROM    C_STG_PTAC_TXN_DIV;
359     --
360     BEGIN
361     --
362         commit_inc := to_number(GET_PARAMETER('post_stage_commit', commit_inc));
363         IF in_ldg_table_name = 'C_LDG_PTAC_TXN_DIV' AND in_stg_table_name = 'C_STG_PTAC_TXN_DIV'
364         THEN
365             --      20130225 SCK Update the stage txn_div_display col with a denormalized string
366             --      derived from an aggregate of both staging and base object
367             cmxlog.debug ('ADDUE: Landing table name is ' || in_ldg_table_name || ' Staging
368             table name is ' || in_stg_table_name);
369             BEGIN
370                 FOR R_PTAC_TXN in C_PTAC_TXN LOOP
371                     post_stage_concat(R_PTAC_TXN.PARTY_ACCT_ID, t_txn_div_display);
372                     UPDATE C_STG_PTAC_TXN_DIV
373                     SET txn_div_display = t_txn_div_display, create_date = sysdate WHERE
374                     TXN_DIV_CD = R_PTAC_TXN.TXN_DIV_CD AND
375                     PARTY_ACCT_ID = R_PTAC_TXN.PARTY_ACCT_ID; -- CURRENT OF C_PTAC_TXN;
376                     commit_count := commit_count + commit_inc;
377                     IF MOD(commit_count, 1000) = 0 THEN
378                         cmxlog.debug ('ADDUE: post_stage_concat is: ' || commit_count || ':'
379                         || R_PTAC_TXN.PARTY_ACCT_ID || ' : ' || t_txn_div_display);
380                         COMMIT;
381                     END IF;
382                 END LOOP;
383                 COMMIT;
384             END;
385         ELSE
386             CMXlog.debug ('ADDUE Post Stage - no action taken');
387         END IF;
388     END post_stage;
389 END ADD_UE;
390 /

```

```

1 CREATE OR REPLACE PACKAGE ADD_ORs.ADD_BATCHJOBS AS
2 /*****
3     NAME:          ADD_BATCHJOBS
4     PURPOSE:
5     REVISIONS:
6     Ver           Date           Author           Description
7     -----
8     2.0           4/24/2011      SCK           Added procedure to execute batch group
9 *****/
10     SUBTYPE cmx_rowid IS CHAR( 14 );
11     SUBTYPE cmx_med_str IS VARCHAR2( 2000 );
12     SUBTYPE cmx_big_str IS VARCHAR2( 8000 );
13     SUBTYPE cmx_table_name IS VARCHAR2( 30 );
14     SUBTYPE cmx_column_name IS VARCHAR2( 30 );
15     SUBTYPE cmx_data_value IS VARCHAR2( 1000 );
16     SUBTYPE cmx_object_name IS VARCHAR2( 128 );
17     SUBTYPE cmx_object_desc IS VARCHAR2( 255 );
18     SUBTYPE cmx_system_name IS VARCHAR2( 50 );
19     SUBTYPE cmx_status_msg IS VARCHAR2( 1024 );
20     SUBTYPE cmx_date_string IS VARCHAR2( 100 );
21     SUBTYPE cmx_user_name IS VARCHAR2( 50 );
22     SUBTYPE cmx_single_char IS VARCHAR2( 1 );
23     SUBTYPE cmx_job_id IS CHAR( 14 );
24     SUBTYPE cmx_error_code IS INT;
25     SUBTYPE cmx_error_message IS VARCHAR2( 4000 );
26
27     -- constants for job status, c_repos_job_status_type
28     SUCCESS INT := 0;
29     WARNING INT := 1;
30     PROCESSING INT := 2;
31     FAILED INT := 99;
32     INCOMPLETE INT := 4;
33     CLEANSE_JOB_FAILED EXCEPTION;
34     LOAD_JOB_FAILED EXCEPTION;
35     MATCHMERGE_JOB_FAILED EXCEPTION;
36     DDL_ERROR EXCEPTION;
37     SQL_DDL_ERROR EXCEPTION;
38     JOB_NOT_VALID EXCEPTION;
39     NO_INDEX_START_TABLE_NAME EXCEPTION;
40     BAD_PARAMETER EXCEPTION;
41     GLOBAL_INITIALIZATION_FAILED EXCEPTION;
42     READ_CONFIG_FILE_FAILED EXCEPTION;
43     GENERIC_ERROR EXCEPTION;
44
45     out_error_code cmx_error_code;
46     out_error_message cmx_error_message;
47
48     -- Global variables
49     job_print_out BOOLEAN := true; -- print messages to console/log file?
50     job_print_debug BOOLEAN := true; -- print messages to the siperian
51     debug log
52
53     -- Procedure to print to console or siperian debug log
54     PROCEDURE x_print (in_msg VARCHAR2);
55
56     --- Generic procedure to initialize all global variables ---
57     PROCEDURE global_initialize(
58         in_host_name VARCHAR2,
59         in_tns_name VARCHAR2,
60         in_db_user VARCHAR2,
61         out_error_code OUT cmx_error_code,

```

```

62         out_error_message          OUT          cmx_error_message );
63
64     -- Control Procedure to execute Stage, Load, Match and Merge jobs --
65     PROCEDURE exec_mrm_process(
66         in_base_table_name  IN   VARCHAR2,
67         in_stg_table_name   IN   VARCHAR2,
68         in_job_type         IN   VARCHAR2,
69         in_host_name        IN   VARCHAR2,
70         in_tns_name         IN   VARCHAR2,
71         in_db_user          IN   VARCHAR2,
72         out_error_code      OUT   cmx_error_code,
73         out_error_message   OUT   cmx_error_message );
74
75     --- Generic procedure to run Stage jobs ---
76     PROCEDURE run_cleanse(
77         in_table_name              VARCHAR2,
78         out_error_code             OUT   cmx_error_code,
79         out_error_message          OUT   cmx_error_message );
80
81     --- Generic procedure to run Load jobs ---
82     PROCEDURE run_load(
83         in_table_name              VARCHAR2,
84         in_force_update_ind        INT,
85         out_error_code             OUT   cmx_error_code,
86         out_error_message          OUT   cmx_error_message );
87
88     --- Generic procedure to run Match and Merge jobs ---
89     PROCEDURE run_matchmerge(
90         in_table_name              VARCHAR2,
91         in_match_set              VARCHAR2,
92         out_error_code             OUT   cmx_error_code,
93         out_error_message          OUT   cmx_error_message );
94
95     PROCEDURE process_load(
96         in_table_name              VARCHAR2,
97         out_error_code             OUT   cmx_error_code,
98         out_error_message          OUT   cmx_error_message );
99
100    PROCEDURE process_stage(
101        in_table_name              VARCHAR2,
102        out_error_code             OUT   cmx_error_code,
103        out_error_message          OUT   cmx_error_message );
104
105    PROCEDURE process_matchmerge(
106        in_table_name              VARCHAR2,
107        out_error_code             OUT   cmx_error_code,
108        out_error_message          OUT   cmx_error_message );
109
110    PROCEDURE execute_batchgroup(
111        in_mrm_server              VARCHAR2,
112        in_batch_group             VARCHAR2,
113        in_autosys_id              VARCHAR2,
114        in_autosys_password        VARCHAR2,
115        out_error_code             OUT   cmx_error_code,
116        out_error_message          OUT   cmx_error_message );
117
118    END;
119    /
120

```

```

1 CREATE OR REPLACE PACKAGE BODY ADD_ORS.ADD_TASK_ASSIGN AS
2 addtask_pkb_version cmxlb.cmx_med_str := 'Version 2.0 (2013-03-13 16:32:08)';
3 addtask_dsi_name cmxlb.cmx_med_str := 'ADDApp'; -- Name of
our app
4 addtask_sa_name cmxlb.cmx_med_str := 'TransactionalCustomer'; -- Name of
the SA that will be attached to the task
5 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
6 addtask_tsk_type cmxlb.cmx_med_str := 'AssignHier'; --
Task type to use when creating task
7 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
8 addtask_x_sender cmxlb.cmx_med_str := 'NewCustomerAssignment@abbott.com';
-- Email address of the sender
9 l_mesg varchar2(200);
10 --
11 --Version History:
12 --Version 1.0 (2012-26-06 15:52:00) initial version;
13 --Version 2.0 (2013-03-13 16:32:08) SCK Generate tasks and assign to users based on Area
and ISO Country Code
14 FUNCTION show_version
15 RETURN cmxlb.cmx_med_str
16 AS
17 BEGIN
18 RETURN addtask_pkb_version;
19 END;
20 --
21 --
22 PROCEDURE get_zip
23 (
24 in_party_acct_id varchar2,
25 out_zip_postal_cd OUT varchar2
26 )
27 AS
28 --
29 BEGIN
30
31 SELECT distinct NVL(AD.ZIP_POSTAL_CD, NULL) into out_zip_postal_cd FROM C_BO_PTAC_ADDR x,
C_BO_ADDRESS ad
32 where
33 AD.ROWID_OBJECT = X.ADDR_ID and
34 X.PARTY_ACCT_ID = in_party_acct_id;
35 IF out_zip_postal_cd IS NOT NULL THEN
36 out_zip_postal_cd := ':' || out_zip_postal_cd;
37 END IF;
38
39 END get_zip;
40 PROCEDURE existing_add_tam
41 (
42 in_party_acct_id varchar2,
43 existing_relationships OUT NUMBER
44 )
45 AS
46 BEGIN
47 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
48 WHERE PA.ROWID_OBJECT = HPR.ROWID_BO_PARTY_ACCOUNT AND
49 HPR.ROWID_HM_ACCOUNTS = HM.ROWID_OBJECT AND
50 HPR.REL_TYPE_CODE = 'CG1 is parent of Customer' AND
51 HM.ADD_TAM = 'Y' AND
52 PA.ROWID_OBJECT = in_party_acct_id;
53

```

```

54 END existing_add_tam;
55 FUNCTION GET_PARAMETER(in_name VARCHAR2, in_dft VARCHAR2) RETURN VARCHAR2 AS
56     result VARCHAR2(255);
57 BEGIN
58     SELECT VALUE INTO result FROM C_PARAMETERS WHERE NAME = in_name;
59     return result;
60 EXCEPTION
61     WHEN OTHERS THEN
62         RETURN in_dft;
63 END;
64
65 FUNCTION send_mail (in_sender varchar2, in_recepients varchar2, in_subject varchar2,
in_message varchar2)
66     RETURN BOOLEAN
67 AS
68     db_name          VARCHAR2(100) := '';
69 BEGIN
70     select user || ' on ' || sys_context('USERENV', 'DB_NAME') into db_name from dual;
71
72     UTL_MAIL.SEND (
73         sender => in_sender,
74         recipients => in_recepients,
75         subject => in_subject || ' ' || db_name,
76         message => in_message);
77     return TRUE;
78 EXCEPTION
79     WHEN OTHERS THEN
80         dbms_output.put_line('Error sending email: ' || sqlcode || '-' || sqlerrm);
81         cmxut.debug_print (l_mesg);
82         return FALSE;
83 END;
84 --
85 --
86 PROCEDURE exception_report (in_email varchar2)
87 AS
88 cursor c_ptac is
89     select iso_alpha3_cd, count(*) cnt
90     from   c_bo_ptac_txn_div dv, c_bo_party_acct pa
91     where
92         PA.ROWID_OBJECT = DV.PARTY_ACCT_ID and
93         dv.cust_match_status = 'E' and pa.bo_class_code='Transactional Customer'
94     group by iso_alpha3_cd
95     order by 1;
96 --
97 l_count integer := 0;
98 l_body varchar2(6000);
99 l_subject varchar2(100) := 'Exception Report - unassigned countries for ' || to_char(sysdate
, 'MM/DD/YYYY');
100 l_recepients varchar2(255);
101 BEGIN
102     l_body          := 'Country    Unassigned New' || chr(10);
103     l_body:= l_body || '   Code      Customers      ' || chr(10);
104     l_body := l_body || '-----' || chr(10);
105     --
106     FOR r_ptac in c_ptac LOOP
107         l_body := l_body || rpad(' ', 3) || r_ptac.iso_alpha3_cd || rpad(' ', 15) || r_ptac.
cnt || chr(10);
108         l_count := l_count + r_ptac.cnt;
109     END LOOP;
110     IF l_count = 0 THEN
111         -- No exceptions
112         dbms_output.put_line('No exceptions were found');

```

```

113         ELSE
114             l_body := chr(10) || l_body || chr(10) || 'Total unassigned new customers: ' ||
l_count;
115         dbms_output.put_line('Subject: ' || l_subject);
116         dbms_output.put_line(l_body);
117         l_receipients := GET_PARAMETER('TASK Exception Email', addtask_x_email);
118         IF send_mail(addtask_x_sender, l_receipients, l_subject, l_body) THEN
119             dbms_output.put_line('Email sent');
120         ELSE
121             dbms_output.put_line('Error sending email');
122         END IF;
123     END IF;
124 END;
125
126 FUNCTION get_user_for_country(in_cntry_cd varchar2)
127     RETURN cmxlb.cmx_rowid
128 AS
129     l_rowid_cntry cmxlb.cmx_rowid;
130     l_rowid_user cmxlb.cmx_rowid;
131     l_user_name varchar2(50);
132     l_global_loc_cd C_LU_ISO_CNTRY.GLOBAL_LOC_CD%TYPE;
133     l_user_found BOOLEAN;
134     l_area_found BOOLEAN;
135 BEGIN
136     -- Get the rowid and area for the country
137     l_mesg := 'Getting rowid and area for country: ' || in_cntry_cd;
138     dbms_output.put_line(l_mesg);
139     BEGIN
140         SELECT ROWID_OBJECT, GLOBAL_LOC_CD
141         INTO     l_rowid_cntry, l_global_loc_cd
142         FROM     C_LU_ISO_CNTRY
143         WHERE
144             iso_alpha3_cd=in_cntry_cd;
145     EXCEPTION
146         WHEN NO_DATA_FOUND THEN
147             -- Invalid country!
148             l_mesg := 'Country code not found! ' || in_cntry_cd;
149             return NULL;
150     END;
151     -- Get the user associated with the country that has the earliest create date
152     l_mesg := 'Looking for assignment for country: ' || in_cntry_cd || ' rowid country: ' ||
l_rowid_cntry;
153     dbms_output.put_line(l_mesg);
154     l_user_found := FALSE;
155     BEGIN
156         SELECT USER_NAME
157         INTO     l_user_name
158         FROM     (
159             SELECT A.USER_NAME FROM
160             C_BO_DSR_USER A, C_BO_DSR_USER_ISO_CNTRY B
161             WHERE
162                 B.ROWID_ISO_CNTRY=l_rowid_cntry AND
163                 B.ROWID_DSR_USER=A.ROWID_OBJECT AND
164                 NVL(B.DELETED_IND,0)=0 AND
165                 NVL(B.HUB_STATE_IND,1)=1 AND
166                 NVL(A.HUB_STATE_IND,1)=1 AND
167                 A.USER_ENABLED_IND=1
168             ORDER BY B.CREATE_DATE)
169         WHERE ROWNUM < 2;
170         l_user_found := TRUE;
171         l_mesg := 'Country level assignment found, rowid_user: ' || l_rowid_user;
172         dbms_output.put_line(l_mesg);

```

```

173 EXCEPTION
174     WHEN NO_DATA_FOUND THEN
175         l_user_found := FALSE;
176 END;
177 --
178 IF NOT l_user_found THEN
179     -- No active user was found for the country - get the user that has the area
responsibility
180     l_mesg := 'Did not find assignment for country: ' || in_cntry_cd || ' checking
area level ' || l_global_loc_cd;
181     dbms_output.put_line(l_mesg);
182     l_area_found := FALSE;
183     --
184 BEGIN
185     SELECT USER_NAME
186     INTO     l_user_name
187     FROM     (
188                 SELECT A.USER_NAME FROM
189                 C_BO_DSR_USER A, C_BO_DSR_USER_ISO_CNTRY B
190                 WHERE
191                     B.ROWID_ISO_CNTRY IS NULL AND
192                     B.GLOBAL_LOC_CD=l_global_loc_cd AND
193                     B.ROWID_DSR_USER=A.ROWID_OBJECT AND
194                     NVL(B.DELETED_IND,0)=0 AND
195                     NVL(B.HUB_STATE_IND,1)=1 AND
196                     NVL(A.HUB_STATE_IND,1)=1 AND
197                     A.USER_ENABLED_IND=1
198                 ORDER BY B.CREATE_DATE)
199     WHERE ROWNUM < 2;
200     l_area_found := TRUE;
201     l_mesg := 'Area level assignment found, rowid_user: ' || l_rowid_user;
202     dbms_output.put_line(l_mesg);
203 EXCEPTION
204     WHEN NO_DATA_FOUND THEN
205         l_area_found := FALSE;
206 END;
207 --
208 --
209 END IF;
210 IF NOT (l_area_found or l_user_found) THEN
211     BEGIN
212         l_mesg := 'Did not find either country or area level assignment for country: ' ||
in_cntry_cd;
213         dbms_output.put_line(l_mesg);
214         cmxut.debug_print (l_mesg);
215         return NULL;
216     END;
217 END IF;
218 --
219 BEGIN
220     l_mesg := 'Looking up user: ' || l_user_name;
221     dbms_output.put_line(l_mesg);
222     cmxut.debug_print (l_mesg);
223     SELECT ROWID_USER
224     INTO     l_rowid_user
225     FROM     C_REPOS_USER
226     WHERE
227         user_name = l_user_name;
228 EXCEPTION
229     WHEN NO_DATA_FOUND THEN
230         l_mesg := 'User not found in C_REPOS_USER! : ' || l_user_name;
231         dbms_output.put_line(l_mesg);

```

```

232         cmxut.debug_print (l_mesg);
233         return NULL;
234     END;
235     RETURN l_rowid_user;
236 END;
237 --
238 PROCEDURE add_assign_tasks(
239     in_max_tasks IN INT,
240     out_unassigned_task_count OUT INT,
241     out_assigned_task_count OUT INT,
242     out_error_msg OUT cmxlb.cmx_message,
243     out_return_code OUT INT)
244 AS
245     -- 20130313 SCK Generate tasks and assign to users based on Area and ISO Country
    Code
246     lock_applied BOOLEAN;
247     l_rowid_sa cmxlb.cmx_rowid;
248     l_rowid_tbl cmxlb.cmx_rowid;
249     l_rowid_user cmxlb.cmx_rowid;
250     l_task_title cmxlb.cmx_big_str;
251     l_rowid_task cmxlb.cmx_rowid;
252     l_match_status char(1);
253     zip_postal_cd varchar2(36);
254     dueDateDay NUMBER;
255     existing_relationships NUMBER;
256     --
257     CURSOR c_txn
258     IS
259         SELECT TX.TXN_DIV_CD,
260                TX.CUST_MATCH_STATUS,
261                TX.ROWID_OBJECT,
262                TX.PARTY_ACCT_ID,
263                TX.HUB_STATE_IND,
264                PA.CUST_NBR_PK,
265                PA.CUST_NM,
266                PA.ISO_ALPHA3_CD
267         FROM C_BO_PTAC_TXN_DIV TX, C_BO_PARTY_ACCT PA
268        WHERE TX.PARTY_ACCT_ID = PA.ROWID_OBJECT AND
269              TX.CUST_MATCH_STATUS IN ('N', 'E') AND
270              TX.TXN_DIV_CD = 'ADD' AND
271              NVL(TX.HUB_STATE_IND,1) = 1
272        FOR UPDATE ;
273     --
274     BEGIN
275         dueDateDay := TO_NUMBER (GET_PARAMETER ('DUE DATE DAYS', '7'));
276         l_mesg := 'due days: ' || dueDateDay;
277         BEGIN
278             SELECT rowid_subject_area
279             INTO l_rowid_sa
280             FROM c_repos_subject_area
281            WHERE dsi_name = addtask_dsi_name AND name = addtask_sa_name;
282         END;
283         l_mesg := l_mesg || ' sa: ' || TRIM(l_rowid_sa);
284         BEGIN
285             SELECT rowid_table
286             INTO l_rowid_tbl
287             FROM c_repos_table
288            WHERE table_name = addtask_tbl_name;
289         END;
290         l_mesg := l_mesg || ' table: ' || TRIM(l_rowid_tbl);
291         dbms_output.put_line(l_mesg);
292         cmxut.debug_print (l_mesg);

```



```

293         FOR r_txn IN c_txn LOOP
294             existing_add_tam(r_txn.PARTY_ACCT_ID, existing_relationships);
295             l_mesg := 'tam_relates: ' || existing_relationships;
296             IF existing_relationships = 0 THEN
297                 l_rowid_user := get_user_for_country(r_txn.iso_alpha3_cd);
298                 l_mesg := l_mesg || ' user_iso: ' || TRIM(l_rowid_user) || ' _' ||
r_txn.iso_alpha3_cd;
299                 IF l_rowid_user IS NULL THEN
300                     l_match_status := 'E';
301                 ELSE
302                     get_zip(r_txn.PARTY_ACCT_ID, zip_postal_cd);
303                     BEGIN
304                         l_task_title := r_txn.cust_nbr_pk || ':' || r_txn.cust_nm
|| ':' || r_txn.iso_alpha3_cd || zip_postal_cd;
305                         l_mesg := 'Creating task with title: ' || l_task_title ||
' for rowid_user: ' || l_rowid_user;
306                         dbms_output.put_line(l_mesg);
307                         cmxut.debug_print (l_mesg);
308                         l_rowid_task := c_repos_task_seq.nextval;
309                         --
310                         l_mesg := 'Inserting into c_repos_task_assignment';
311                         dbms_output.put_line(l_mesg);
312                         cmxut.debug_print (l_mesg);
313                         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)
314                             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);
315                         END;
316                         -- Create task data
317                         BEGIN
318                             l_mesg := 'Inserting into c_repos_task_data';
319                             dbms_output.put_line(l_mesg);
320                             cmxut.debug_print (l_mesg);
321                             INSERT INTO C_REPOS_TASK_DATA (ROWID_OBJECT, ROWID_TABLE,
ROWID_TASK, ROWID_TASK_DATA)
322                                 VALUES (r_txn.PARTY_ACCT_ID, l_rowid_tbl, l_rowid_task,
c_repos_task_data_seq.nextval);
323                             END;
324                             l_match_status := 'A';
325                         END IF;
326                     ELSE
327                         l_match_status := 'A';
328                     END IF;
329                     l_mesg := l_mesg || ' c_m_status: ' || l_match_status;
330                     dbms_output.put_line(l_mesg);
331                     cmxut.debug_print (l_mesg);
332                     --
333                     UPDATE C_BO_PTAC_TXN_DIV
334                     SET CUST_MATCH_STATUS = l_match_status
335                     WHERE ROWID_OBJECT = r_txn.rowid_object;
336                     --
337                     UPDATE C_BO_PTAC_TXN_DIV_XREF
338                     SET CUST_MATCH_STATUS = l_match_status
339                     WHERE ROWID_OBJECT = r_txn.rowid_object;
340                 END LOOP;
341                 COMMIT;
342                 exception_report(addtask_x_email);
343             EXCEPTION
344

```

```
345         WHEN OTHERS THEN
346             dbms_output.put_line(l_mesg);
347             cmxut.debug_print (l_mesg);
348             dbms_output.put_line(sqlcode || '-' || sqlerrm);
349             cmxut.debug_print (sqlcode || '-' || sqlerrm);
350             out_error_msg := 'ERROR ' || l_mesg || '::' || sqlcode || '-' || sqlerrm;
351             out_return_code:=-10222;
352         END;
353     END ADD_TASK_ASSIGN;
354 /
355
```