```
1
2  set serverout on;
3
4
5
6  CREATE OR REPLACE
7  PACKAGE BODY PK_WCUNEX_PACK
8  AS
9  WS_ERR_CODE  NUMBER;
10 WS_ERR_MSG  VARCHAR2(20);
11
12
13 --*************************************************************************************
14 --
15 --  NAME:    WCUNEX_PACK_BODY.PKB
16 --  AUTHOR:  JAMES CRAIG
17 --  DATE:    10-JUN-2009
18 --  SCHEMA:  PROD
19 --  PURPOSE:
20 --
21 --  FUNCTIONS:
22 --
23 --  PROCEDURES:  TEMP_UPDATE_TUNX, TEMP_INSERT_UNEX, DETERMINE_AFTERMOD,
24 --               CALCULATE_TTL_PREM, CALCULATE_TTL_EXPOSURE, INSERT_EXPOSURES,
25 --               RESET_PERIOD_SPLIT
26 --
27 --
28 --  MODIFICATION HISTORY:
29 --  PROJECT NUMBER  DATE          INITIALS    GENERAL COMMENTS / OBJECT(S) MODIFIED
30 --  --------------  ----------     --------    ---------------------------------------
31 --
32 --  PLI32502.12477   25-JUN-2009    JPC        INITIAL CREATION
33 --  PLI32504.15785   18-SEP-2009    JPC        MODIFIED TEMP_UPDATE_TUNX,
   DETERMINE_AFTERMOD
34 --  PLI32502.16157   19-OCT-2009    JPC        MODIFIED DETERMINE_AFTERMOD
35 --  PLI32502.16356   08-DEC-2009    JPC        Add AUTHID statement to spec
36 --  PLI32502.16650   16-DEC-2009    JPC        MODIFIED DETERMINE_AFTERMOD; use
   P_SEND_MAIL for error-trapping
37 --  PLI32502.16866   06-JAN-2010    JPC       Add sqlcode for error-trapping
38 --  PLI32502.16991   14-JAN-2010    JPC       Modified DETERMINE_AFTERMOD
39 --  PLI32502.17367   08-MAR-2010    JPC        Change codes in DETERMINE_AFTERMOD
40 --  PLI32502.19870   05-JAN-2011    JPC       Add exposure transmit indicator
41 --  PLI32502.20830   14-APR-2011    JPC        Modify code assignemnts (GET_EXPO_XMIT)
42 --  PLI32502.20637   25-APR-2011    JPC        Modify code assignemnts
   (DETERMINE_AFTERMOD)
43 --  PLI32502.21102   23-MAY-2011    JPC        More code assignments (DETERMINE_AFTERMOD)
44 --  PLI32502.21210   10-JUN-2011    JPC       Add Determine_Split_Period
45 --  PLI32502.21570   10-AUG-2011    JPC        Change 9885 to A for DE
46 --  PLI32502.21724   14-SEP-2011    JPC       Modify code assignment (DETERMINE_AFTERMOD)
47 --  PLI32502.22149   29-NOV-2011    JPC        Change 0990 to N for PA
   (DETERMINE_AFTERMOD)
48 --  PLI32502.21661   15-DEC-2011    JPC        Add Code 0988 (GET_EXPO_XMIT)
49 --  PLI32502.22495   23-JAN-2012    JPC       Change 9887 to N for DE
50 --  PLI32502.23503   20-AUG-2012    JPC        9885,9887 are N for DE
51 --  PLI32502.23513   20-AUG-2012    JPC        Add 0990 and 0931 for NY
52 --  PLI32502.23835   20-SEP-2012    JPC       Correct 0990 mod indicator
```

```
53  -- PLI32502.23833   24-SEP-2012    JPC        Change 0887 to 'E' for MA
    (DETERMINE_AFTERMOD)
54  -- PLI32502.23933   09-OCT-2012    JPC        Make mod indicator for 0120 'N'
    (DETERMINE_AFTERMOD)
55  -- PLI32502.24598   23-JAN-2013    JPC        ADDITION OF ANNIV DATE
    (Determine_Split_Period)
56  -- PLI32502.24619   11-APR-2014    JPC        alter DETERMINE_AFTERMOD, GET_EXPO_XMIT
57  -- PLI32502.27508   25-APR-2014    JPC        Add State exclusions (Determine_Split_Period)
58  -- PLI32502.28178   19-AUG-2014    JPC        9664 Exception for NC (DETERMINE_AFTERMOD)
59  -- PLI32502.28206   27-AUG-2014    JPC        Corrections for 0063 and 9846
60  -- PLI32502.28448   15-OCT-2014    JPC        Fix DevPx for Mass.
61  -- PLI32502.29027   22-JAN-2015    JPC        CA 0930 Corrections
62  -- PLI32502.29821   01-JUL-2015    JPC        Change logic for Determine_Split_Period
63  -- PLI32502.30030   20-AUG-2018    JPC        Use cursor DEVIATED_PREM
64  --*****************************************************************************************
65
66
67  --
68  --#########################################################################################
    ########
69  --
70  -- NAME:       TEMP_UPDATE_TUNX
71  -- AUTHOR:
72  -- DATE:
73  -- PURPOSE:
74  --
75  -- INPUTS:
76  --
77  -- OUTPUTS:
78  --
79  -- MODIFICATION HISTORY:
80  -- PROJECT NUMBER  DATE           INITIALS    COMMENT
81  -- --------------  -----------     --------    ---------------------------------------
82  -- PLI32502.12477   11-JUN-2009    JPC        INITIAL CREATION
83  -- PLI32502.15785   18-SEP-2009    JPC        ADDED EXCEPTION HANDLING
84  --
    ########################################################################################
    ######
85  PROCEDURE TEMP_UPDATE_TUNX
86        (
87              SPLIT_NO  IN NUMBER,
88              POLICY_NO IN VARCHAR2,
89              EXPO_EFF  IN DATE,
90              WS_SQLCODE      IN OUT NUMBER )
91  IS
92
93  bad_split_proc exception;
94
95
96  BEGIN
97         UPDATE TEMP_UNIT_EXPO
98        SET SPLIT_IND               = SPLIT_NO
99          WHERE POLICY_NUMBER        = POLICY_NO
100             AND EXPOSURE_EFFECTIVE = EXPO_EFF;
101
102
```

```
103  if sqlcode <> 0 then RAISE bad_split_proc;
104  END IF;
105
106
107    EXCEPTION
108     WHEN bad_split_proc THEN
109       WS_ERR_CODE := sqlcode;
110       ws_err_msg := 'bad_split_proc';
111       P_SEND_MAIL ('MAIL.FLORISTSMUTUAL.COM',
112             'FLORISTSMUTUAL.COM',
113             'WCUNEX_PACK_BODY',
114             'NO_REPLY@HORTICA-INSURANCE.COM',
115             'CRAIG, JAMES' ,
116             'JCRAIG@HORTICA-INSURANCE.COM',
117             'PROCESS FAILED! '||WS_ERR_CODE,
118             WS_ERR_MSG||' FAILED: '||POLICY_NO,
119             WS_ERR_CODE);
120        --RAISE;
121      WHEN OTHERS THEN
122          WS_ERR_CODE := SQLCODE;
123          WS_ERR_MSG := 'TEMP_UPDATE_TUNX';
124       P_SEND_MAIL ('MAIL.FLORISTSMUTUAL.COM',
125             'FLORISTSMUTUAL.COM',
126             'WCUNEX_PACK_BODY',
127             'NO_REPLY@HORTICA-INSURANCE.COM',
128             'CRAIG, JAMES' ,
129             'JCRAIG@HORTICA-INSURANCE.COM',
130             'PROCESS FAILED! '||WS_ERR_CODE,
131             WS_ERR_MSG||' FAILED: '||POLICY_NO,
132             WS_ERR_CODE);
133        --RAISE;
134  END;
135  --
136  --##############################################################################
     ########
137  --
138  --  NAME:       TEMP_INSERT_UNEX
139  --  AUTHOR:
140  --  DATE:
141  --  PURPOSE:
142  --
143  --  INPUTS:
144  --
145  --  OUTPUTS:
146  --
147  --  MODIFICATION HISTORY:
148  --  PROJECT NUMBER  DATE           INITIALS    COMMENT
149  --  --------------  -----------     --------    ---------------------------------------
150  --  PLI32502.12477   11-JUN-2009    JPC         INITIAL CREATION
151  --
152  --
     ##############################################################################
     ######
153  PROCEDURE TEMP_INSERT_UNEX
154        (
155             UNEX_ID IN NUMBER,
```

```
156             UNPL_ID IN NUMBER,
157             RPT_DTE IN DATE,
158             UNEX_ST IN VARCHAR2,
159             CLASSCD IN VARCHAR2,
160             MODVAL  IN NUMBER,
161             MODEFF  IN DATE,
162             BURTEFF IN DATE,
163             PAYROLL IN NUMBER,
164             NET_PX  IN NUMBER,
165             MOD_PX  IN NUMBER,
166             SPLITNO IN NUMBER,
167             BURT_RT IN NUMBER,
168             AFTRMOD IN VARCHAR2,
169             xmit_yn in varchar2,
170             WS_SQLCODE      IN OUT NUMBER)
171   IS
172
173
174
175   BEGIN
176
177         INSERT
178          INTO UNIT_STATS_EXPOSURE
179             (
180                  UNEX_WC_NO            ,
181                  UNEX_UNPL_WC_NO       ,
182                  UNEX_WC_REPORT_DATE   ,
183                  UNEX_STAT_GB_STATE_NO ,
184                  UNEX_WC_EXIT_CLASS_CODE ,
185                  UNEX_WC_PREV_REP_IND  ,
186                  UNEX_WC_EXIT_COVG_CODE ,
187                  UNEX_WC_EXP_MOD       ,
188                  UNEX_WC_EXP_MOD_EFF_DATE ,
189                  UNEX_WC_BURT_EFF_DATE ,
190                  UNEX_WC_EXIT_COV_AMOUNT ,
191                  UNEX_WC_EXIT_NET_PREM ,
192                  UNEX_WC_EXIT_MOD_NET_PREM,
193                  UNEX_WC_SURC_RATE     ,
194                  UNEX_WC_SPLIT_PERIOD_IND ,
195                  UNEX_WC_BURT_RATE     ,
196                  UNEX_WC_RATING_TIER_ID ,
197                  UNEX_WC_UPDATE_TYPE   ,
198                  UNEX_WC_EXIT_ACT      ,
199                  UNEX_WC_RECORD_TYPE   ,
200                  UNEX_XMIT_IND
201             )
202             VALUES
203             (
204                  UNEX_ID                                    ,
205                  UNPL_ID                                    ,
206                  RPT_DTE                                    ,
207                  UNEX_ST                                    ,
208                  CLASSCD                                    ,
209                  0                                          ,
210                  DECODE (CLASSCD, '0900', '00', '0063', '00', '0935', '00', '01'),
211                  DECODE (CLASSCD, '0900', '00', '0063', '00', MODVAL)          ,
```

```
                MODEFF                                          ,
                BURTEFF                                         ,
                PAYROLL                                         ,
                NET_PX                                         ,
                MOD_PX                                          ,
                0                                             ,
                SPLITNO                                        ,
                BURT_RT                                         ,
                ' '                                           ,
                'R'                                             ,
                NULL                                             ,
                AFTRMOD                                           ,
                xmit_yn
            ) ;
  EXCEPTION
    WHEN OTHERS THEN
        WS_ERR_CODE := sqlcode;
        ws_err_msg := 'TEMP_INSERT_UNEX';
     P_SEND_MAIL ('MAIL.FLORISTSMUTUAL.COM',
            'FLORISTSMUTUAL.COM',
            'WCUNEX_PACK_BODY',
            'NO_REPLY@HORTICA-INSURANCE.COM',
            'CRAIG, JAMES' ,
            'JCRAIG@HORTICA-INSURANCE.COM',
            'PROCESS FAILED! '||WS_ERR_CODE,
            WS_ERR_MSG||' FAILED: '||UNPL_ID||' expoid: '||UNEX_ID||' class: '||classcd,
            WS_ERR_CODE);
      --RAISE;
END TEMP_INSERT_UNEX;
--
--##############################################################################
########
--
--  NAME:      DETERMINE_AFTERMOD
--  AUTHOR:
--  DATE:
--  PURPOSE:
--
--  INPUTS:
--
--  OUTPUTS:
--
--  MODIFICATION HISTORY:
--  PROJECT NUMBER  DATE          INITIALS    COMMENT
--  --------------  ----------    --------    ----------------------------------------
--  PLI32502.12477   11-JUN-2009    JPC        INITIAL CREATION
--  PLI32502.15785   18-SEP-2009    JPC        Added 0990, 9874 to New York (31) list
--  PLI32502.16157   19-OCT-2009    JPC        Added 0990 to Michigan (21) list; added 0032 and
0887 to Massachusetts (20)
--  PLI32502.16650   14-DEC-2009    JPC        Redesignate various class codes
--  PLI32502.16991   14-JAN-2010    JPC        Edit designations for CA class codes
--  PLI32502.17367   08-MAR-2010    JPC        Change codes for 0174 and 9889 in MN
--  PLI32502.20637   25-APR-2011    JPC        Add 0930 to assessment group
--  PLI32502.21102   23-MAY-2011    JPC        Change E/N/A designations
--  PLI32502.21570   10-AUG-2011    JPC        Change 9885 to A for DE
--  PLI32502.21724   14-SEP-2011    JPC        Change 9887 to N for PA
```

```
-- PLI32502.22149   29-NOV-2011    JPC       Change 0990 to N for PA
-- PLI32502.22495   23-JAN-2012    JPC       Change 9887 to N for DE
-- PLI32502.22496   23-JAN-2012    JPC       Change 9889 to N for PA
-- PLI32502.23503   20-AUG-2012    JPC       9885,9887 are N for DE
-- PLI32502.23513   20-AUG-2012    JPC       Add 0990 and 0931 for NY
-- PLI32502.23835   20-SEP-2012    JPC       Correct 0990 mod indicator for NY
-- PLI32502.23833   24-SEP-2012    JPC       Change 0887 to 'E' for MA (20)
-- PLI32502.23933   09-OCT-2012    JPC       Make mod indicator for 0120 'N'
-- PLI32502.23835   24-OCT-2012    JPC       Moving changes
-- PLI32502.24619   11-APR-2014    JPC       alter NY, PA, WI class codes; add 9846
-- PLI32502.28178   19-AUG-2014    JPC       9664 Exception for NC
-- PLI32502.28206   27-AUG-2014    JPC       Corrections for 0063 and 9846
--
################################################################################
######
FUNCTION DETERMINE_AFTERMOD (
            WS_AFTERMOD CHAR,
            WS_CLASS    CHAR,
            WS_STATE_NO CHAR,
            WS_STDPX    NUMBER)
      RETURN CHAR
IS
      MOD_INDICATOR VARCHAR2 (1) ;
BEGIN
      IF WS_AFTERMOD       = 'Y' THEN
            MOD_INDICATOR := 'E';
      ELSIF WS_AFTERMOD     = 'N' THEN
            MOD_INDICATOR := 'A';
      ELSE
            MOD_INDICATOR := 'N';
      END IF;

      --
      if SUBSTR (WS_CLASS, 1, 4)    IN ('0120', '9846') then
            MOD_INDICATOR         := 'N';
      elsif SUBSTR (WS_CLASS, 1, 4)    IN ('0063') then
            MOD_INDICATOR         := 'A';
      end if;

      --
if ws_state_no not in ('04') then
      IF SUBSTR (WS_CLASS, 1, 4)    IN ('9722', '9724', '9950', '9658', '0063') THEN
            MOD_INDICATOR         := 'N';
      ELSIF SUBSTR (WS_CLASS, 1, 4) IN ('9139', '9896') THEN
            MOD_INDICATOR         := 'A';
      ELSIF SUBSTR (WS_CLASS, 1, 4) IN ('0930') THEN
            MOD_INDICATOR         := 'E';
      END IF;
else
    IF SUBSTR (WS_CLASS, 1, 4) IN ('9139', '9896') THEN
            MOD_INDICATOR         := 'A';
    ELSIF SUBSTR (WS_CLASS, 1, 4)    IN ('9950', '9658') THEN
            MOD_INDICATOR         := 'N';
    END IF;
end if;
      --
```

```
320
321
322
323         --
324         IF WS_STATE_NO IN ('04') THEN
325               -- (04 CA)
326               IF SUBSTR (WS_CLASS, 1, 4)     in ('0990', '9722', '9724', '9812', '9848', '0930') THEN
327                     MOD_INDICATOR          := 'N';
328               END IF;
329               --
330         ELSIF WS_STATE_NO IN ('07') THEN
331               -- (07 DE)
332               IF SUBSTR (WS_CLASS, 1, 4) IN ('9885','9887') THEN
333                     MOD_INDICATOR       := 'N';
334               END IF;
335               --
336         ELSIF WS_STATE_NO IN ('10') THEN
337               -- (10 GA)
338               IF SUBSTR (WS_CLASS, 1, 4) IN ('9846') THEN
339                     MOD_INDICATOR       := 'N';
340               END IF;
341               --
342         ELSIF WS_STATE_NO IN ('20') THEN
343               -- (20 MA)
344               IF SUBSTR (WS_CLASS, 1, 4)    IN ('9887', '9885', '0990') THEN
345                     MOD_INDICATOR          := 'N';
346               ELSIF SUBSTR (WS_CLASS, 1, 4) IN ('9886', '9889', '0032') THEN
347                     MOD_INDICATOR          := 'A';
348               ELSIF SUBSTR (WS_CLASS, 1, 4) IN ('9037','0887') THEN
349                     MOD_INDICATOR          := 'E';
350               END IF;
351               --
352         ELSIF WS_STATE_NO IN ('21') THEN
353               -- (21 MI)
354               IF SUBSTR (WS_CLASS, 1, 4) IN ('9887', '9889', '0990') THEN
355                     MOD_INDICATOR       := 'N';
356               END IF;
357               --
358         ELSIF WS_STATE_NO IN ('22') THEN
359               -- (22 MN)
360               IF SUBSTR (WS_CLASS, 1, 4) IN ('0174', '0990', '9887', '9889') THEN
361                     MOD_INDICATOR       := 'N';
362               ELSIF SUBSTR (WS_CLASS, 1, 4) IN ('9663') THEN
363                     MOD_INDICATOR       := 'A';
364               END IF;
365               --
366         ELSIF WS_STATE_NO IN ('29') THEN
367               -- (29 NJ)
368               IF SUBSTR (WS_CLASS, 1, 4) IN ('0990', '9887', '9874') THEN
369                     MOD_INDICATOR       := 'N';
370               END IF;
371               --
372         ELSIF WS_STATE_NO IN ('31') THEN
373               -- (31 NY)
374               IF SUBSTR (WS_CLASS, 1, 4)    IN ('9884', '9885', '9886', '9874', '9896', '9046', '0990')
                  THEN
```

```
375                 MOD_INDICATOR         := 'N';
376             ELSIF SUBSTR (WS_CLASS, 1, 4) IN ('9664','0931') THEN
377                 MOD_INDICATOR         := 'E';
378             END IF;
379             --
380         ELSIF WS_STATE_NO IN ('32') THEN
381             -- (32 NC)
382             IF WS_STDPX > 0 or WS_CLASS = '9664' THEN
383               IF SUBSTR (WS_CLASS, 1, 4)    IN ('0930', '9896', '9664') THEN
384                 MOD_INDICATOR         := 'E';
385               ELSIF SUBSTR (WS_CLASS, 1, 4) IN ('0990', '9722', '9724') THEN
386                 MOD_INDICATOR         := 'N';
387               END IF;
388             ELSE
389               MOD_INDICATOR         := 'N';
390             END IF;
391             --
392         ELSIF WS_STATE_NO IN ('35') THEN
393             -- (35 OK)
394             IF SUBSTR (WS_CLASS, 1, 4) IN ('9664') THEN
395                 MOD_INDICATOR       := 'E';
396             END IF;
397             IF SUBSTR (WS_CLASS, 1, 4) IN ('9885') THEN
398                 MOD_INDICATOR       := 'N';
399             END IF;
400             --
401         ELSIF WS_STATE_NO IN ('37') THEN
402             -- (37 PA)
403             IF SUBSTR (WS_CLASS, 1, 4)    IN ('9889','9887', '0931', '0990') then  --,'9890') THEN
404                 MOD_INDICATOR         := 'N';
405             ELSIF SUBSTR (WS_CLASS, 1, 4) IN ( '9664') THEN
406                 MOD_INDICATOR         := 'E';
407             ELSIF SUBSTR (WS_CLASS, 1, 4) IN ( '9890','9885') THEN
408                 MOD_INDICATOR         := 'A';
409             END IF;
410             --
411         ELSIF WS_STATE_NO IN ('48') THEN
412             -- (48 WI)
413             IF SUBSTR (WS_CLASS, 1, 4) IN      ('0990', '9874', '9885', '9886', '9887', '9889') THEN
414                 MOD_INDICATOR       := 'N';
415             ELSIF SUBSTR (WS_CLASS, 1, 4) IN ( '0930', '0106') THEN
416                 MOD_INDICATOR         := 'E';
417             END IF;
418             --
419         ELSIF WS_STATE_NO NOT IN ('07', '21', '22', '32')
420             THEN
421             IF SUBSTR (WS_CLASS, 1, 4) IN ('0990', '9874', '9885', '9886', '9887', '9889') THEN
422                 MOD_INDICATOR       := 'N';
423             END IF;
424             --
425         END IF;
426         RETURN MOD_INDICATOR;
427     EXCEPTION
428     WHEN OTHERS THEN
429         WS_ERR_CODE := SQLCODE;
430         WS_ERR_MSG  := 'DETERMINE_AFTERMOD';
```

```
431       P_SEND_MAIL ('MAIL.FLORISTSMUTUAL.COM',
432                 'FLORISTSMUTUAL.COM',
433                 'WCUNEX_PACK_BODY',
434                 'NO_REPLY@HORTICA-INSURANCE.COM',
435                 'CRAIG, JAMES',
436                 'JCRAIG@HORTICA-INSURANCE.COM',
437                 'PROCESS FAILED! '||WS_ERR_CODE,
438                  WS_ERR_MSG||' FAILED!',
439                  WS_ERR_CODE) ;
440
441 END DETERMINE_AFTERMOD;
442 --
443 --##############################################################################
    ########
444 --
445 --  NAME:      CALCULATE_TTL_PREM
446 --  AUTHOR:
447 --  DATE:
448 --  PURPOSE:
449 --
450 --  INPUTS:
451 --
452 --  OUTPUTS:
453 --
454 --  MODIFICATION HISTORY:
455 --  PROJECT NUMBER  DATE           INITIALS    COMMENT
456 --  --------------  -----------    --------    ---------------------------------------
457 --  PLI32502.12477   11-JUN-2009    JPC        INITIAL CREATION
458 --
459 --
    ##############################################################################
    ######
460 FUNCTION CALCULATE_TTL_PREM
461       (
462            NET_PREM NUMBER,
463            MOD_PREM NUMBER
464       )
465       RETURN NUMBER
466 IS
467       TOTAL_PREM NUMBER;
468
469 BEGIN
470       TOTAL_PREM := NET_PREM + MOD_PREM;
471       RETURN TOTAL_PREM;
472
473   EXCEPTION
474     WHEN OTHERS THEN
475         WS_ERR_CODE := sqlcode;
476         ws_err_msg := 'CALCULATE_TTL_PREM';
477      P_SEND_MAIL ('MAIL.FLORISTSMUTUAL.COM',
478            'FLORISTSMUTUAL.COM',
479            'WCUNEX_PACK_BODY',
480            'NO_REPLY@HORTICA-INSURANCE.COM',
481            'CRAIG, JAMES' ,
482            'JCRAIG@HORTICA-INSURANCE.COM',
483            'PROCESS FAILED! '||WS_ERR_CODE,
```

```
484             WS_ERR_MSG||' FAILED!',
485             WS_ERR_CODE);
486       --RAISE;
487   END CALCULATE_TTL_PREM;
488   --
489   --##############################################################################
      ########
490   --
491   --  NAME:      CALCULATE_TTL_EXPOSURE
492   --  AUTHOR:
493   --  DATE:
494   --  PURPOSE:
495   --
496   --  INPUTS:
497   --
498   --  OUTPUTS:
499   --
500   --  MODIFICATION HISTORY:
501   --  PROJECT NUMBER  DATE           INITIALS    COMMENT
502   --  --------------  -----------    --------    ---------------------------------------
503   --  PLI32502.12477   11-JUN-2009    JPC        INITIAL CREATION
504   --
505   --
      ##############################################################################
      ######
506   FUNCTION CALCULATE_TTL_EXPOSURE
507         (
508             TTL_PREM    NUMBER,
509             TTL_PAYROLL NUMBER
510         )
511         RETURN NUMBER
512   IS
513         TOTAL_EXPOSURE NUMBER;
514
515   BEGIN
516         TOTAL_EXPOSURE := TTL_PREM + TTL_PAYROLL;
517         RETURN TOTAL_EXPOSURE;
518    EXCEPTION
519      WHEN OTHERS THEN
520          WS_ERR_CODE := sqlcode;
521          ws_err_msg := 'CALCULATE_TTL_EXPOSURE';
522       P_SEND_MAIL ('MAIL.FLORISTSMUTUAL.COM',
523             'FLORISTSMUTUAL.COM',
524             'WCUNEX_PACK_BODY',
525             'NO_REPLY@HORTICA-INSURANCE.COM',
526             'CRAIG, JAMES' ,
527             'JCRAIG@HORTICA-INSURANCE.COM',
528             'PROCESS FAILED! '||WS_ERR_CODE,
529             WS_ERR_MSG||' FAILED!',
530             WS_ERR_CODE);
531       --RAISE;
532   END  CALCULATE_TTL_EXPOSURE;
533   --
534   --##############################################################################
      ########
535   --
```

```
536 -- NAME:      INSERT_EXPOSURES
537 -- AUTHOR:
538 -- DATE:
539 -- PURPOSE:
540 --
541 -- INPUTS:
542 --
543 -- OUTPUTS:
544 --
545 -- MODIFICATION HISTORY:
546 -- PROJECT NUMBER  DATE           INITIALS    COMMENT
547 -- -------------  -----------    --------    ---------------------------------------
548 -- PLI32502.12477   11-JUN-2009    JPC        INITIAL CREATION
549 --
550 --
    ##########################################################################################
    ######
551 PROCEDURE INSERT_EXPOSURES
552     (
553         UNEX_ID NUMBER,
554         UNPL_ID NUMBER,
555         RPT_DATE DATE,
556         STATE_NUMBER CHAR,
557         CLASS_CODE   CHAR,
558         MODVAL       NUMBER,
559         MODEFF DATE,
560         RATE_EFF DATE,
561         PAYROLL  NUMBER,
562         NETPREM  NUMBER,
563         MODPREM  NUMBER,
564         SPLIT    NUMBER,
565         THE_RATE NUMBER,
566         USE_MOD  CHAR,
567         POLYNO   CHAR,
568         ANNIV DATE,
569         TTL_EXPO NUMBER,
570         TTL_PREM NUMBER,
571         SECTOR   CHAR,
572         xmit char,
573         WS_SQLCODE      IN OUT NUMBER
574     )
575 IS
576
577
578
579 BEGIN
580     -- (=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)
581     -- IF THE CLASS CODE IS 9046 AND THE TOTAL PREMIUM IS 0, WE ARE NOT TO
582     -- REPORT THE CLASS CODE
583     -- (=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)=(=)
584     IF STATE_NUMBER = '31' THEN
585         IF (CLASS_CODE in ('9046','0063') AND TTL_PREM = 0) THEN
586             NULL;
587         ELSIF CLASS_CODE IN ('0932', '9749') THEN
588             NULL;
589         ELSE
```

```
590                 TEMP_INSERT_UNEX(UNEX_ID,
591                                 UNPL_ID,
592                                 RPT_DATE,
593                                 STATE_NUMBER,
594                                 CLASS_CODE,
595                                 MODVAL,
596                                 MODEFF,
597                                 RATE_EFF,
598                                 PAYROLL,
599                                 NETPREM,
600                                 MODPREM,
601                                 SPLIT,
602                                 THE_RATE,
603                                 USE_MOD,
604                                 xmit,
605                                 WS_SQLCODE);
606                 COMMIT;
607             END IF;
608         ELSIF STATE_NUMBER = '32' THEN
609             IF (TTL_EXPO = 0 AND USE_MOD = 'E' AND ANNIV IS NULL) THEN
610                 NULL;
611             ELSE
612                 TEMP_INSERT_UNEX(UNEX_ID,
613                                 UNPL_ID,
614                                 RPT_DATE,
615                                 STATE_NUMBER,
616                                 CLASS_CODE,
617                                 MODVAL,
618                                 MODEFF,
619                                 RATE_EFF,
620                                 PAYROLL,
621                                 NETPREM,
622                                 MODPREM,
623                                 SPLIT,
624                                 THE_RATE,
625                                 USE_MOD,
626                                 xmit,
627                                 WS_SQLCODE);
628                 COMMIT;
629             END IF;
630         ELSE --STATE_NUMBER NOT IN ('31','32','04') THEN
631                 TEMP_INSERT_UNEX(UNEX_ID,
632                                 UNPL_ID,
633                                 RPT_DATE,
634                                 STATE_NUMBER,
635                                 CLASS_CODE,
636                                 MODVAL,
637                                 MODEFF,
638                                 RATE_EFF,
639                                 PAYROLL,
640                                 NETPREM,
641                                 MODPREM,
642                                 SPLIT,
643                                 THE_RATE,
644                                 USE_MOD,
645                                 xmit,
```

```
                                WS_SQLCODE);
                    COMMIT;
        END IF;
    EXCEPTION
      WHEN OTHERS THEN
          WS_ERR_CODE := sqlcode;
          ws_err_msg := 'INSERT_EXPOSURES';
       P_SEND_MAIL ('MAIL.FLORISTSMUTUAL.COM',
              'FLORISTSMUTUAL.COM',
              'WCUNEX_PACK_BODY',
              'NO_REPLY@HORTICA-INSURANCE.COM',
              'CRAIG, JAMES' ,
              'JCRAIG@HORTICA-INSURANCE.COM',
              'PROCESS FAILED! '||WS_ERR_CODE,
              WS_ERR_MSG||' FAILED: '||UNPL_ID||' SQLCODE: '||WS_ERR_CODE,
              WS_ERR_CODE);


       --RAISE;
  END INSERT_EXPOSURES;
  --
  --##########################################################################
  ########
  --
  --  NAME:       RESET_PERIOD_SPLIT
  --  AUTHOR:
  --  DATE:
  --  PURPOSE:
  --
  --  INPUTS:
  --
  --  OUTPUTS:
  --
  --  MODIFICATION HISTORY:
  --  PROJECT NUMBER  DATE           INITIALS    COMMENT
  --  --------------  -----------     --------    ---------------------------------------
  --  PLI32502.12477   11-JUN-2009    JPC         INITIAL CREATION
  --
  --
  ##########################################################################
  ######

  PROCEDURE RESET_PERIOD_SPLIT
        (
              CLASS_CODE CHAR,
              POLYNO     CHAR,
              EXPOEFF DATE,
              WS_SQLCODE      IN OUT NUMBER
        )
  IS
        PERIOD NUMBER;


  BEGIN


```

```
699         IF CLASS_CODE NOT IN
700             (
701                 '0900', '0063', '9740', '9741'
702             )
703             THEN
704             PERIOD := 0;
705         END IF;
706         --
707         TEMP_UPDATE_TUNX
708         (
709             PERIOD, POLYNO, EXPOEFF,WS_SQLCODE
710         )
711         ;
712         COMMIT;
713
714
715
716
717     EXCEPTION
718       WHEN OTHERS THEN
719           ws_err_msg := 'RESET_PERIOD_SPLIT';
720           WS_ERR_CODE := sqlcode;
721
722
723           dbms_output.put_line(WS_ERR_MSG||' '||WS_ERR_CODE);
724
725
726        P_SEND_MAIL ('MAIL.FLORISTSMUTUAL.COM',
727             'FLORISTSMUTUAL.COM',
728             'WCUNEX_PACK_BODY',
729             'NO_REPLY@HORTICA-INSURANCE.COM',
730             'CRAIG, JAMES' ,
731             'JCRAIG@HORTICA-INSURANCE.COM',
732             'PROCESS FAILED! '||WS_ERR_CODE,
733             WS_ERR_MSG||' FAILED: '||POLYNO,
734             WS_ERR_CODE);
735       --  RAISE;
736   END RESET_PERIOD_SPLIT;
737   --
738   --##############################################################################
      ########
739   --
740   --  NAME:       GET_EXPO_XMIT
741   --  AUTHOR:     J. CRAIG
742   --  DATE:       9 DEC 2010
743   --  PURPOSE:    FLAG EXPOSURES TO SEND/NOT SEND ON US REPORT
744   --
745   --  INPUTS:  CLASS CODE, TOTAL PREMIUM
746   --
747   --  OUTPUTS: TRANSMIT INDICATOR
748   --
749   --  MODIFICATION HISTORY:
750   --  PROJECT NUMBER  DATE           INITIALS    COMMENT
751   --  --------------  -----------     --------    ---------------------------------------
752   --  PLI32502.19870  09-DEC-2010     JPC         INITIAL CREATION
753   --  PLI32502.20830  14-APR-2011     JPC         Modify code assignemnts
```

```
-- PLI32502.21661  15-DEC-2011    JPC       Add Code 0988
-- PLI32502.24619  11-APR-2014    JPC       Remove 9807; alter xmit criteria
--
##########################################################################################
FUNCTION GET_EXPO_XMIT
      (
            V_STAT_CODE CHAR, V_TOT_PX NUMBER, V_STATE CHAR
      )
      RETURN CHAR
IS
      WS_XMIT_IND VARCHAR2(1) := 'X';
BEGIN
      IF V_STAT_CODE IN
            ('9887','9889','9722','9724','9139','9108','0990','0930','0063','0988') THEN
            WS_XMIT_IND := 'N';
      END IF;
      --
      IF V_STAT_CODE IN
            ('0035', '8036', '8742', '8810', '9037', '9740') THEN
            WS_XMIT_IND := 'Y';
      END IF;
      --
      IF V_STAT_CODE IN ('0990','9887','9889','9722','9724','0063', '0032') THEN
            IF V_TOT_PX        <> 0 THEN
                  WS_XMIT_IND := 'Y';
            ELSE
                  WS_XMIT_IND := 'N';
            END IF;
      END IF;
      --
      IF V_STAT_CODE = '0930' then
        IF V_STATE  IN ('04') THEN
             WS_XMIT_IND := 'Y';
        END IF;
      END IF;
      --

      IF V_STAT_CODE = '9807' then
        IF V_STATE  IN ('36') THEN
             WS_XMIT_IND := 'N';
        END IF;
      END IF;

      --
      IF WS_XMIT_IND      = 'X' THEN
            WS_XMIT_IND := 'Y';
      END IF;
      --
      RETURN WS_XMIT_IND;

      --
EXCEPTION
    WHEN OTHERS THEN
        ws_err_msg := 'GET_EXPO_XMIT';
        WS_ERR_CODE := sqlcode;
```

```
808
809
810         dbms_output.put_line(WS_ERR_MSG||' '||WS_ERR_CODE);
811
812
813      P_SEND_MAIL ('MAIL.FLORISTSMUTUAL.COM',
814            'FLORISTSMUTUAL.COM',
815            'WCUNEX_PACK_BODY',
816            'NO_REPLY@HORTICA-INSURANCE.COM',
817            'CRAIG, JAMES' ,
818            'JCRAIG@HORTICA-INSURANCE.COM',
819            'PROCESS FAILED! '||WS_ERR_CODE,
820            WS_ERR_MSG||' FAILED!',
821            WS_ERR_CODE);
822       --
823  END GET_EXPO_XMIT;
824  --##############################################################################
     ########
825  --
826  --  NAME:       Determine_Split_Period
827  --
828  --  MODIFICATION HISTORY:
829  --  PROJECT NUMBER  DATE           INITIALS    COMMENT
830  --  --------------  -----------     --------    ---------------------------------------
831  --  PLI32502.21210   11-JUN-2011    JPC         INITIAL CREATION
832  --  PLI32502.24598   22-JAN-2013    JPC         ADDITION OF ANNIV DATE
833  --  PLI32502.27508   25-APR-2014    JPC         Add State exclusions
834  --  PLI32502.29821   01-JUL-2015    JPC         Change logic to use a cursor
835  --
     ##############################################################################
     ######
836       FUNCTION DETERMINE_SPLIT_PERIOD (
837                 V_CLASS    CHAR,
838                 V_POL_ID   NUMBER,
839                 V_SPLIT_NO NUMBER,
840                 V_STATE CHAR)
841            RETURN DATE
842       IS
843            SPLIT_DATE DATE;
844            FIRST_DATE DATE;
845            MID_DATE   DATE;
846            LAST_DATE DATE;
847
848            cursor date_find is
849            SELECT MIN (TUNX.EXPOSURE_EFFECTIVE) as fdate,
850                 nvl(TUNX.ANNIV_DTE,'31-DEC-3000') as mdate,
851                 MAX (TUNX.EXPOSURE_EFFECTIVE) as ldate
852            FROM   TEMP_UNIT_EXPO TUNX
853            WHERE  TUNX.POLICY_NUMBER = V_POL_ID
854                 AND TUNX.CLASS_CODE = V_CLASS
855            GROUP BY TUNX.ANNIV_DTE;
856          BEGIN
857
858          for df in date_find
859          loop
860
```

```
                FIRST_DATE := df.fdate;
                MID_DATE   := df.mdate;
                LAST_DATE  := df.ldate;

            --
           IF V_STATE not in ('01','10','12','17','18','22','30','42') then
            IF V_SPLIT_NO       = 0 THEN
              IF TO_CHAR(MID_DATE,'YYYYMMDD') < '30001231' then
              if V_CLASS in ('9740','9741') THEN
                SPLIT_DATE := LAST_DATE;
                else
                SPLIT_DATE := MID_DATE;
                end if;
              end if;
            ELSE
                SPLIT_DATE := LAST_DATE;
            END IF;
           ELSE

              IF V_SPLIT_NO       = 0 THEN
                   SPLIT_DATE := FIRST_DATE;

              ELSE
                   SPLIT_DATE := LAST_DATE;
                   DBMS_OUTPUT.PUT_LINE('DETERMINE_SPLIT_PERIOD '||V_CLASS);
              END IF;
            END IF;
            --
            RETURN SPLIT_DATE;
end loop;
            --
EXCEPTION
   WHEN OTHERS THEN
       ws_err_msg := 'DETERMINE_SPLIT_PERIOD';
       WS_ERR_CODE := sqlcode;


       dbms_output.put_line(WS_ERR_MSG||' '||WS_ERR_CODE);


    P_SEND_MAIL ('MAIL.FLORISTSMUTUAL.COM',
            'FLORISTSMUTUAL.COM',
            'WCUNEX_PACK_BODY',
            'NO_REPLY@HORTICA-INSURANCE.COM',
            'CRAIG, JAMES' ,
            'JCRAIG@HORTICA.COM',
            'PROCESS FAILED! '||WS_ERR_CODE,
            WS_ERR_MSG||' FAILED!',
            WS_ERR_CODE);
      --
      END DETERMINE_SPLIT_PERIOD;
      --



--#############################################################################
```

```
    ########
917 --
918 --  NAME: Get_Deviated
919 --
920 --  MODIFICATION HISTORY:
921 --  PROJECT NUMBER  DATE           INITIALS    COMMENT
922 --  --------------  -----------    --------    ---------------------------------------
923 --  PLI32502.28448   15-OCT-2014    JPC         Fix DevPx for Mass.
924 --
    ##########################################################################################
    ######
925
926 PROCEDURE GET_DEVIATED(
927    POLICYID  IN NUMBER,
928    POLYEFF   IN DATE,
929    POLICYNUM IN VARCHAR2,
930    STATENUM  IN VARCHAR2,
931    DATEIN    IN DATE)
932 IS
933   WS_POL TUNX.POLICY_NUMBER%TYPE;
934   WS_RATE TUNX.MULTIPLIER%TYPE;
935   WS_DEVMOD TUNX.MODIFIER_VALUE%TYPE;
936   WS_DEVPX TUNX.NET_PREMIUM%TYPE;
937   WS_STDPX TUNX.STANDARD_PREMIUM%TYPE;
938   WS_CLASS TUNX.CLASS_CODE%TYPE;
939   WS_SECTION VARCHAR2(12);
940   ws_sqlcode NUMBER;
941   WS_UNPL_ID UNEX.UNEX_UNPL_WC_NO%TYPE;
942   WS_POLICY TUNX.POLICY_NUMBER%TYPE;
943   V_POLICY TUNX.POLICY_NUMBER%TYPE;
944   WS_HOLD_CLASS UNEX.UNEX_WC_EXIT_CLASS_CODE%TYPE;
945   WS_NETPX UNEX.UNEX_WC_EXIT_NET_PREM%TYPE;
946   WS_STAND TUNX.STANDARD_PREMIUM%TYPE;
947   WS_MODPX UNEX.UNEX_WC_EXIT_MOD_NET_PREM%TYPE;
948   WS_PAYROLL TUNX.PAYROLL_AMOUNT%TYPE;
949   WS_STATE_NO UNEX.UNEX_STAT_GB_STATE_NO%TYPE;
950   WS_UNEX_WC_NO UNEX.UNEX_WC_NO%TYPE;
951   WS_SPLITNO UNEX.UNEX_WC_SPLIT_PERIOD_IND%TYPE;
952   WS_AFTERMOD UNEX.UNEX_WC_RECORD_TYPE%TYPE;
953   WS_MODVAL UNEX.UNEX_WC_EXP_MOD%TYPE;
954   WS_MODEFF UNEX.UNEX_WC_EXP_MOD_EFF_DATE%TYPE;
955   WS_BURT UNEX.UNEX_WC_BURT_RATE%TYPE;
956   WS_BURT_EFF UNEX.UNEX_WC_BURT_EFF_DATE%TYPE;
957   WS_REPORTDATE DATE;
958   WS_XMIT unex.UNEX_XMIT_IND%type;
959   --
960   CURSOR DEVIATED_PREM
961   IS
962     SELECT DISTINCT TUNX.POLICY_NUMBER AS DPOL_NUM ,
963 --      NVL (TUNX.MULTIPLIER, 0)        AS RATE,
964       NVL(TUNX.MODIFIER_VALUE,0)       AS DEVIATION,
965       SUM (NET_PREMIUM)                AS NETSTNDPX,
966       SUM (TUNX.STANDARD_PREMIUM)      AS STDPX,
967       SUM (TUNX.STANDARD_PREMIUM)-SUM (NET_PREMIUM) as devpx
968 --      CLASS_CODE                     AS CLSCDE
969     FROM TEMP_UNIT_EXPO TUNX
```

```
970      WHERE SUBSTR (CLASS_CODE, 1, 4) IN
971       ( SELECT DISTINCT SUBSTR (CLASS_CODE, 1, 4) AS CLASSCODE
972       FROM STCL
973       WHERE USE_MOD_IND = 'Y'
974       AND CAT_CODE     IN ('MANUAL', 'MODFAC', 'INCLIM', 'MERIT')
975       )
976    AND SUBSTR(CLASS_CODE, 1, 4) IN
977     (SELECT RADE.CLASS_CODE FROM RADE WHERE NVL(DEVIATION_PCT,1) IS NOT NULL
978     )
979    AND SUBSTR (CLASS_CODE, 1, 4) NOT IN ('9807', '9812','6199')
980    AND TUNX.POLICY_NUMBER            = POLICYNUM
981    AND TUNX.STATE_NUMBER             = STATENUM
982    GROUP BY TUNX.POLICY_NUMBER ,
983  --    NVL (TUNX.MULTIPLIER, 0) ,
984      NVL(TUNX.MODIFIER_VALUE,0)
985  --    CLASS_CODE
986    ORDER BY TUNX.POLICY_NUMBER ;
987    -- *********************************************************************
988  BEGIN
989    -- *********************************************************************
990    WS_SECTION := 'WCUNEX2_DV';
991    IF STATENUM = '20' THEN -- INSERT DATA FOR CLASS CODE 9037
992    -- --- -- -   SELECT SUM (NET_PREMIUM)                    AS NETSTNDPX,
993    -- --- -- -     SUM (TUNX.STANDARD_PREMIUM)                AS STNDPX,
994    -- --- -- -     SUM (TUNX.STANDARD_PREMIUM)-SUM (NET_PREMIUM) AS DEVIATION,
995    -- --- -- -     NVL (TUNX.MODIFIER_VALUE, 0)               AS RATE
996    -- --- -- -   INTO WS_NETPX,
997    -- --- -- -        WS_STDPX,
998    -- --- -- -        WS_DEVPX,
999    -- --- -- -        WS_DEVMOD
1000   -- --- -- -   FROM TEMP_UNIT_EXPO TUNX
1001   -- --- -- -   WHERE SUBSTR (CLASS_CODE, 1, 4) IN
1002   -- --- -- -     ( SELECT DISTINCT SUBSTR (CLASS_CODE, 1, 4) AS CLASSCODE
1003   -- --- -- -     FROM STCL
1004   -- --- -- -     WHERE USE_MOD_IND = 'Y'
1005   -- --- -- -     AND CAT_CODE     IN ('MANUAL', 'MODFAC', 'INCLIM', 'MERIT')
1006   -- --- -- -     )
1007   -- --- -- -   AND SUBSTR(CLASS_CODE, 1, 4) IN
1008   -- --- -- -     (SELECT RADE.CLASS_CODE FROM RADE WHERE NVL(DEVIATION_PCT,1) IS
       NOT NULL
1009   -- --- -- -     )
1010   -- --- -- -   AND SUBSTR (CLASS_CODE, 1, 4) NOT IN ('9807', '9812','6199')
1011   -- --- -- -   AND TUNX.POLICY_NUMBER            = POLICYNUM
1012   -- --- -- -   AND TUNX.STATE_NUMBER             = STATENUM
1013   -- --- -- -   GROUP BY NVL (TUNX.MODIFIER_VALUE, 0)
1014   -- --- -- -   ORDER BY TUNX.POLICY_NUMBER ;
1015
1016   for aa in DEVIATED_PREM
1017   loop
1018               WS_NETPX := aa.NETSTNDPX;
1019               WS_STDPX := aa.STDPX    ;
1020               WS_DEVPX := aa.devpx;
1021               WS_DEVMOD:= aa.DEVIATION    ;
1022
1023
1024
```

```
1025      --
1026      WS_SECTION := 'DEVIATE_04';
1027      IF WS_CLASS IS NOT NULL THEN
1028        DBMS_OUTPUT.PUT_LINE ('9037 POLICY '||POLICYNUM||' INSERT REPORT THIS:
          '||WS_DEVPX) ;
1029        DBMS_OUTPUT.PUT_LINE ('******');
1030      END IF;
1031      --
1032      SELECT UNIT_STATS_EXPOSURE_SEQUENCE.NEXTVAL INTO WS_UNEX_WC_NO
          FROM DUAL;
1033      WS_CLASS      := '9037';
1034      WS_MODPX      := WS_DEVPX;
1035      WS_PAYROLL    := 0;
1036      WS_SPLITNO    := 0;
1037      WS_MODVAL     := WS_DEVMOD;
1038      WS_BURT       := 0;
1039      WS_MODEFF     := POLYEFF;
1040      WS_BURT_EFF   := POLYEFF;
1041      WS_AFTERMOD   := 'N';
1042      WS_XMIT       := 'Y';
1043      WS_STATE_NO   := STATENUM;
1044      WS_UNPL_ID    := POLICYID;
1045      WS_REPORTDATE := DATEIN;
1046      --
1047      WS_SECTION := 'WCUNEX2_09';
1048      WCX.TEMP_INSERT_UNEX (WS_UNEX_WC_NO, WS_UNPL_ID, WS_REPORTDATE,
          WS_STATE_NO, WS_CLASS, WS_MODVAL, WS_MODEFF, WS_BURT_EFF,
          WS_PAYROLL, WS_DEVPX, WS_MODPX, WS_SPLITNO, WS_BURT, WS_AFTERMOD,
          WS_XMIT, WS_SQLCODE) ;
1049      COMMIT;
1050    end loop;
1051      --
1052    END IF;
1053    EXCEPTION
1054    WHEN TOO_MANY_ROWS THEN
1055      DBMS_OUTPUT.PUT_LINE ('9037TMR POLICY: '||POLICYNUM||' CODE: '||SQLCODE) ;
1056    WHEN NO_DATA_FOUND THEN
1057      DBMS_OUTPUT.PUT_LINE ('9037NDF POLICY: '||POLICYNUM||' CODE: '||SQLCODE) ;
1058    WHEN OTHERS THEN
1059      DBMS_OUTPUT.PUT_LINE ('9037OTH POLICY: '||POLICYNUM||' CODE: '||SQLCODE) ;
1060    END GET_DEVIATED;
1061
1062    --
      ##########################################################################
      ######
1063
1064
1065    END PK_WCUNEX_PACK;
1066    /
1067    SHO ERR
1068    DROP   SYNONYM WCX;
1069
1070    -- use this for testing
1071    --CREATE OR REPLACE SYNONYM WCX FOR PK_WCUNEX_PACK;
1072
1073    -- use this for production
```

```
1074 CREATE OR REPLACE PUBLIC SYNONYM WCX FOR PK_WCUNEX_PACK;
1075
1076 GRANT EXECUTE ON PK_WCUNEX_PACK TO PUBLIC;
```