```
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;

```

```sql
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;
/
```