

```

CONN &&apps_login/&&apps_passwd&&connect_string
PROMPT CREATING PACKAGE XXDHI_RHP_AR_RECEIPTS_TFR_API
CREATE OR REPLACE PACKAGE xxdhi_rhp_ar_receipts_tfr_api
/* $Header: XXXXXXXX.pkh, Version 1.0, 07-FEB-2003      UKEXPI  $
*****
*
*                               History Log
*
*****
* App/Release   : Oracle e-Business Suite RXXX
* Oracle App    : SHORTNAME - FULL_NAME
* Module        : SHORTNAME - DESCRIPTION
* File          : XXDHI_RHP_AR_RECEIPTS_00.sql
* Author        : Anil Passi, xxClientxx  Company, Program xxProjectxx
* Version       : 1.0
* Created       : 10-FEB-2003
* Description    : This package gives API which RHP interface calls
* This API will be used for the following
* Create RHP Receipt, Unidentified
* Create RHP Receipt, Identified but Unapplied
* Reverse unapplied receipts
*
* Change History Log
* =====
*
* Version Date      Author      Change
* =====
* 1.0      10-FEB-2003  Anil Passi  Initial Version
*
* Parameters      :
*
*****/
IS
PROCEDURE process_rhp_receipts (
  p_rhp_receipt_id      IN      INTEGER
, p_rhp_receipt_routing_id IN  INTEGER
, p_perform_reversal    IN      VARCHAR2 DEFAULT 'N'
, p_success_flag OUT BOOLEAN
);
END xxdhi_rhp_ar_receipts_tfr_api;
/

SHOW errors package XXDHI_RHP_AR_RECEIPTS_TFR_API ;

```

```

CONN &&apps_login/&&apps_passwd&&connect_string
PROMPT CREATING PACKAGE BODY XXDHI_RHP_AR_RECEIPTS_TFR_API
CREATE OR REPLACE PACKAGE BODY xxdhi_rhp_ar_receipts_tfr_api
/* $Header: XXXXXXXX.pkh, Version 1.0, 07-FEB-2003      UKEXPI  $
*****
*
*                               History Log
*
*****
* App/Release   : Oracle e-Business Suite RXXX
* Oracle App    : SHORTNAME - FULL_NAME
* Module        : SHORTNAME - DESCRIPTION
* File          : XXDHI_RHP_AR_RECEIPTS_00.sql
* Author        : Anil Passi, xxClientxx  Company, Program xxProjectxx
* Version       : 1.0
* Created       : 10-FEB-2003
* Description    : This package gives API which RHP interface calls
* This API will be used for the following
* Create RHP Receipt, Unidentified
*
*****

```

```

* Create RHP Receipt, Identified but Unapplied
* Reverse unapplied receipts
*
* Change History Log
* =====
*
* Version Date      Author      Change
* =====
* 1.0      10-FEB-2003  A. Passi  Initial Version
*
* Parameters      :
*
*****/
IS
--      g_receipt_method_name  VARCHAR2 ( 30 )      := 'SL test payment method';
--      g_receipt_method_name  CONSTANT VARCHAR2 ( 30 )      := 'RHP Method';

g_process_name          CONSTANT VARCHAR2 ( 30 )      := 'RHP';
g_api_version           CONSTANT NUMBER              := 1.0;

g_reversal_category     CONSTANT VARCHAR2 ( 30 )      := 'REV';
g_reversal_reason_code  CONSTANT VARCHAR2 ( 30 )      := 'PAYMENT REVERSAL';
g_reversal_comments     CONSTANT VARCHAR2 ( 30 )      := 'Reversed from RHP';

/*
Since this API is always called in the context of a specific
receipt, there is no harm in making the records privately global
to this package. It can be argued this being not the best
programming practice, but Oracle itself defines various global
record type variables so as to avoid passing the
record values from one method to another. In this
case performance and simplicity wins over best practice.
*/
CURSOR c_rhp_receipt (
  p_rhp_receipt_id      IN      INTEGER
)
IS
SELECT *
FROM xxdhi_rhp_receipts
WHERE receipt_id = p_rhp_receipt_id;

p_rhp_receipt          c_rhp_receipt%ROWTYPE;

CURSOR c_rhp_receipt_routing (
  p_rhp_receipt_id      IN      INTEGER
, p_rhp_receipt_routing_id IN  INTEGER
)
IS
SELECT *
FROM xxdhi_rhp_receipt_routings
WHERE receipt_id = p_rhp_receipt_id
AND receipt_routing_id = p_rhp_receipt_routing_id;

p_rhp_receipt_routing  c_rhp_receipt_routing%ROWTYPE;

/*-----BEGIN PRIVATE METHODS HERE-----*/
PROCEDURE debug (
  p_debug_level          IN      INTEGER DEFAULT 5

```

```

),p_debug_msg          IN          VARCHAR2
)
IS
BEGIN
  IF p_debug_level = 99 THEN
    RETURN;
  END IF;

  XXDHI_DEBUG_API.ADD ( 'DEBUG ' || p_debug_msg );

END debug;

FUNCTION validate_currency
  RETURN BOOLEAN
IS
BEGIN
  RETURN xxdhi_util_pkg.validate_currency (
    p_currency_code => p_rhp_receipt.originating_currency
    ,p_accounting_date => p_rhp_receiptreceipt_date
  );
END validate_currency;

FUNCTION does_exchange_rate_exsit (
  p_func_curr          OUT          VARCHAR2
,p_org_id              OUT          INTEGER
,p_foriegn_currency   OUT          BOOLEAN
)
  RETURN BOOLEAN
IS
  v_org_id             INTEGER;
  v_func_curr          fnd_currencies.currency_code&TYPE;
BEGIN
  v_org_id              :=
    xxdhi_util_pkg.get_org_id_from_bank_account (
      p_bank_account_id => p_rhp_receipt.bank_account_id
    );
  v_func_curr           :=
    xxdhi_util_pkg.get_sob_curr_from_org_ap_sys ( p_org_id => v_org_id );

  IF NVL ( v_func_curr, 'XX' ) !=
    NVL ( p_rhp_receipt.originating_currency, 'XX' ) THEN
p_foriegn_currency := TRUE ;
  debug( p_debug_msg=>'p_foriegn_currency := TRUE ;' );
  IF 'N' =
    gl_currency_api.rate_exists (
      x_from_currency => p_rhp_receipt.originating_currency
      ,x_to_currency   => v_func_curr
      ,x_conversion_date => p_rhp_receiptreceipt_date
      ,x_conversion_type => xxdhi_util_pkg.g_conversion_type_code
    ) THEN
    /* func curr needs to be send back, as its needed in message*/
    p_func_curr       := v_func_curr;
    p_org_id          := v_org_id;
    RETURN FALSE;
  END IF;
ELSE
  p_foriegn_currency := FALSE ;
  debug( p_debug_msg=>'p_foriegn_currency := FALSE ;' );
END IF;

p_func_curr          := v_func_curr;
p_org_id              := v_org_id;
RETURN TRUE;

```

```

END does_exchange_rate_exsit;

FUNCTION validate_accounting_date (
  p_org_id              IN          INTEGER
)
  RETURN BOOLEAN
IS
  n_sob_id              INTEGER;
BEGIN
  n_sob_id              :=
    xxdhi_util_pkg.get_sob_from_org_id ( p_org_id => p_org_id );
  RETURN xxdhi_util_pkg.validate_gl_accounting_date (
    p_accounting_date => p_rhp_receiptreceipt_date
    ,p_sob_id          => n_sob_id
  );
END validate_accounting_date;

/* This procedure will set just the first message from global pl/sql table
into the message stack.
The RHP screen will have to resolve this error message first and foremost */

PROCEDURE set_first_message_on_stack
IS
  v_msg_string          VARCHAR2 ( 2000 );
BEGIN
  FOR i IN 1 .. fnd_msg_pub.count_msg
  LOOP
    v_msg_string         :=
      fnd_msg_pub.get ( p_msg_index => i, p_encoded => fnd_api.g_false );
    fnd_message.set_name ( 'AR', 'GENERIC_MESSAGE' );
    fnd_message.set_token ( 'GENERIC_TEXT', v_msg_string );
    RETURN;
  END LOOP;
END set_first_message_on_stack;

/*
Just read from the global variable of type record and
create a new receipt using the API
*/
FUNCTION create_new_receipt
( p_foriegn_currency IN BOOLEAN )
  RETURN BOOLEAN
IS
  v_msg_count           INTEGER;
  v_msg_data            VARCHAR2 ( 2000 );
  v_cr_id               INTEGER;
  v_return_status       VARCHAR2 ( 1 );
BEGIN
  IF p_foriegn_currency THEN
    ar_receipt_api_pub.create_cash (
      p_api_version      => g_api_version
      ,p_init_msg_list   => fnd_api.g_true
      ,p_commit          => fnd_api.g_false
      ,p_validation_level => fnd_api.g_valid_level_full
      ,x_return_status   => v_return_status
      ,x_msg_count       => v_msg_count
      ,x_msg_data        => v_msg_data
      ,p_currency_code   => p_rhp_receipt.originating_currency
      ,p_amount          => p_rhp_receiptrouting.routed_amount
      ,p_receipt_number  => g_process_name || '-' ||
        p_rhp_receiptreceipt_id ||
        '-' ||

```

```

        p_rhp_receipt_routingreceipt_routing_id
,p_receipt_date => p_rhp_receiptreceipt_date
,p_cr_id => v_cr_id
,p_receipt_method_name => g_receipt_method_name
,p_customer_number => p_rhp_receipt_routing.ar_customer_number
,p_comments => p_rhp_receipt.originating_customer
,p_customer_receipt_reference => SUBSTR(p_rhp_receipt.sender_to_receiver_info,1,30)
,p_remittance_bank_account_id => p_rhp_receipt.bank_account_id
,p_exchange_rate_type => xxdhi_util_pkg.g_conversion_type_code
,p_exchange_rate_date => p_rhp_receiptreceipt_date
);
ELSE
ar_receipt_api_pub.create_cash (
    p_api_version => g_api_version
,p_init_msg_list => fnd_api.g_true
,p_commit => fnd_api.g_false
,p_validation_level => fnd_api.g_valid_level_full
,x_return_status => v_return_status
,x_msg_count => v_msg_count
,x_msg_data => v_msg_data
,p_currency_code => p_rhp_receipt.originating_currency
,p_amount => p_rhp_receipt_routing.routed_amount
,p_receipt_number => g_process_name || '-' ||
    p_rhp_receiptreceipt_id ||
    '-' ||
    p_rhp_receipt_routingreceipt_routing_id
,p_receipt_date => p_rhp_receiptreceipt_date
,p_cr_id => v_cr_id
,p_receipt_method_name => g_receipt_method_name
,p_customer_number => p_rhp_receipt_routing.ar_customer_number
,p_comments => p_rhp_receipt.originating_customer
,p_customer_receipt_reference => SUBSTR(p_rhp_receipt.sender_to_receiver_info,1,30)
,p_remittance_bank_account_id => p_rhp_receipt.bank_account_id
);
END IF;

IF v_return_status = 'E' THEN
    set_first_message_on_stack;
    RETURN FALSE;
END IF;

RETURN TRUE;
END create_new_receipt;

/*
Just read from the global variable of type record and
reverse existing receipt using the API
*/
FUNCTION reverse_existing_receipt
(
    p_cash_receipt_id IN INTEGER
)
RETURN BOOLEAN
IS
    v_msg_count INTEGER;
    v_msg_data VARCHAR2 ( 2000 );
    v_cr_id INTEGER;
    v_return_status VARCHAR2 ( 1 );
BEGIN
    debug( p_debug_msg=>'Now invoking the receipt reversal for '
        || 'p_cash_receipt_id=>' || p_cash_receipt_id
        );
    ar_receipt_api_pub.reverse (
        p_api_version => g_api_version
,p_init_msg_list => fnd_api.g_true
,p_commit => fnd_api.g_false
,p_validation_level => fnd_api.g_valid_level_full

```

```

        ,x_return_status => v_return_status
        ,x_msg_count => v_msg_count
        ,x_msg_data => v_msg_data
    ,p_cash_receipt_id => p_cash_receipt_id
    ,p_reversal_category_code => g_reversal_category
    ,p_reversal_gl_date => TRUNC(SYSDATE)
    ,p_reversal_date => TRUNC(SYSDATE)
    ,p_reversal_reason_code => g_reversal_reason_code
    ,p_reversal_comments => g_reversal_comments
);

IF v_return_status = 'E' THEN
    set_first_message_on_stack;
    RETURN FALSE;
END IF;

RETURN TRUE;
END reverse_existing_receipt ;

/*
The standard API does not give a very user friendly error message when
payment method association with bank account does not exist.
Hence we double check if the error is caused for this reason.
If so, then we replace the top message on error stack with a more
user friendly message.
*/
PROCEDURE pay_meth_not_defined_bank_acct
IS
    CURSOR c_check
    IS
        SELECT 'x'
        FROM ar_receipt_methods rm
            ,ap_bank_accounts ba
            ,ar_receipt_method_accounts rma
            ,ar_receipt_classes rc
        WHERE rm.name = g_receipt_method_name
            AND rma.bank_account_id = p_rhp_receipt.bank_account_id
            AND ba.bank_account_id = p_rhp_receipt.bank_account_id
            AND (
                p_rhp_receiptreceipt_date BETWEEN rm.start_date
                AND NVL ( rm.end_date, p_rhp_receiptreceipt_date )
            )
            AND (
                (
                    rc.creation_method_code =
                    DECODE (
                        'XX'
                    , 'BR_REMITTED', 'BR_REMIT'
                    , 'BR_FACTORED_WITH_RECOURSE', 'BR_REMIT'
                    , 'BR_FACTORED_WITHOUT_RECOURSE', 'BR_REMIT'
                    , '@*%?&'
                    )
                )
                OR ( rc.creation_method_code = 'MANUAL' )
                OR (
                    rc.creation_method_code = 'AUTOMATIC'
                    AND
                    -- rc.remit_flag = 'Y' and
                    -- OSTEINME 2/27/2001: removed remit_flag
                    -- condition for iReceivables CC functionality.
                    -- See bug 1659109.
                    rc.confirm_flag = 'N'
                )
            )
            AND ba.account_type = 'INTERNAL'
            AND NVL ( ba.inactive_date, p_rhp_receiptreceipt_date + 1 ) >
                p_rhp_receiptreceipt_date
            AND p_rhp_receiptreceipt_date BETWEEN rma.start_date
            AND NVL ( rma.end_date, p_rhp_receiptreceipt_date )

```

```

AND ba.currency_code =
  DECODE (
    ba.receipt_multi_currency_flag
    , 'Y', ba.currency_code
    , 'GBP'
  )
AND rc.receipt_class_id = rm.receipt_class_id
AND rm.receipt_method_id = rma.receipt_method_id
AND rma.bank_account_id = ba.bank_account_id
--APANDIT: changes made for the misc receipt creation api.
AND (
  ( NVL ( 'XX', '*&#$$' ) <> 'MISC' )
  OR (
    rm.receipt_class_id NOT IN
    ( SELECT arc.receipt_class_id
      FROM ar_receipt_classes arc
      WHERE arc.notes_receivable = 'Y'
            OR arc.bill_of_exchange_flag = 'Y' )
  )
);

p_check c_check%ROWTYPE;
BEGIN
OPEN c_check;
FETCH c_check INTO p_check;

IF c_check%NOTFOUND THEN
/*
yes the error is because of payment method not
being associated with the bank account
*/
find_message.set_name ( 'AR', 'GENERIC_MESSAGE' );
find_message.set_token (
  'GENERIC_TEXT'
  , 'The payment method =>' || g_receipt_method_name || '<=' || CHR ( 10 ) ||
  'does not have association with the Bank Account in which funds are being
received.' ||
  CHR ( 10 ) ||
  CHR ( 10 ) ||
  'Also ensure that for foreign currency receipt, ' ||
  'the bank account is of type multi currency'
);
END IF;

CLOSE c_check;
END pay_meth_not_defined_bank_acct;

FUNCTION validate_reversal_flag_valid
(p_cash_receipt_id OUT INTEGER )
RETURN BOOLEAN IS
CURSOR c_check IS
SELECT
  cash_receipt_id
  ,status
FROM
  ar_cash_receipts
WHERE
  receipt_number = g_process_name || '-' ||
  p_rhp_receipt_receipt_id ||
  '-' ||
  p_rhp_receipt_routing.receipt_routing_id
;
p_check c_check%ROWTYPE ;
no_existing_receipt EXCEPTION ;
receipt_already_applied EXCEPTION ;
receipt_already_reversed EXCEPTION ;
BEGIN

```

```

OPEN c_check ;
FETCH c_check INTO p_check ;
CLOSE c_check ;

IF p_check.cash_receipt_id IS NULL THEN
  RAISE no_existing_receipt ;
END IF ;

IF p_check.status = 'APP' THEN
  RAISE receipt_already_applied ;
END IF ;

IF p_check.status = 'REV' THEN
  RAISE receipt_already_reversed ;
END IF ;

p_cash_receipt_id := p_check.cash_receipt_id ;
RETURN TRUE ;

EXCEPTION
WHEN no_existing_receipt THEN
  find_message.set_name ( 'AR', 'AR_RAPI_CASH_RCPT_ID_INVALID' );
  RETURN FALSE ;
WHEN receipt_already_applied THEN
  find_message.set_name ( 'AR', 'GENERIC_MESSAGE' );
  find_message.set_token (
    'GENERIC_TEXT'
    , 'This receipt has been applied to transaction in xxProjectxx Oracle Receivables.'
    || chr(10) || 'Please unapply the receipt before reversing'
  );
  RETURN FALSE ;
WHEN receipt_already_reversed THEN
  find_message.set_name ( 'AR', 'GENERIC_MESSAGE' );
  find_message.set_token (
    'GENERIC_TEXT'
    , 'This receipt has already been reversed in xxProjectxx Oracle Receivables.'
  );
  RETURN FALSE ;
END validate_reversal_flag_valid ;

/*
FUNCTION does_receipt_already_exist RETURN BOOLEAN

FUNCTION validate_reversal_flag_valid RETURN BOOLEAN

FUNCTION validate_reversal_if_applied RETURN BOOLEAN
*/

/*-----END PRIVATE METHODS HERE-----*/

/*****
*****
This is the beginning of MAIN code
*****
*****/

PROCEDURE process_rhp_receipts (
  p_rhp_receipt_id IN INTEGER
  ,p_rhp_receipt_routing_id IN INTEGER
  ,p_perform_reversal IN VARCHAR2 DEFAULT 'N'
  ,p_success_flag OUT BOOLEAN
)
IS
  records_do_exist_in_db EXCEPTION;
  invalid_currency_code EXCEPTION;
  conversion_rate_does_not_exist EXCEPTION;
  invalid_conversion_type EXCEPTION;
  not_in_open_period EXCEPTION;

```

```

can_not_find_org_id          EXCEPTION;

v_func_curr                  fnd_currencies.currency_code%TYPE;
v_org_id                     INTEGER;
b_foriegn_currency          BOOLEAN ;
n_cash_receipt_id           INTEGER;

BEGIN

debug ( p_debug_level => 99, p_debug_msg => NULL );
debug ( p_debug_msg=>' ' );
debug ( p_debug_msg=>'xxdhi_util_pkg.init() ' );
xxdhi_util_pkg.init;

IF xxdhi_util_pkg.get_instance = 'DGZTY4' THEN
  g_receipt_method_name := 'SL test payment method' ;
END IF ;

IF xxdhi_util_pkg.get_instance = 'DGZGE5' THEN
  g_receipt_method_name := 'RHP Method' ;
END IF ;

/* open and close the first cursor */
OPEN c_rhp_receipt ( p_rhp_receipt_id );
FETCH c_rhp_receipt INTO p_rhp_receipt;
CLOSE c_rhp_receipt;

/* open and close the second cursor */
OPEN c_rhp_receipt_routing ( p_rhp_receipt_id, p_rhp_receipt_routing_id );
FETCH c_rhp_receipt_routing INTO p_rhp_receipt_routing;
CLOSE c_rhp_receipt_routing;

/* check if the cursor fetch succeeded */
IF p_rhp_receipt_receipt_id IS NULL
OR p_rhp_receipt_routing_receipt_id IS NULL THEN
  RAISE records_do_exist_in_db;
END IF;

IF NOT validate_currency THEN
  debug ( p_debug_msg => 'Invalid currency from xxdhi receipt API' );
  RAISE invalid_currency_code;
END IF;

IF NOT does_exchange_rate_exsit (
  p_func_curr => v_func_curr
, p_org_id => v_org_id
, p_foriegn_currency => b_foriegn_currency
) THEN
debug ( p_debug_msg=> ' IN does_exchange_rate_exsit v_func_curr ' || v_func_curr );
  RAISE conversion_rate_does_not_exist;
END IF;

debug ( p_debug_msg=>'v_org_id is ' || v_org_id );
IF v_org_id IS NULL THEN
  RAISE can_not_find_org_id;
END IF;

/*RHP will not be setting the org_id context. Hence set this up for API*/
debug (
  p_debug_msg => 'Setting org_id to ' || v_org_id ||
  ' xxdhi receipt API'
);
DBMS_APPLICATION_INFO.set_client_info ( v_org_id );

```

```

IF xxdhi_util_pkg.g_conversion_type_code IS NULL THEN
  RAISE invalid_conversion_type;
END IF;

IF NOT validate_accounting_date ( p_org_id => v_org_id ) THEN
  RAISE not_in_open_period;
END IF;

IF p_perform_reversal = 'N' THEN
  /* Call the receipt creation API */
  debug ( p_debug_msg=>
    'p_perform_reversal is N. Hence calling create_new_receipt');
  p_success_flag
  := create_new_receipt ( p_foriegn_currency => b_foriegn_currency );

  IF NOT p_success_flag THEN
    pay_meth_not_defined_bank_acct;
  END IF;

  RETURN;
END IF;

IF p_perform_reversal = 'Y' THEN
  /*
  If the reversal is being performed but
  a. Receipt does not exist
  b. Receipt is already applied to transaction
  then validate_reversal_flag_valid will place a message
  on the stack
  */
  debug (
    p_debug_msg=>
    'Perform reversal is Y.Calling validate_reversal_flag_valid' );
  IF NOT validate_reversal_flag_valid
    ( p_cash_receipt_id => n_cash_receipt_id ) THEN
    debug ( p_debug_msg=>'validate_reversal_flag_valid returned false' );
    p_success_flag := FALSE ;
  RETURN ;
  END IF ;
  debug( p_debug_msg=>'After calling validate_reversal_flag_valid '
  || 'n_cash_receipt_id=>' || n_cash_receipt_id );
  /* This bit will be executed if reversal can be legitimately performed*/
  p_success_flag := reverse_existing_receipt
    (
      p_cash_receipt_id => n_cash_receipt_id
    );
  RETURN;
END IF ;
EXCEPTION
  WHEN records_do_exist_in_db THEN
  debug ( p_debug_msg=> ' EXCEPTION records_do_exist_in_db ' );
  p_success_flag := FALSE;
  fnd_message.set_name ( 'XXDHI', 'NO_DB_RECORD_IN_RHP' );
  fnd_message.set_token ( 'RECEIPT_ID', p_rhp_receipt_id );
  fnd_message.set_token (
    'RECEIPT_ROUTING_ID'
    ,p_rhp_receipt_routing_id
  );
  WHEN invalid_currency_code THEN
  debug( p_debug_msg=> ' EXCEPTION invalid_currency_code ' );
  p_success_flag := FALSE;
  fnd_message.set_name ( 'SQLGL', 'R_PPOS0026' );
  WHEN conversion_rate_does_not_exist THEN
  debug( p_debug_msg=> ' EXCEPTION conversion_rate_does_not_exist ' );

```

```
p_success_flag          := FALSE;
fnd_message.set_name ( 'XTR', 'XTR_2207' );
fnd_message.set_token ( 'CURR1', v_func_curr );
fnd_message.set_token ( 'CURR2', p_rhp_receipt.originating_currency );
fnd_message.set_token (
    'XCHG_DATE'
    ,TO_CHAR ( p_rhp_receiptreceipt_date )
);
fnd_message.set_token ( 'C_TYPE', xxdhi_util_pkg.g_conversion_type );
WHEN invalid_conversion_type THEN
debug( p_debug_msg=> ' EXCEPTION invalid_conversion_type ' );
p_success_flag          := FALSE;
fnd_message.set_name ( 'SQLGL', 'GL_JE_INVALID_CONVERSION_TYPE' );
WHEN not_in_open_period THEN
debug( p_debug_msg=> ' EXCEPTION invalid_conversion_type ' );
p_success_flag          := FALSE;
fnd_message.set_name ( 'SQLGL', 'GL_JE_NOT_OPEN_OR_FUTURE_ENT' );
WHEN can_not_find_org_id THEN
debug( p_debug_msg=> ' EXCEPTION invalid_conversion_type ' );

p_success_flag          := FALSE;
fnd_message.set_name ( 'AR', 'GENERIC_MESSAGE' );
fnd_message.set_token (
    'GENERIC_TEXT'
    ,'Operating Unit can not be found'
);
WHEN OTHERS THEN
debug( p_debug_msg=> ' EXCEPTION OTHERS ' || sqlerrm );
p_success_flag          := FALSE;
fnd_message.set_name ( 'FND', 'FS-UNKNOWN' );
fnd_message.set_token ( 'ERROR', SQLERRM );
END process_rhp_receipts;

END xxdhi_rhp_ar_receipts_tfr_api;
/

SHOW errors package body XXDHI_RHP_AR_RECEIPTS_TFR_API ;
```