

# **XxProjxx Treasury Project**

## **Design Document for Generic Notification Workflow API**

Type of document:  
Design Document

Authors:  
Anil Passi

Valid From:  
22<sup>nd</sup> April 2002

Contact:

XxProjxx Treasury Team

Concerns:

Distributed to:  
XxProjxx Treasury Team  
XxProjxx Treasury Users  
XxProjxx GL Team

## Document Control

### Change Record

Date	Author	Version	Change Reference
22 Apr 02	Anil Passi	Draft 1a	No Previous Document
03-Dec-02	Anil Passi	Draft 1b	Replaced SRX by XXDHI

### Reviewers

Name	Position
	XxClientxx Re Project Manager
	XxClientxx Re Business Analyst
	XxClientxx Re Business Analyst
	XxClientxx Re Functional/Technical Analyst
	Functional/Technical Consultant
	Technical Consultant

### Distribution

	Name	Location
1	Library Master	Project Library
2		Project Manager
3		
4		

#### Note To Holders:

If you receive an electronic copy of this document and print it out, please write your name on the equivalent of the cover page, for document control purposes.

If you receive a hard copy of this document, please write your name on the front cover, for document control purposes.

---

# Contents

- 1 Overview..... 4
  - 1.1 Introduction..... 4
  - 1.2 Workflow API Flow Diagram ..... 5
  - 1.3 API - Send Notification..... 6
  - 1.4 API - Associate Query to Notification..... 8
  - 1.5 Example ( Send a plain text message)..... 9
  - 1.6 Example (Send static HTML message)..... 10
  - 1.7 Example ( Static Message plus Dynamic From Query) ..... 11
  - 1.8 Example ( Static Message plus Dynamic From Multiple Queries ) ..... 12
- 2 Workflow Item Type ..... 13
- 3 Workflow Processes ..... 14
  - 3.1 Workflow API Process ..... 14
- 4 Other Workflow Components ..... 15
  - 4.1 Workflow Attributes ..... 15
  - 4.2 Workflow Functions..... 15
  - 4.3 Notifications and Messages ..... 15
  - 4.4 Workflow Lookups..... 15
- 5 Testing ..... 16
- 6 Technical Details ..... 17
  - 6.1 General ..... 17
  - 6.2 Table access/bespoke table creation ..... 17
  - 6.3 Database Triggers..... 17
  - 6.4 Database Package ..... 17
  - 6.5 Prototype Source Code ..... 18
  - 6.6 System Setup Required..... 27
  - 6.7 Configuration Items ..... 28
  - 6.8 Outline Installation Instructions..... 29
- 7 Open and Closed Issues for this Deliverable ..... 30
  - 7.1 Open Issues ..... 30
  - 7.2 Closed Issues..... 30
- 8 Review Comments..... 31
  - 8.1 Draft 1A..... 31

---

# 1 Overview

---

## 1.1 Introduction

During the analysis phase of program XxProjxx, it was realised that there will be several requirements to develop different notification workflows. In order to minimise the Workflow development effort, it has been decided to develop a generic workflow which can be used by any developer to create notifications. This workflow can be executed by a simple pl/sql call.

The API will send Email/Oracle notification with plain/HTML text to the desired user. The email can be static text or dynamic text(result of query) or a combination of both. The result of the query will be displayed and formatted in HTML tabular format by the API.

### 1.1.1 Advantages

---

- One workflow handles most of the notification tasks
- No need to develop a new workflow for each notification requirement.
- Can be used by all the developers, using a simple pl/sql procedure call
- Workflow expertise not required to generate notifications.

### 1.1.2 Main features

---

- Notification can be a Static Text
- Notification can be a Static HTML Text
- Notification can be an output of a query
- Result of Select Statements displayed nicely in tabular format
- Notification can be an output of more than one SELECT Statements
- Notification can be the combination of Static HTML/Text and SELECT Statement

### 1.1.3 Possible future enhancements to this API

---

Enhancing this API to handle responses to notifications. We need to decide upon a given set of responses which this API should serve. For example "Approve/Reject", "Yes/No", "Action Completed" etc. A new parameter can then be added to the API, this parameter will be called response type. The response type can be APPROVAL, YES\_NO etc.

A callback mechanism to be incorporated when an action is taken on workflow notification.

Note: The new parameter has already been added, but the callback function is not currently implemented. The callback function should be implemented only if there is a strong requirement from technical developers.

### 1.1.4 Limitations

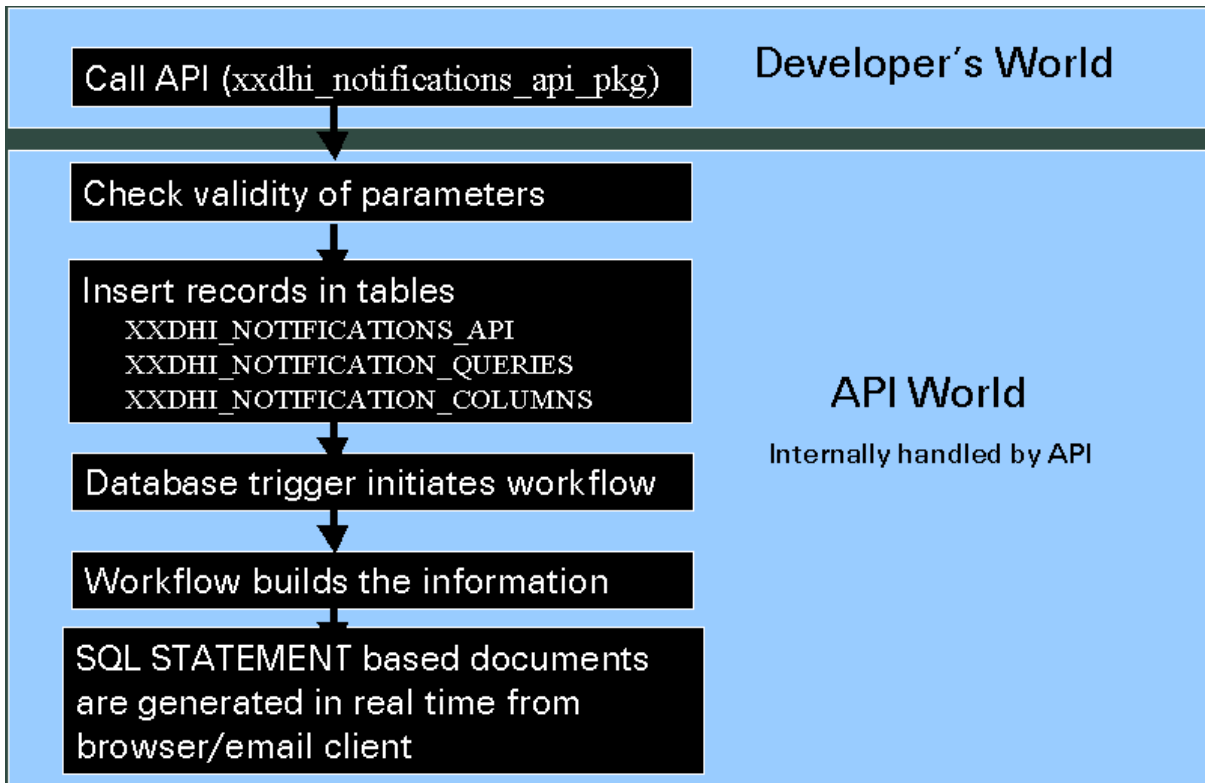
---

The static text displayed will be no more than 4000 characters. The reason being that Oracle allows upto a maximum of 4000 characters in a VARCHAR2 column. Although more text columns can be added to the API so as to multiply this size limit, but this will be done if requirements force us to do so.

## 1.2 Workflow API Flow Diagram

Simple PL/SQL API call will create notification.

All the complexity of creating the notification will be handled within the API.



### 1.3 API - Send Notification

In order to send a notification, XXDHI\_NOTIFICATIONS\_API\_PKG.SEND\_NOTIFICATION should be called.

This package procedure has the following parameters.

#### 1.3.1 X\_EMAIL\_ADDRESS Data Type=IN VARCHAR2, Required=Yes

This is the email address of the person to whom notification must be sent. In Oracle Workflows, notifications are sent to ROLES. A Role can be a user, a supplier contact, responsibility etc. if required, this API will create an Ad hoc role for the email user.

#### 1.3.2 X\_USER\_NAME Data Type=IN VARCHAR2, Required=No

This parameter is optional. The value passed to this parameter will be validated against FND\_USER.USER\_NAME. Where possible, a value should be passed to this parameter.

#### 1.3.3 X\_MESSAGE\_TYPE Data Type=IN VARCHAR2, Required=Yes

This parameter can have one of the following possible values

##### 1.3.3.1 TEXT

The body of the email will be Static Text.

##### 1.3.3.2 HTML

The body of the email will be Static HTML.

##### 1.3.3.3 QUERY

The body of the email will contain the result of a Query displayed nicely in tabular format. A separate package procedure XXDHI\_NOTIFICATIONS\_API\_PKG.ADD\_QUERY must be called to associate a query with the notification.

##### 1.3.3.4 TEXT\_AND\_QUERY


The body of the email will contain a static text(Plain/HTML). The static text message will be followed by the result of a query displayed in tabular format. A separate package procedure XXDHI\_NOTIFICATIONS\_API\_PKG.ADD\_QUERY must be called to associate a query with the notification.

#### 1.3.4 X\_MESSAGE\_SUBJECT Data Type=IN VARCHAR2, Required=Yes

The message subject will be in plain text format. The caller of the API can construct a meaningful text subject. Workflow will display this text as the Email Subject.

#### 1.3.5 X\_PROCESS\_SHORT\_CODE Data Type=IN VARCHAR2, Required=Yes

Process short code should be no more than 4 Characters long. Process code will be concatenated with a unique number to construct an Item Key for each notification. For example, to search for all the processes which belong to process code RIP, RIP% can be entered in the process search window.



The screenshot shows a search interface with three radio buttons at the top: 'Any Status' (selected), 'Active' (with a green flag icon), and 'Complete' (with a checkered flag icon). Below the radio buttons are two input fields: 'Item Type' with a dropdown menu showing 'All', and 'Item Key' with a text box containing 'RIP%'.

**1.3.6 X\_MESSAGE\_TEXT****Data Type=IN VARCHAR2, Required=Yes**

Message text will constitute the body of email message.

This parameter can either be passed plain text or HTML text.

**1.3.7 X\_NOTIFICATION\_API\_ID****Data Type=OUT NUMBER, Required=Yes**

This parameter will return a unique reference number for each notification. The calling process may store this as a reference number against any transaction.

The Workflow User Item Key value, which is a unique reference associated with every workflow process, will be constructed as PROCESSCODE-NOTIFICATION\_API\_ID. For example for process code RIP and notification\_api\_id 1000, the workflow item key will be RIP-1000.

## 1.4 API - Associate Query to Notification

One or more queries can be associated with the notification. The result of the queries will be displayed in a tabular format in the email. Although the static text message of the notification can be constructed dynamically to include data from the tables, but Query based approach minimises the network traffic. Secondly this approach helps to show the data within Oracle on real-time basis. Workflow will establish connection with the database and display the latest information in the tables when the email is accessed. A call must be made to the following package procedure `XXDHI_NOTIFICATIONS_API_PKG.ADD_QUERY`.

This package procedure has the following parameters

### 1.4.1 X\_NOTIFICATION\_API\_ID **Data Type=IN NUMBER, Required=Yes**

The unique reference id returned by `XXDHI_notifications_api_pkg.send_notification`.

### 1.4.2 X\_QUERY\_TITLE\_TEXT **Data Type=IN VARCHAR2, Required=Yes**

The title/label of the given query. This text will be displayed in bold letters prior to displaying the result of SQL Query.

### 1.4.3 X\_FROM\_CLAUSE **Data Type=IN VARCHAR2, Required=Yes**

From clause of the SQL Statement. This clause must not contain the keyword "FROM".

### 1.4.4 X\_WHERE\_CLAUSE **Data Type=IN VARCHAR2, Required=Yes**

Where clause of the SQL Statement. This clause must not contain the keyword "WHERE". If the where clause contains bind variables then a value for parameter `X_BIND_VALUES` must be provided.

### 1.4.5 X\_BIND\_VALUES **Data Type=IN VARCHAR2, Required=No**

Value of the bind variables separated by comma.

### 1.4.6 X\_COLUMN\_TITLE ( 1 To 15) **Data Type=IN VARCHAR2, Required=No**

Title of the column for SELECT STATEMENT.

### 1.4.7 X\_COLUMN\_NAME( 1 To 15) **Data Type=IN VARCHAR2, Required=No**

Name of the column for SELECT STATEMENT.

## 1.5 Example ( Send a plain text message)

```
XXDHI_NOTIFICATIONS_API_PKG.SEND_NOTIFICATION
(  
  X_EMAIL_ADDRESS=>'anil_passi@xxClientxxre.com'  
  ,X_USER_NAME=>'UKEXPI'  
  ,X_MESSAGE_TYPE=>'TEXT'  
  ,X_PROCESS_SHORT_CODE=>'TEST'  
  ,X_MESSAGE_SUBJECT=>'Please authorise the outstanding FX deal 403. '  
  ,X_MESSAGE_TEXT=>'The FX Deal 403 is outstanding. Please authorise it ASAP.'  
  ,X_NOTIFICATION_API_ID=>n_not_id  
);
```

The above call to the API will result in the notification as below.

Detail notification for Passi, Mr. Anil (Anil) - Microsoft Internet Explorer provided by Swiss Re

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Full Screen Print

Address [C:\TEMP\~6103328.htm](#) Go Link

**Notification Details** **ORACLE**

To **Passi, Mr. Anil (Anil)** Sent **17-APR-2002 12:52:14** Closed **17-APR-2002 12:52:42**

Subject **Please authorize the outstanding FX deal 403**

The FX Deal 403 is outstanding. Please authorize it asap

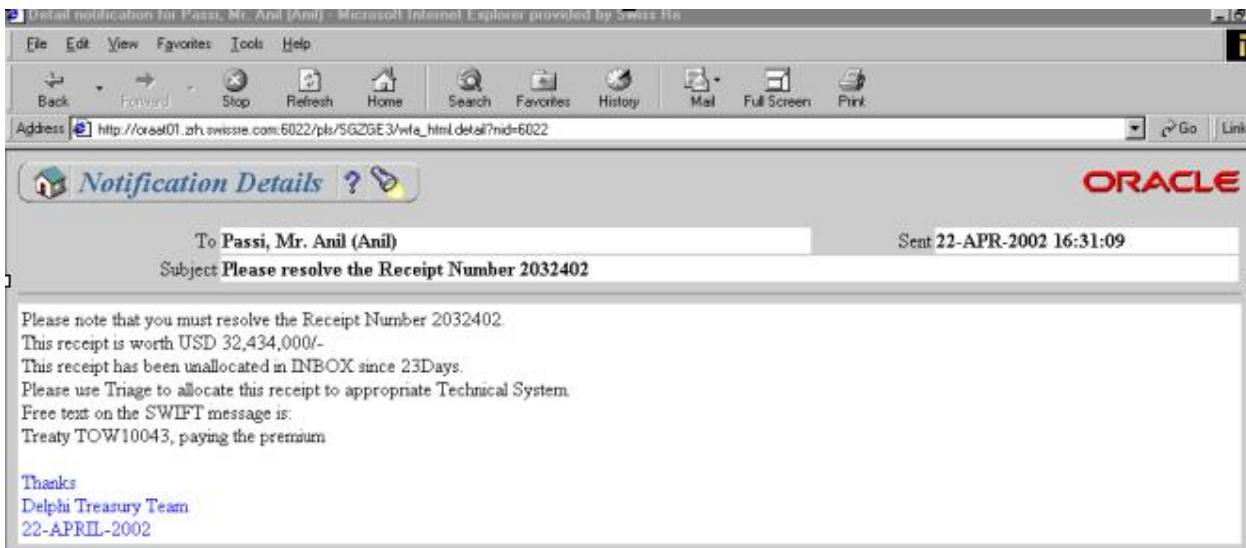
## 1.6 Example (Send static HTML message)

```

XXDHI_NOTIFICATIONS_API_PKG.SEND_NOTIFICATION
(
  X_EMAIL_ADDRESS=>'anil_passi@rcomext.com'
  ,X_USER_NAME=>'UKEXPI'
  ,X_MESSAGE_TYPE=>'HTML'
  ,X_PROCESS_SHORT_CODE=>'TEST'
  ,X_MESSAGE_SUBJECT=>' Please resolve the Receipt Number 2032402 '
  ,X_MESSAGE_TEXT=>
  '<body>
  <p>Please note that you must resolve the Receipt Number 2032402.<br>
  This receipt is worth USD 32,434,000/-<br>
  This receipt has been unallocated in INBOX since 23Days.<br>
  Please use Triage to allocate this receipt to appropriate Technical System.<br>
  Free text on the SWIFT message is: <br>
  Treaty TOW10043, paying the premium<br>
  </p>
  <p><font color="#0000FF">Thanks<br>
  XxProjxx Treasury Team<br>
  22-APRIL-2002<br>
  </font></p>
  </body>'
  ,X_NOTIFICATION_API_ID=>n_not_id
);

```

The above call to the API will result in the notification as below.



## 1.7 Example ( Static Message plus Dynamic From Query)

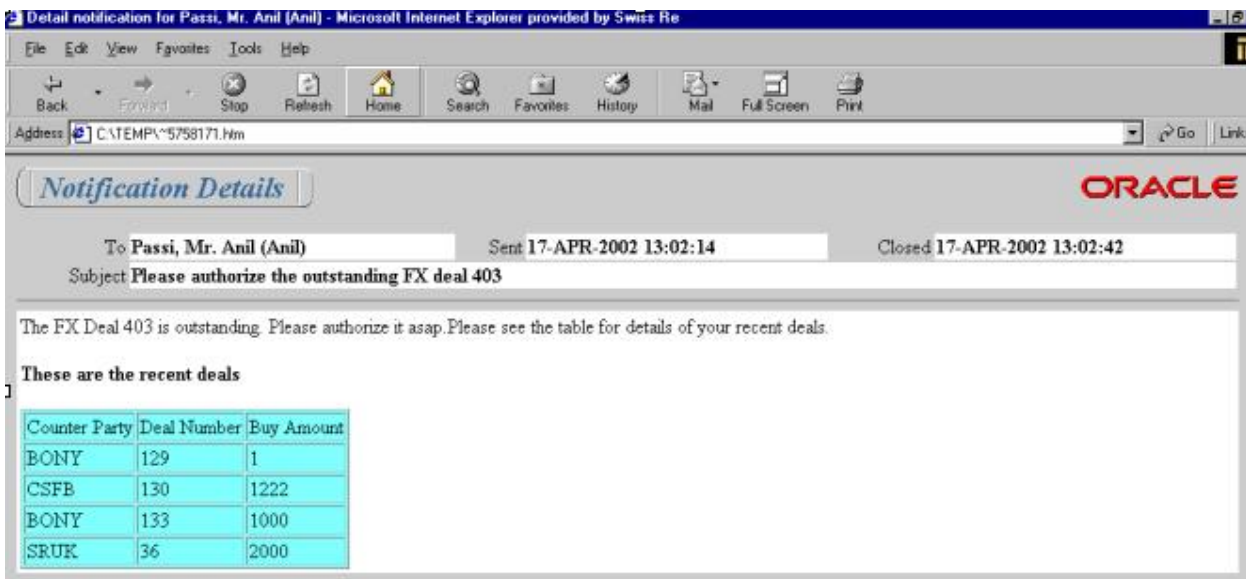
XXDHI\_NOTIFICATIONS\_API\_PKG.SEND\_NOTIFICATION

```
(
  X_EMAIL_ADDRESS           =>'anil_passi@rcomext.com'
  ,X_USER_NAME              =>'UKEXPI'
  ,X_MESSAGE_TYPE          =>'TEXT'
  ,X_PROCESS_SHORT_CODE    =>'TEST'
  ,X_MESSAGE_SUBJECT       =>'Please authorise the outstanding FX deal 403. '
  ,X_MESSAGE_TEXT          =>'The FX Deal 403 is outstanding. Please authorise it ASAP!'
  ,X_NOTIFICATION_API_ID   =>n_not_id
);
```

XXDHI\_notifications\_api\_pkg.add\_query

```
(
  X_NOTIFICATION_API_ID     => n_not_id
  ,X_QUERY_TITLE_TEXT      => 'These are the recent deals'
  ,X_FROM_CLAUSE           => 'xtr_deals'
  ,X_WHERE_CLAUSE          => 'ROWNUM < 5 AND dealer_code = "UKEXPI"'
  ,X_BIND_VALUES           => NULL
  ,x_column_title_1       => 'Counter Party'
  ,x_column_name_1        => 'CPARTY_CODE'
  ,x_column_title_2       => 'Deal Number'
  ,x_column_name_2        => 'deal_no'
  ,x_column_title_3       => 'Buy Amount'
  ,x_column_name_3        => 'buy_amount'
);
```

The above API calls will result in the following Notification



Notification Details

To: Passi, Mr. Anil (Anil) Sent: 17-APR-2002 13:02:14 Closed: 17-APR-2002 13:02:42

Subject: Please authorize the outstanding FX deal 403

The FX Deal 403 is outstanding. Please authorize it asap. Please see the table for details of your recent deals.

These are the recent deals

Counter Party	Deal Number	Buy Amount
BONY	129	1
CSFB	130	1222
BONY	133	1000
SRUK	36	2000

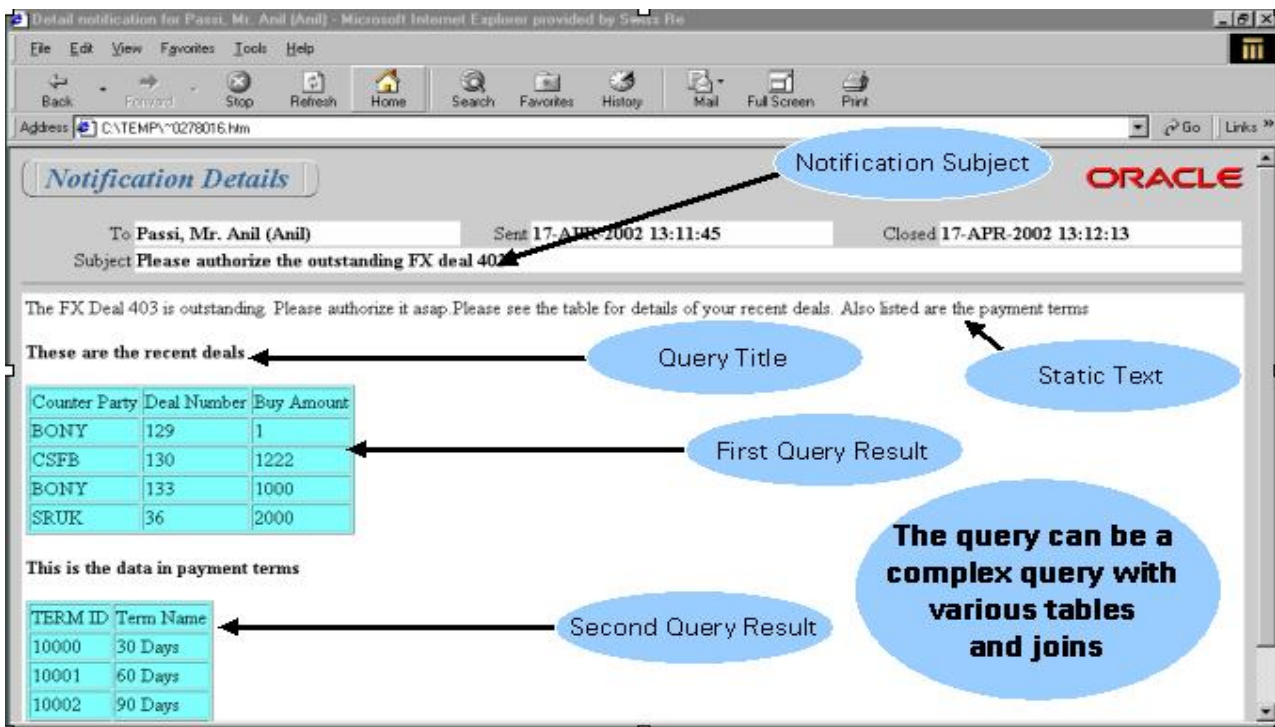
### 1.8 Example ( Static Message plus Dynamic From Multiple Queries )

**NOTE: There is no upper limit to the number of queries which can be passed to the notification API.**

```

XXDHI_NOTIFICATIONS_API_PKG.SEND_NOTIFICATION
(
  X_EMAIL_ADDRESS           =>'anil_passi@rcomext.com'
  X_USER_NAME               =>'UKEXPI'
  X_MESSAGE_TYPE            =>'TEXT'
  X_PROCESS_SHORT_CODE     =>'TEST'
  X_MESSAGE_SUBJECT        =>'Please authorise the outstanding FX deal 403.'
  X_MESSAGE_TEXT           =>'The FX Deal 403 is outstanding. Please authorise it ASAP.'
  X_NOTIFICATION_API_ID    =>n_not_id
);
XXDHI_notifications_api_pkg.add_query ----->(First Query)
(
  X_NOTIFICATION_API_ID => n_not_id
  X_QUERY_TITLE_TEXT    => 'These are the recent deals'
  X_FROM_CLAUSE         => 'xtr_deals'
  X_WHERE_CLAUSE        => 'ROWNUM < 5 AND dealer_code = "UKEXPI"'
  X_BIND_VALUES         => NULL
  x_column_title_1     => 'Counter Party'
  x_column_name_1      => 'CPARTY_CODE'
  x_column_title_2     => 'Deal Number'
  x_column_name_2      => 'deal_no'
  x_column_title_3     => 'Buy Amount'
  x_column_name_3      => 'buy_amount'
);
XXDHI_notifications_api_pkg.add_query -----> (Second Query)
(
  X_NOTIFICATION_API_ID => n_not_id
  X_QUERY_TITLE_TEXT    => 'This is the data in payment terms'
  X_FROM_CLAUSE         => 'ap_terms'
  X_WHERE_CLAUSE        => 'ROWNUM < 5'
  X_BIND_VALUES         => NULL
  x_column_title_1     => 'TERM ID'
  x_column_name_1      => 'TERM_ID'
  x_column_title_2     => 'Term Name'
  x_column_name_2      => 'name'
);
  
```

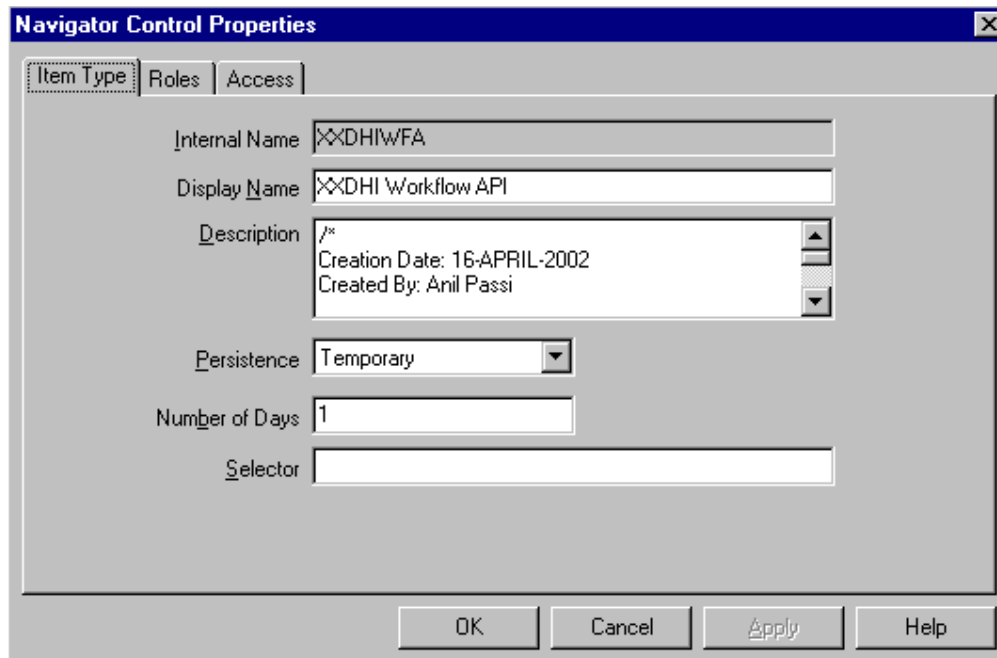
The below HTML notification is generated and emailed automatically to user after example API is invoked.



## 2 Workflow Item Type

The item type is the unique name given to every workflow in Oracle.

Internal Name : XXDHIWFA



Display Name : XXDHI Workflow API

All the objects within the item type of this workflow stored in the file XXDHIWFA.wft. This file will be used to install the workflow on any other environment. The installation instructions been detail in a latter section of this document.

---

## 3 Workflow Processes

We can design workflow processes to accomplish various business tasks. A workflow process comprises of various activities. For this workflow only one process definition is required.

---

### 3.1 Workflow API Process

This workflow process constitutes of the following steps



---

#### 3.1.1 Initiate Workflow Variables

This step assigns a unique reference number to the workflow attribute NOTIFICATION\_API\_ID. NOTIFICATION\_API\_ID is used to uniquely track each notification.

In addition to the above, this process step stores the message subject and pl/sql document reference in workflow attributes.

---

#### 3.1.2 Create and Set Roles

This process step creates a role for the email address, if a role currently does not exist.

---

#### 3.1.3 HTML Document Notification

This process step builds the document which is displayed on the email.

## 4 Other Workflow Components

### 4.1 Workflow Attributes

Attributes are used as global variables in workflow. The attributes defined in workflow item type are shared by all the processes within the item type. Following attributes will be defined for this workflow. Its certain that more attributes will be added during the build process of the workflow. The additional attributes added must be reflected in this document.

Attribute Short Name	Attribute Display Name	Data Type
NOTIFICATION_API_ID	Notification API ID	Number
NOTIFICATION_DOCUMENT	Notification Document	Document
SEND_TO_ROLE	Send To Role	Role
NOTIFICATION_SUBJECT	Notification Subject	Text

### 4.2 Workflow Functions

Functions are used control the flow of the processes and also to perform database activities. The functions defined in workflow item type are shared by all the processes within the item type. Following functions will be defined for this workflow. Its certain that more functions will be added during the build process of the workflow. The additional functions added must be reflected in this document.

Function Short Name	Function Display Name	Type
CREATE_SET_ROLE	Create and Set Role	PL/SQL
INIT_WORKFLOW	Initiate Workflow Variables	PL/SQL

### 4.3 Notifications and Messages

Every notification is associated with a workflow message. The message can either be a static message or it can be dynamically constructed from a pl/sql procedure. This workflow will use following messages and notifications.

Notification Name	Associated Message Name
HTML Document Notification	HTML Document Notification

### 4.4 Workflow Lookups

Lookups can be used in workflows for the following reasons

- To define the result of an activity, process, notification

## 5 Testing

In order to test this module, the following types of test should be run:

Type of Test	Details e.g. data to be used
Create static notification for an email address. If the email address is not registered in Local roles, then a new role must be created	
Create notification for a Oracle Query.	
Create notification for multiple Oracle Queries. More than one table will be displayed.	
Create a complex query using various table joins	
Create a query using bind variables and bind values	

## 6 Technical Details

This section contains technical notes to guide the person building the program e.g. tables to be accessed, templates and techniques to be used.

### 6.1 General

#### 6.1.1 Design Methodology

Where possible standard tables will be used in the design. This will reduce the programming effort and will allow XxClientxx Re to use the standard APPS reports where possible. All the bespoke objects will begin with XXDHI, this will be standard naming convention for customizations in Oracle Treasury XxProjxx Project.

### 6.2 Table access/bespoke table creation

Following tables will be accessed by this workflow definition.

OBJECT	SEL	INS	UPD	DEL	CRE
XXDHI_NOTIFICATIONS_API	X	X			X
XXDHI_NOTIFICATION_QUERIES	X	X			X
XXDHI_NOTIFICATION_COLUMNS	X	X			X
WF_LOCAL_ROLES	X	X			
WF_ROLES	X				
WF_ITEMS					
DUAL	X				

### 6.3 Database Triggers

Trigger XXDHI\_NOTIFICATIONS\_API\_ARI will be created in the apps schema. This trigger will be executed when a record is inserted in table XXDHI\_NOTIFICATIONS\_API.

```

CREATE OR REPLACE TRIGGER XXDHI_NOTIFICATIONS_API_ARI
AFTER INSERT on XXDHI_NOTIFICATIONS_API
FOR EACH ROW
BEGIN

--This call will start the workflow
XXDHI_notifications_api_pkg.START_WORKFLOW_API
(
    P_NOTIFICATION_API_ID => :new.notification_api_id
    ,P_PROCESS_CODE => :new.process_short_code
);
END ;

```

### 6.4 Database Package

The entire notification creation logic for this workflow will reside in package XXDHI\_NOTIFICATIONS\_API\_PKG.

This package will contain following procedures which will be invoked from various activities in workflow

```

PROCEDURE ADD_QUERY

PROCEDURE CONSTRUCT_SQL_STATEMENT

PROCEDURE CREATE_SET_ROLES

FUNCTION EMAIL_TO_ROLE RETURNS VARCHAR2

PROCEDURE GET_COLUMN_TITLE_LIST

PROCEDURE GET_NOTIFICATION_DOCUMENT

PROCEDURE INIT_WORKFLOW_API_VARIABLES

PROCEDURE SEND_NOTIFICATION

PROCEDURE START_WORKFLOW_API

```

## 6.5 Prototype Source Code

```

PACKAGE BODY XXDHI_notifications_api_pkg
--Set the following global variables
--These variables will primarily be used to build the HTML table
g_bold_text_begin  VARCHAR2(30) := '<p><strong>';
g_bold_text_end    VARCHAR2(30) := '</strong></p>';

g_table_begin     VARCHAR2(40) := '<table border="1" bgcolor="#80FFFF">';
g_table_end       VARCHAR2(30) := '</table>';

g_record_begin    VARCHAR2(5)  := '<tr>';
g_record_end      VARCHAR2(5)  := '</tr>';

g_column_begin    VARCHAR2(5)  := '<td>';
g_column_end      VARCHAR2(5)  := '</td>';

g_item_owner     VARCHAR2(30) := 'UKEXPI';

g_response_value  VARCHAR2(30) := 'XXDHI_RESPONSE_VALUE';

/*
This procedure will build the first record for HTML table
Since we have one to many relation between Query and column list,
We will loop for each column record and build the title record
*/
PROCEDURE get_column_title_list
  Parameters
    P_NOTIFICATION_QUERY_ID IN INTEGER
  ,P_COLUMN_TITLE_LIST     OUT VARCHAR2
  LOOP FOR all the columns in  XXDHI_notification_columns
    where
      NOTIFICATION_QUERY_ID = P_NOTIFICATION_QUERY_ID
    ORDER BY notification_column_id

    Build the column list in variable P_COLUMN_TITLE_LIST
  END LOOP ;
--Now add the record tags to begin and end record
  P_COLUMN_TITLE_LIST := g_record_begin || P_COLUMN_TITLE_LIST || g_record_end ;
END get_column_title_list ;

/*
The output of all the columns will be concatenated.
*/
PROCEDURE get_column_list
  Parameters
    P_NOTIFICATION_QUERY_ID IN INTEGER
  ,P_COLUMN_LIST           OUT VARCHAR2
  BEGIN
  FOR p_rec IN ( SELECT
    "" || g_column_begin || "" || ""
    ||
    COLUMN_NAME || ""
    ||

```

```

        "" || g_column_end || "" "HTML_COL"
    from
        XXDHI_notification_columns
    where
        NOTIFICATION_QUERY_ID = P_NOTIFICATION_QUERY_ID
    ORDER BY notification_column_id
)
LOOP
    Build variable P_COLUMN_LIST
END LOOP ;
END get_column_list ;

/* Please note, this procedure will append FROM and WHERE keywords to the respect clauses. Hence the parameters passed to this API should not use FROM and WHERE in the
clauses, as the API appends these keywords. */
PROCEDURE construct_sql_statement
Parameters
P_NOTIFICATION_QUERY_ID IN INTEGER
,p_query_statement OUT VARCHAR2

v_column_list VARCHAR2(4000) ;
BEGIN
    --Get the column list
    Call get_column_list

    --Now pass the column list
    OPEN and Fetch Cursor c_get_stmt into p_query_statement.
    This builds the query statement
END construct_sql_statement ;

/* Not doing anything intelligent here. Inserting records in XXDHI_NOTIFICATION_QUERIES and XXDHI_NOTIFICATION_COLUMNS At the moment, max limit of columns
for a query is 15, but this can be easily expanded. No changes will be required to the data model if u add further parameters for column names.
Column width is for future use only. The developer using this API may ignore column width. The HTML table construct will take care of column sizing automatically */
PROCEDURE add_query
(
X_NOTIFICATION_API_ID          IN INTEGER
,X_QUERY_TITLE_TEXT            IN VARCHAR2
,X_FROM_CLAUSE                 IN VARCHAR2
,X_WHERE_CLAUSE                IN VARCHAR2
,X_BIND_VALUES                 IN VARCHAR2 DEFAULT NULL
,x_column_title_1..15          IN VARCHAR2 DEFAULT NULL
,x_column_width_1..15          IN VARCHAR2 DEFAULT NULL
,x_column_name_1..15           IN VARCHAR2 DEFAULT NULL
) IS
    n_notification_query_id INTEGER ;
    n_notification_column_id INTEGER ;
BEGIN
    /*
    First create query record
    Next create the column records one by one
    */

    /*
    Note at the moment no validation done for column
    name, width and title be mutually inclusive
    */

    select XXDHI_notification_queries_s.nextval INTO n_notification_query_id from dual ;

    /* Create the query record */
    XXDHI_NOTIFICATIONS_INSERT_PKG.XXDHI_NOTIFICATION_QUERIES
    (
        X_NOTIFICATION_QUERY_ID =>n_notification_query_id
    ,X_NOTIFICATION_API_ID      =>X_NOTIFICATION_API_ID
    ,X_QUERY_TITLE_TEXT         =>X_QUERY_TITLE_TEXT
    ,X_FROM_CLAUSE              =>X_FROM_CLAUSE
    ,X_WHERE_CLAUSE             =>X_WHERE_CLAUSE
    ,X_BIND_VALUES              =>X_BIND_VALUES
    ) ;

    /* Now create notification columns */
END add_query ;

PROCEDURE send_notification
(
x_email_address              IN VARCHAR2
,x_user_name                 IN VARCHAR2 DEFAULT NULL
,x_message_type              IN VARCHAR2
,x_message_subject           IN VARCHAR2
,x_process_short_code        IN VARCHAR2
,x_message_text              IN VARCHAR2 DEFAULT NULL
,x_intranet_accessible       IN VARCHAR2 DEFAULT 'Y'
,X_RESPONSE_LOOKUP_TYPE      IN VARCHAR2 DEFAULT NULL
,X_NOTIFICATION_API_ID       OUT INTEGER
,X_RESPONSE_CALLBACK_SQL     IN VARCHAR2 DEFAULT NULL
) IS
    N_NOTIFICATION_API_ID    INTEGER ;
BEGIN

```

```

        XXDHL_notifications_insert_pkg.XXDHL_NOTIFICATIONS_API
        (
        X_NOTIFICATION_API_ID=>N_NOTIFICATION_API_ID
        ,X_EMAIL_ADDRESS =>X_EMAIL_ADDRESS
        ,X_MESSAGE_TYPE =>X_MESSAGE_TYPE
        ,X_MESSAGE_SUBJECT =>X_MESSAGE_SUBJECT
        ,X_PROCESS_SHORT_CODE =>X_PROCESS_SHORT_CODE
        ,X_USER_NAME =>X_USER_NAME
        ,X_INTRANET_ACCESSIBLE=>X_INTRANET_ACCESSIBLE
        ,X_MESSAGE_TEXT =>X_MESSAGE_TEXT
        ,X_RESPONSE_LOOKUP_TYPE=>X_RESPONSE_LOOKUP_TYPE
        ,X_RESPONSE_CALLBACK_SQL=>X_RESPONSE_CALLBACK_SQL
        );
END send_notification ;

PROCEDURE START_WORKFLOW_API
(
P_NOTIFICATION_API_ID IN INTEGER
,P_PROCESS_CODE IN VARCHAR2
) IS
L_save_thr NUMBER ;
L_itemkey VARCHAR2(100) ;
L_itemtype VARCHAR2(100) := 'XXDHIWFA';
BEGIN
L_save_thr := Wf_Engine.threshold; -- to allow run in trigger

Wf_Engine.threshold := -1;

L_itemkey := P_PROCESS_CODE || '-'
|| TO_CHAR(P_NOTIFICATION_API_ID) ;

Wf_Engine.CreateProcess(L_itemtype, L_itemkey,'API_PROCESS');

Wf_Engine.SetItemUserKey
(
itemtype => L_itemtype
,itemkey => L_itemkey
,userkey => P_PROCESS_CODE || '-'
||
TO_CHAR(P_NOTIFICATION_API_ID)
);

Wf_Engine.SetItemOwner (
itemtype => L_itemtype
,itemkey => L_itemkey
,owner => g_item_owner );

wf_engine.SetItemAttrNumber
(itemtype => L_itemtype,
itemkey => L_itemkey,
aname => 'NOTIFICATION_API_ID',
avalue => TO_CHAR(P_NOTIFICATION_API_ID) );

Wf_Engine.StartProcess(L_itemtype,L_itemkey);
Wf_Engine.threshold := L_save_thr ;

END START_WORKFLOW_API ;

/* COnvert the email to a role name
This is done, becos we must give a role name to each email */
FUNCTION email_to_role
(
p_email IN VARCHAR2
) RETURN VARCHAR2 IS
BEGIN
RETURN
UPPER(
REPLACE (
REPLACE ( p_email , ',', '_' ) , '@', '_' )
);
END email_to_role ;

/*
Returns the ROLE for a given
user and email address combination.
*/
FUNCTION get_role_for_user_email
(
p_user IN VARCHAR2
,p_email IN VARCHAR2
) RETURN VARCHAR2 IS
CURSOR c_get_role IS
SELECT
*
FROM
wf_roles

```

```

WHERE
    name = p_user
    and
    UPPER(email_address) = UPPER(p_email) ;

CURSOR c_get_email IS
SELECT
*
FROM
    wf_roles
WHERE
    name = email_to_role ( p_email )
    and
    email_address = p_email ;

p_get_role c_get_role%ROWTYPE ;

BEGIN
OPEN c_get_role ;
FETCH c_get_role INTO p_get_role ;
IF c_get_role%FOUND
THEN
    CLOSE c_get_role ;
    RETURN p_get_role.name ;
END IF ;
CLOSE c_get_role ;

OPEN c_get_email ;
FETCH c_get_email INTO p_get_role ;
IF c_get_email%FOUND
THEN
    CLOSE c_get_email ;
    RETURN p_get_role.name ;
END IF ;
CLOSE c_get_email ;

RETURN NULL ;
END get_role_for_user_email ;

/*
Create a ROLE
and return the ROLE name
*/
FUNCTION create_role_email
(
    p_user IN VARCHAR2
    ,p_email IN VARCHAR2
) RETURN VARCHAR2
IS
    v_role          WF_ROLES.NAME%TYPE := email_to_role ( p_email ) ;
BEGIN
    Wf_Directory.CreateAdHocRole
    (
        ROLE_NAME => v_role
        , ROLE_DISPLAY_NAME => v_role
        , ROLE_DESCRIPTION => v_role
        , NOTIFICATION_PREFERENCE => 'MAILHTML'
        , EMAIL_ADDRESS => p_email
        , STATUS => 'ACTIVE'
        , EXPIRATION_DATE => SYSDATE
    ) ;
    RETURN v_role ;
END create_role_email ;

PROCEDURE CREATE_SET_ROLES
(
    itemtype in varchar2,
    itemkey in varchar2,
    actid in number,
    funcmode in varchar2,
    result in out varchar2
)
IS
    v_email_address XXDHI_notifications_api.email_address%TYPE ;
    v_user_name XXDHI_notifications_api.user_name%TYPE ;
    v_role_name WF_ROLES.NAME%TYPE ;
    n_notification_api_id INTEGER ;

    CURSOR c_get_email_and_user
    IS
    SELECT
        EMAIL_ADDRESS
        ,USER_NAME
    FROM
        XXDHI_notifications_api
    WHERE

```

```

notification_api_id = n_notification_api_id ;

l_error VARCHAR(100) := 'CREATE_SET_ROLES' ;
e_user_exception EXCEPTION ;

BEGIN
n_notification_api_id :=
wf_engine.getitemattnumber
(
itemtype => itemtype
,itemkey => itemkey
,aname => 'NOTIFICATION_API_ID'
);
OPEN c_get_email_and_user ;
FETCH c_get_email_and_user INTO v_email_address, v_user_name ;
CLOSE c_get_email_and_user ;
IF v_email_address IS NULL AND v_user_name IS NULL THEN
l_error := 'Email and User name are null' ;
RAISE e_user_exception ;
END IF ;
v_role_name :=
get_role_for_user_email
(
p_user => v_user_name
,p_email => v_email_address
);
IF v_role_name IS NULL THEN
--create role here
v_role_name := create_role_email
(
p_user => v_user_name
,p_email => v_email_address
);
END IF ;

IF v_role_name IS NULL THEN
l_error := 'Unable to create role' ;
RAISE e_user_exception ;
END IF ;

--Set the role here
wf_engine.SetItemAttrText
(itemtype => itemtype,
itemkey => itemkey,
aname => 'SEND_TO_ROLE',
avalue => v_role_name );

result := 'COMPLETE:Y' ;
EXCEPTION
WHEN OTHERS THEN
Wf_core.Context
(
'XXDHL_NOTIFICATIONS_API_PKG'
,'CREATE_SET_ROLES'
,itemkey
,TO_CHAR(actid)
,funcmode,l_error);
RAISE;

END CREATE_SET_ROLES ;

/* This will initialise the message body
Will also initialize the message document procedure
along with the parameters for generating the doc*/
PROCEDURE init_workflow_api_variables
(
itemtype in varchar2,
itemkey in varchar2,
actid in number,
funcmode in varchar2,
result out varchar2
)
IS

n_notification_api_id INTEGER ;

v_message_subject XXDHL_notifications_api.MESSAGE_SUBJECT%TYPE ;

CURSOR c_get( p_notification_api_id IN INTEGER )
IS
SELECT
MESSAGE_SUBJECT
FROM
XXDHL_notifications_api
WHERE
notification_api_id = p_notification_api_id ;
BEGIN
--Get the notification API Id

```

```

n_notification_api_id :=
wf_engine.getitemattnumber
(
  itemtype => itemtype
  ,itemkey => itemkey
  ,aname   =>'NOTIFICATION_API_ID'
);

/* get the message subject */
OPEN c_get( p_notification_api_id=> n_notification_api_id );
FETCH c_get INTO v_message_subject ;
CLOSE c_get ;

wf_engine.SetItemAttrText
(
  itemtype => itemtype,
  itemkey => itemkey,
  aname   => 'NOTIFICATION_SUBJECT',
  avalue  => v_message_subject
);

--And then set the document attribute
wf_engine.SetItemAttrText (itemtype => itemtype,
  itemkey => itemkey,
  aname   => 'NOTIFICATION_DOCUMENT',
  avalue  => 'PLSQL:XXDHL_NOTIFICATIONS_API_PKG.GET_NOTIFICATION_DOCUMENT' ||
    itemtype || ':' ||
    to_char(n_notification_api_id) );

result := 'COMPLETE' ;
END init_workflow_api_variables ;

/*
This procedure will be the backbone of
generating the message for this API workflow
*/
PROCEDURE GET_NOTIFICATION_DOCUMENT
(document_id      in          varchar2,
display_type     in          varchar2,
document         in out     varchar2,
document_type    in out     varchar2)
IS
l_item_type      wf_items.item_type%TYPE;
l_item_key       wf_items.item_key%TYPE;
l_notification_api_id INTEGER ;
v_long_variable VARCHAR2(4000) ;
v_message_type  XXDHL_notifications_api.MESSAGE_TYPE%TYPE ;

type notification_ref_cursor is REF CURSOR ;
notification_cursor notification_ref_cursor ;

/*
For some reasons bulk fetch didnt work, hence commenting
the pl/sql table rec_tab
*/
--type result_tab is table of varchar2(4000);
--rec_tab result_tab ;

CURSOR c_get_text_html ( p_notification_api_id IN INTEGER )
IS
SELECT
  MESSAGE_TEXT
  ,MESSAGE_TYPE
FROM
  XXDHL_notifications_api
WHERE
  NOTIFICATION_API_ID = p_notification_api_id
  and
  MESSAGE_TYPE IN ( 'TEXT', 'HTML', 'TEXT_AND_QUERY' )
;

BEGIN
l_item_type := substr(document_id, 1, instr(document_id, ':') - 1);
l_notification_api_id := substr(document_id, instr(document_id, ':') + 1 ) ;

OPEN c_get_text_html ( p_notification_api_id => l_notification_api_id ) ;
FETCH c_get_text_html INTO document, v_message_type ;
CLOSE c_get_text_html ;

/*
The main message text has been captured as above
Next we need to generate the dynamic SQL and display its
contents in the table format of HTML
*/

IF v_message_type NOT IN ( 'QUERY', 'TEXT_AND_QUERY' ) THEN
RETURN ;

```

```

END IF ;
FOR p_rec IN ( SELECT
                notification_query_id
                ,QUERY_TITLE_TEXT
            from
            XXDHL_notification_queries
        where
            NOTIFICATION_API_ID = l_notification_api_id
            order by notification_query_id
        )
LOOP
    v_long_variable := NULL ;
    XXDHL_notifications_api_pkg.construct_sql_statement
    (
        P_NOTIFICATION_QUERY_ID => p_rec.NOTIFICATION_QUERY_ID
        ,p_query_statement => v_long_variable
    );

    document := document
                || chr(10)
                || g_bold_text_begin
                || p_rec.QUERY_TITLE_TEXT
                || g_bold_text_end
                || chr(10);
    document := document || chr(10) || g_table_begin || chr(10);

    /*
    Although bulk fetch might be more efficient, but
    it invokes invalid cursor in this case
    FETCH          notification_cursor BULK COLLECT INTO rec_tab ;
    FOR i in 1..notification_cursor%rowcount
    LOOP
        document := document || rec_tab(i);
    END LOOP ;
    */

    /*
    I am reusing the v_long_variable variable for
    different purposes ,because i do not wish to
    define another 4000 byte variable to hold data
    */
    OPEN          notification_cursor FOR v_long_variable ;

    get_column_title_list
    (
        P_NOTIFICATION_QUERY_ID => p_rec.NOTIFICATION_QUERY_ID
        ,P_COLUMN_TITLE_LIST => v_long_variable
    );
    --This will append the Column titles
    document := document || v_long_variable ;
    LOOP
        FETCH          notification_cursor INTO v_long_variable ;
        IF notification_cursor%NOTFOUND THEN
            EXIT ;
        END IF ;
        document := document || v_long_variable ;
    END LOOP ;

    document := document || chr(10) || g_table_end || chr(10);

END LOOP ;

END GET_NOTIFICATION_DOCUMENT ;

PROCEDURE get_response_type
(
    itemtype in varchar2,
    itemkey in varchar2,
    actid in number,
    funcmode in varchar2,
    result in out varchar2
)
IS
n_notification_api_id INTEGER ;

v_response_type XXDHL_notifications_api.RESPONSE_LOOKUP_TYPE%TYPE ;

CURSOR c_get( p_notification_api_id IN INTEGER )
IS
SELECT
    RESPONSE_LOOKUP_TYPE
FROM
    XXDHL_notifications_api
WHERE
    notification_api_id = p_notification_api_id ;
BEGIN
    --Get the notification API Id

```

```

n_notification_api_id :=
wf_engine.getitemattrnumber
(
  itemtype => itemtype
  ,itemkey => itemkey
  ,aname =>'NOTIFICATION_API_ID'
);

/* get the response type */
OPEN c_get( p_notification_api_id=> n_notification_api_id );
FETCH c_get INTO v_response_type ;
CLOSE c_get ;

result := NVL ( v_response_type, 'NONE' );
END get_response_type ;

PROCEDURE set_callback_result
(
  itemtype in varchar2,
  itemkey in varchar2,
  actid in number,
  funcmode in varchar2,
  result in out varchar2
) IS
l_nid NUMBER ;

l_activity_result_code varchar2(200) ;
n_notification_api_id INTEGER ;
BEGIN
IF (FUNCMODE IN ('RESPOND')) THEN
SELECT notification_id,
  activity_result_code
  INTO l_nid,
  l_activity_result_code
  FROM wf_item_activity_statuses

WHERE item_type = itemtype
AND item_key = itemkey
AND process_activity = actid ;

n_notification_api_id := Wf_Notification.GetAttrNumber(l_nid, 'NOTIFICATION_API_ID');

UPDATE XXDHI_notifications_api
SET
RESPONSE_LOOKUP_RESULT = l_activity_result_code
WHERE
notification_api_id = n_notification_api_id ;

ELSIF (FUNCMODE = 'TIMEOUT') THEN
NULL;

END IF ;
result := 'COMPLETE:APPROVED' ;
EXCEPTION
WHEN OTHERS THEN
result := SQLERRM ;
END set_callback_result ;

PROCEDURE execute_callback
(
  itemtype in varchar2,
  itemkey in varchar2,
  actid in number,
  funcmode in varchar2,
  result in out varchar2
)
IS
n_notification_api_id INTEGER ;

v_RESPONSE_LOOKUP_RESULT XXDHI_notifications_api.RESPONSE_LOOKUP_RESULT%TYPE ;
v_RESPONSE_CALLBACK_SQL XXDHI_notifications_api.RESPONSE_CALLBACK_SQL%TYPE ;

CURSOR c_get( p_notification_api_id IN INTEGER )
IS
SELECT
  RESPONSE_LOOKUP_RESULT
  ,RESPONSE_CALLBACK_SQL
FROM
  XXDHI_notifications_api
WHERE
  notification_api_id = p_notification_api_id ;
BEGIN
--Get the notification API Id
n_notification_api_id :=
wf_engine.getitemattrnumber
(
  itemtype => itemtype

```

```
        ,itemkey => itemkey
        ,aname   =>'NOTIFICATION_API_ID'
    );

/* get the response type */
OPEN c_get( p_notification_api_id=> n_notification_api_id );
FETCH c_get INTO v_RESPONSE_LOOKUP_RESULT, v_RESPONSE_CALLBACK_SQL ;
CLOSE c_get ;

IF v_RESPONSE_CALLBACK_SQL IS NULL THEN
    result := 'COMPLETE:Y' ;
    RETURN ;
ELSE
    SELECT
        REPLACE( v_RESPONSE_CALLBACK_SQL
                , g_response_value
                , v_RESPONSE_LOOKUP_RESULT )
    INTO
        v_RESPONSE_CALLBACK_SQL
    FROM
        DUAL ;
END IF ;
execute immediate v_RESPONSE_CALLBACK_SQL ;

    result := 'COMPLETE:Y' ;
EXCEPTION
    WHEN OTHERS THEN
        --need to write proper code when we
        --implement prototype
        result := 'COMPLETE:Y' ;
END execute_callback ;

END XXDHI_notifications_api_pkg ;
```

---

## 6.6 System Setup Required

### 6.6.1 Value Sets

---

None

### 6.6.2 Descriptive Flex Fields

---

None

### 6.6.3 Common Lookups

---

None

---

## 6.7 Configuration Items

This section lists all the Software Configuration Items that will be produced by this customisation. The sequence of listing should preferably be the same as their install sequence.

Configuration Item Short Name	Program Name	Installed Location	Program Language
XXDHIWFA00.sql		\$XXSRC_TOP/install/sql	PL/SQL
XXDHIWFA01.sql		\$XXSRC_TOP/install/sql	PL/SQL
XXDHIWFA02.sql		\$XXSRC_TOP/install/sql	PL/SQL
XXDHIWFA03.sql		\$XXSRC_TOP/install/sql	PL/SQL
XXDHIWFA04.sql		\$XXSRC_TOP/install/sql	PL/SQL
XXDHIWFA05.sql		\$XXSRC_TOP/install/sql	PL/SQL

---

## 6.8 Outline Installation Instructions

### 6.8.1 Dependencies

---

None

### 6.8.2 Manual Pre-Installation Steps

---

Open the Workflow Builder in the test environment and from the database open item type "XXDHI Workflow API". Once the workflow has been opened, use File->Save As->XXDHIWFA.wft

FTP the file to \$XXSRC\_TOP/wf in the test environment.

### 6.8.3 Installation Steps

---

In SqlPlus, run the following scripts

XXDHIWFA00.sql

XXDHIWFA01.sql

XXDHIWFA02.sql

XXDHIWFA03.sql

XXDHIWFA04.sql

XXDHIWFA05.sql

On Unix, in the new environment , run the following command

```
wfload apps/<apps_passwd> $XXSRC_TOP/wf/XXDHIWFA.wft
```

---

## 7 Open and Closed Issues for this Deliverable

---

### 7.1 Open Issues

Issue	Resolution	Responsibility	Target Date	Impact Date

---

### 7.2 Closed Issues

Issue	Resolution	Responsibility	Target Date	Impact Date

## 8 Review Comments

### 8.1 Draft 1A

**Outcome:** please delete one of the following to indicate your choice

**Y** **ACCEPTED** (provided comments on this form are addressed)

**Q** **NOT ACCEPTED** (wish to re-review once all review comments have been actioned)

	Page/ Ref	Comment	Action (this section to be completed by Author to indicate action taken to address each review comment)
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			