

# Fueling the Workflow Engine

By Dan Stober  
Intermountain Healthcare  
April 15, 2008

# Agenda

1. **Workflow Tables**
2. **Workflow Scripts and APIs**
3. **Error Handling**
4. **Coding to Implement Function Activities**
5. **Three Examples**
6. **Other code:**
  - **A Concurrent Program**
  - **Selector Functions**
  - **Document Type Attributes**
7. **Conclusion and Wrap Up - Questions**

# Objectives

- 1. Learn how to custom procedures to be called from Workflow**
- 2. Discover how to communicate back to the calling Workflow process**
- 3. Understand the different types of procedures you can write for Workflow and their uses**
- 4. See several examples**
- 5. Know when to COMMIT and when not to**

# Target Audience

- **Primarily: Developers**
  - **Some Experience with WF**
    - Intermediate level
    - Knowledge of WF tables helpful
  - **Proficient**
    - PLSQL, SQL
  - **Exposure, helpful**
    - SQL
- **Are you just curious?**

# About the Presenter

- Attended California State Univ Fresno
- BS – Business, Accounting Specialty
- Software Engineer
  - Oracle EBS HR/Payroll - Since 2001
- Prior Presentations:
  - Oracle Open World – 2007, 2006
  - Collaborate – 2006 (2)
  - OAUG Live 2004
  - UTOUG Training Days – 2007 (2), 2008
- Perspective



# Why Am I Here?

- Researching presentations and writing white papers makes me better
  - Read documentation thoroughly and test
  - Intentional breaking of application prepares me to understand it when I see it in production
- I learn something from an attendee at every presentation I do
  - Please! Feel free to contribute!
- I truly enjoy it!

# Intermountain Healthcare



- Non-profit integrated health care system
- Based in Salt Lake City
- 22 Hospitals in two states
- 750 Employed physicians
- 30,000 employees
- Largest non-government employer in Utah
- An innovator in applying technology to Health Care
  - Decision Support
  - Care Protocols
  - Clinical Data Mining

# Intermountain Healthcare



- 10g Database
- Oracle E-Business Suite – 11.5.10.2
  - HR – 1999
  - Payroll – 1999
  - Employee Self Service – 2002
  - Managers' Self Service (SSHR) - 2003
  - Advanced Benefits – 2004
  - Compensation Workbench – 2006
  - Release 12 – 2008?
- Oracle Workflow since 2003



# Section 1

## Start Your Engines!



## Understanding the Workflow Tables

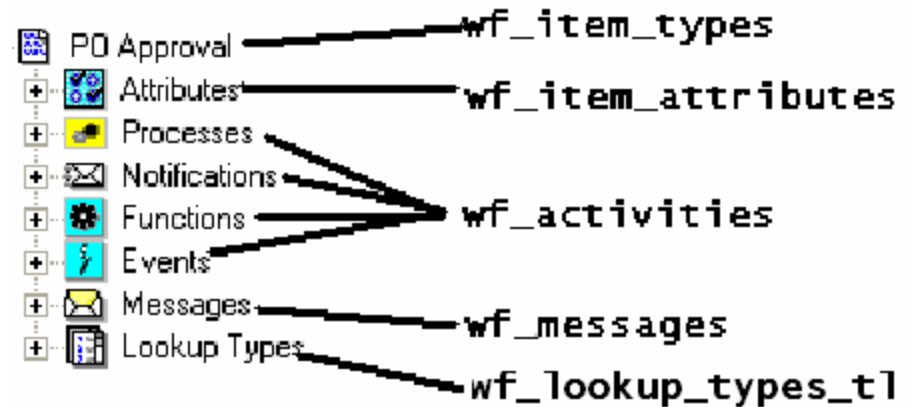
# Workflow Tables Dichotomy

- Definition Tables
  - Roadmap to our process
  - Guide signs
- Runtime Tables
  - Vehicles on our routes

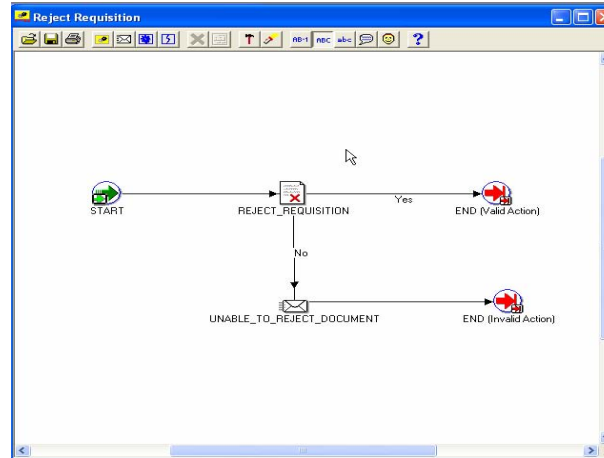


# Workflow Definition Tables

- wf\_item\_types
- wf\_item\_attributes
- wf\_activities
- wf\_activity\_attributes
- wf\_activity\_attr\_values
- wf\_messages
- wf\_message\_attributes
- wf\_process\_activities
- wf\_activity\_transitions
- wf\_lookup\_types\_tl
- wf\_lookups\_tl



# Process Activities & Transitions



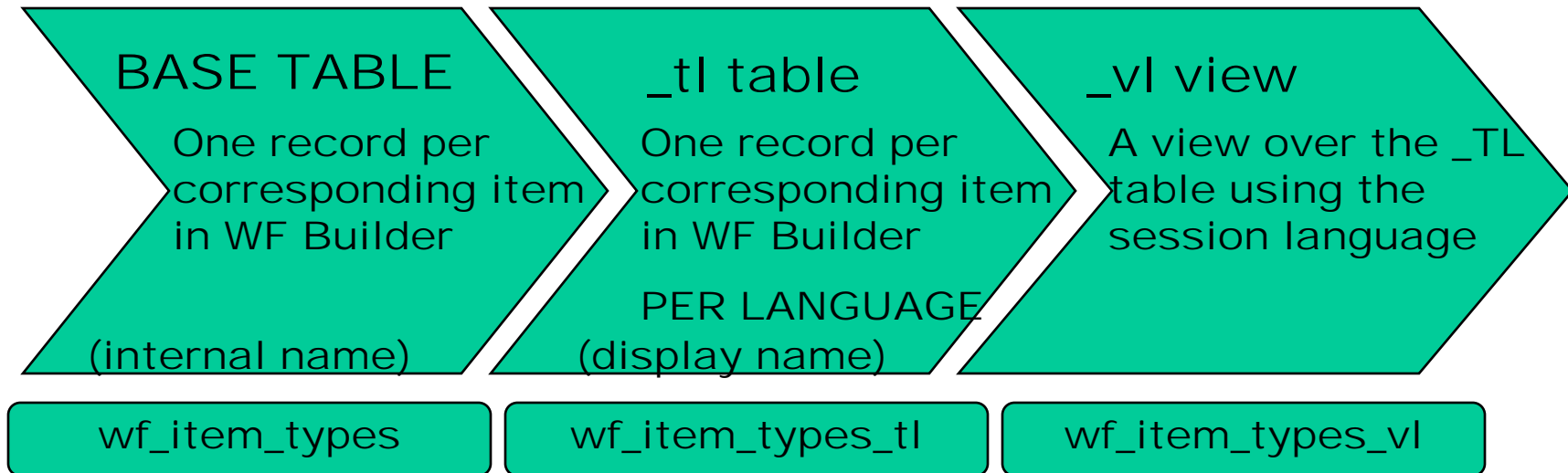
## *Wf\_process\_activities*

- Activity included in process
- Start / end activity
- Instance label
- Instance ID (not depicted)

## *wf\_activitiy\_transitions*

- Transition From activity
- Transition To activity
- Result for Transition (if any)

# Definition Tables – Naming Scheme



```
SQL> select userenv(' LANG' ) from dual
      2 /

USERENV(' LANG' )
-----
US

1 row selected.

SQL>
```

\* Lookups tables follow a slightly different scheme

# Workflow Tables

Definition Tables	Runtime Tables
wf_item_types	wf_items
wf_item_attributes	wf_item_attribute_values
wf_activities	wf_item_activity_statuses
wf_activity_attributes	wf_item_activity_statuses_h
wf_activity_attr_values	wf_notifications
wf_messages	
wf_message_attributes	
wf_process_activities	
wf_activity_transitions	
wf_lookup_types_tl	
wf_lookups_tl	

# wf\_item\_activity\_statuses

- The GPS Trail for vehicles on WF Hwy
- Which activities have been visited?
- What happened at each point?
- wf\_item\_activity\_statuses\_**h** – history

# Section 2

## Delivered Code



**Workflow Scripts and APIs**



# Workflow Scripts

- Where are they?
  - \$FND\_TOP/sql
- Why use them?
  - Perform tasks for which there are no APIs
  - Learn from them
- Be careful!
  - Many scripts modify or delete data
  - Don't run it until you've read it

```

Oracle SQL*Plus
File Edit Search Options Help
09:31:55 SQL> @wfrmall

DANGER *** DANGER *** DANGER *** DANGER *** DANGER *** DANGER
-
This Deletes all workflow data. ALL OF IT.
-
DANGER *** DANGER *** DANGER *** DANGER *** DANGER *** DANGER
-

Deleting from wf_item_activity_statuses_h
    
```

# Workflow Scripts

- Reporting Scripts

- wfstatus.sql
- wfverchk.sql
- wfprotck.sql
- wfbkgchk.sql

- Data Cleanup Scripts

- wfrmitt.sql
- wfrmita.sql
- wfchact.sql
- wfchitt.sql
- wfchacta.sql
- wfchita.sql
- wfchlut.sql
- wfchluc.sql
- wfverupd.sql

STUDY THIS ONE



- See Metalink Note 108185.1

# Workflow APIs

- Public API Packages
  - WF\_ENGINE
  - WF\_CORE
  - WF\_STANDARD
  - WF\_PURGE
  - WF\_NOTIFICATIONS
  - WF\_ITEM

# Some APIs in wf\_engine

- CreateProcess
- StartProcess
- LaunchProcess
- GetItemAttrXXXX
- SetItemAttrXXXX
- GetActivityAttrXXXX
- AddItemAttr
- AssignActivity

# Initiate Processes - APIs

- **Create and Start a process instance**
- **LaunchProcess** combines functionality
- You choose **item\_key** (unique)
- **COMMIT** required (generally)
- **Must have selector function**
  - or use third param  
in **createProcess**  
and **launchProcess** to specify the  
process

```

wf_engine.createProcess
(  itemtype      IN VARCHAR2
,  itemkey       IN VARCHAR2
,  process       IN VARCHAR2
,  user_key      IN VARCHAR2
,  owner_role    IN VARCHAR2);

wf_engine.startProcess
(  itemtype      IN VARCHAR2
,  itemkey       IN VARCHAR2);

-- OR --

wf_engine.launchProcess
(  itemtype      IN VARCHAR2
,  itemkey       IN VARCHAR2
,  process       IN VARCHAR2
,  user_key      IN VARCHAR2
,  owner_role    IN VARCHAR2);
    
```

# Add Item Attribute

Other APIS to ...

- Add an ItemAttribute
- Add an activityAttribute
- Update mutliple attributes
- Get info about attribute setup

# Item Attribute Values: SET and GET

- Read and update attribute values
- Three varieties of each
  - Depending upon datatype of attribute
- "get" has Optional 4<sup>th</sup> parameter
  - ignore\_notfound

wf_engine.setItemAttrText	wf_engine.getItemAttrText
wf_engine.setItemAttrNumber	wf_engine.getItemAttrNumber
wf_engine.setItemAttrDate	wf_engine.getItemAttrDate

```

wf_engine.setItemAttrText
(   itemtype   IN VARCHAR2
,   itemkey    IN VARCHAR2
,   aname      IN VARCHAR2
,   avalue     IN VARCHAR2 );

wf_engine.getItemAttrText
(   itemtype   IN VARCHAR2
,   itemkey    IN VARCHAR2
,   aname      IN VARCHAR2 )
RETURN VARCHAR2;
    
```

- "\_\_\_Text" version works on all of them
- Also, versions to make calls in bulk

wf_engine.setItemAttrTextArray	wf_engine.getItemAttrTextArray
wf_engine.setItemAttrNumberArray	wf_engine.getItemAttrNumberArray
wf_engine.setItemAttrDateArray	wf_engine.getItemAttrDateArray

# Activity Attribute Values: GET

- Read activity attribute values
  - From setup constant or item\_attribute
- No APIs for SET functionality
- Additional Param is instance\_id from WPA
- Three varieties
  - Depending upon datatype of attribute
- "get" has Optional 5<sup>th</sup> parameter
  - ignore\_notfound

```

wf_engine.getItemAttrText
(   itemtype   IN VARCHAR2
,   itemkey    IN VARCHAR2
,   actid      IN NUMBER
,   aname      IN VARCHAR2 )
RETURN VARCHAR2;
    
```

wf_engine.getActivityAttrText
wf_engine.getActivityAttrNumber
wf_engine.getActivityAttrDate

- "\_\_\_Text" version works on all of them
- No bulk (Array) APIs for activity attributes



# Activity Attribute APIs

- Get info about an Activity Attribute
- itemkey is required
- actid is instance\_id from wf\_process\_activities
- atype = Datatype
  - Specific to activity\_id
- subtype
  - Send /Respond
- Format mask
- PLSQL Procedure

```

wf_engine.getActivityAttrInfo
( itemtype      IN VARCHAR2
  , itemkey     IN VARCHAR2
  , actid       IN NUMBER
  , aname       IN VARCHAR2
  , atype       OUT VARCHAR2
  , subtype     OUT VARCHAR2
  , format      OUT VARCHAR2 );
    
```

# Activity Attribute APIs - GET

- Use these APIs to read an activity attribute value from a running process
- PLSQL Functions (Return)
- Three varieties depending upon datatype or attribute
  - getItemAttrText is generic
  - Formats may play havoc
- Optional parameter for ignore\_notfound
  - Default FALSE

```

wf_engine.getActivityAttrText
( itemtype  IN VARCHAR2
, itemkey   IN VARCHAR2
, actid     IN NUMBER
, aname     IN VARCHAR2)
RETURN VARCHAR2;

wf_engine.getActivityAttrNumber
( itemtype  IN VARCHAR2
, itemkey   IN VARCHAR2
, actid     IN NUMBER
, aname     IN VARCHAR2)
RETURN NUMBER;

wf_engine.getActivityAttrDate
( itemtype  IN VARCHAR2
, itemkey   IN VARCHAR2
, actid     IN NUMBER
, aname     IN VARCHAR2)
RETURN DATE;
    
```

# Activity Attribute APIs

- No SET APIs for Activity Attributes
- Activity attributes must be constants or based on item\_attribute values

# Assign Activity

```
wf_engine.AssignActivity
( itemtype      IN VARCHAR2
, itemkey      IN VARCHAR2
, activity      IN VARCHAR2
, performer    IN VARCHAR2 );
```

- Assigns a new performer to an activity
- If already notified, resends notif
- “Activity” - internal name of node
  - Or use instance\_name if necessary

# WF\_ITEM package

- Available APIs
  - SetItemUserKey
  - GetItemUserKey
  - SetItemParent
  - SetItemOwner
  - Item\_exist: Conflicting documentation
    - Metalink Workflow FAQ – Makes mention of it
    - Package specification – Private
- Makes use of cache (session variables) for performance

# WF\_CORE package

- Used for error handling
- Hold and pass session variables
- Hold and pass error message tokens
  - Tie to WF\_RESOURCES
- Procedures
  - GetError
  - Context

# Why use the APIs?

- Only supported method to update values

But even for retrieving data...

- APIs are much more efficient
  - Reduced context switches due to caching
- Oracle maintains the code

# Section 3

## Accidents will happen!



## Error Handling in Workflow



# How WF Handles Errors

- Delivered Error Handler
  - WFERROR item\_type
  - Making your own error process
- Error handling in your PLSQL
  - Anticipating and raising errors
  - WF\_CORE

# How WF Handles Errors

- SAVEPOINT before each activity
- Unhandled Exception in Activity causes ROLLBACK
  - “Unhandled” includes Exception Handler with RAISE
- WF Engine searches for defined error handler process
  - First in Activity
  - Then, in process
  - Finally, in any parent processes
- Executes Handler (ie: runs designated process)

# Defining WF Error Process

- Define:
  - by Process or...
  - by Individual Activity
- In WF Builder
- Properties of Activity
- On “Details” tab

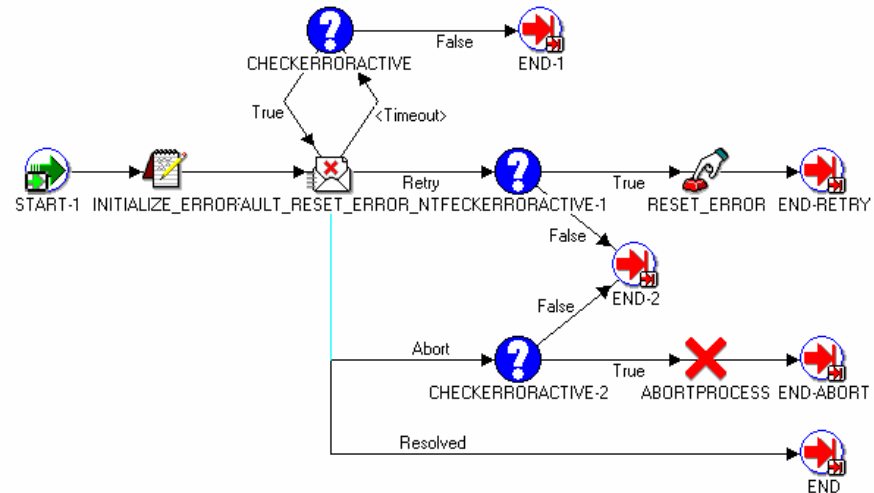
The screenshot shows a dialog box titled "Navigator Control Properties" with a close button (X) in the top right corner. The dialog has four tabs: "Activity", "Details", "Roles", and "Access". The "Details" tab is selected. The dialog contains the following fields:

- Error Item Type:
- Error Process:
- Effective:
- On Revisit:  (with a dropdown arrow)
- Version:

At the bottom of the dialog, there are four buttons: "OK", "Cancel", "Apply", and "Help".

# Delivered Process

- Item\_type  
WFERROR
- Process name  
DEFAULT\_ERROR
- System: Error
- Notifies SYSADMIN
  - Can be overridden
    - Create an attribute called #WF\_ADMINISTRATOR
  - Calls wf\_standard.initializeErrors



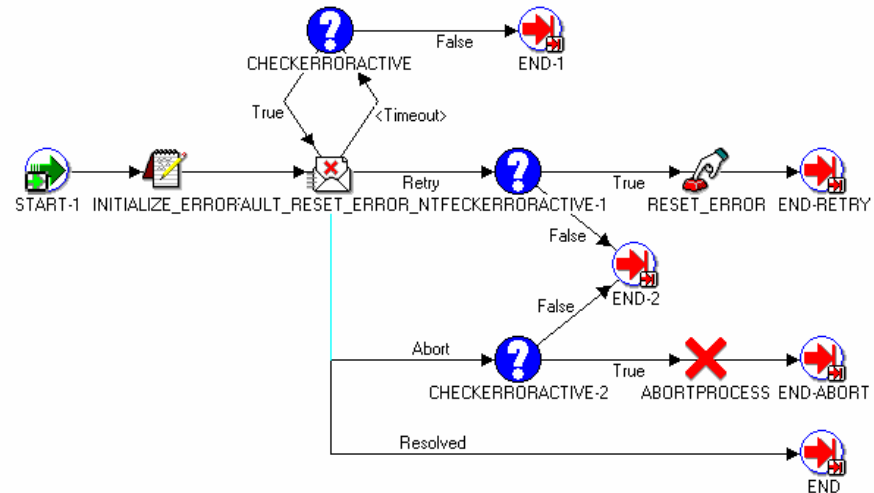
# Delivered Process

- 20 attributes, including...

- Error Item\_type
- Error Item\_key
- Error Activity ID
- Error Message
- Error Name

- Notifies SYSADMIN

- Can be overridden
- Create an attribute called WF\_ADMINISTRATOR
- Handled by wf\_standard.initializeErrors



# Create Your Own Error Process

- Developer need only create those attrs actually needed
- To reference attrs in notification message, must be created up front
  - Use internal\_names
- WF Engine calls `wf_standard.initialize_errors`
- Set up your error process on `item_type`  
AND on individual activities

# Raising Errors from PLSQL

Propagate exceptions that cannot be handled within PLSQL...

- In PLSQL Exception handler
  - Include call to wf\_core.context
    - wf\_core.context(package\_name, pro, itemtype,itemkey,to\_char(actid));
  - RAISE it to WF Engine
    - RAISE statement is considered “Unhandled”

- ResultOut activity status is ‘ERROR’

Allows you to specify a result code, if desired

# PLSQL Exception Handling

- Trap the Exception
- Record the Details
- Re-Raise the Exception

```

PROCEDURE my_wf_proc (i temtype  IN  VARCHAR2
                    , i temkey   IN  VARCHAR2
                    , actid     IN  NUMBER
                    , funcmode  IN  VARCHAR2
                    , resul tOut OUT VARCHAR2 )

```

...

< Executable statements >

```

EXCEPTION WHEN OTHERS THEN

```

```

    wf_core.context ('my_wf_proc' , NULL, i temtype,
                   i temkey, TO_CHAR(actid), funcmode);

```

```

    RAISE;

```

```

END;

```



# WF\_CORE

- wf\_core.context
- Serves as session variable
  - multi-dimensional
- Information about the exception is placed onto the error stack
- RAISE is still “unhandled”
- Oracle’s public APIs will populate WF\_CORE

```

wf_core.context(pkg_name IN VARCHAR2
                , proc_name IN VARCHAR2
                , arg1 IN VARCHAR2 DEFAULT '*none*'
                , arg2 IN VARCHAR2 DEFAULT '*none*'
                , arg3 IN VARCHAR2 DEFAULT '*none*'
                , arg4 IN VARCHAR2 DEFAULT '*none*'
                , arg5 IN VARCHAR2 DEFAULT '*none*'
                , arg6 IN VARCHAR2 DEFAULT '*none*'
                , arg7 IN VARCHAR2 DEFAULT '*none*'
                , arg8 IN VARCHAR2 DEFAULT '*none*'
                , arg9 IN VARCHAR2 DEFAULT '*none*'
                , arg10 IN VARCHAR2 DEFAULT '*none*');
    
```

```

EXCEPTION
WHEN OTHERS THEN
    wf_Core.context( pkg_name => 'THIS_PACKAGE_NAME'
                    , proc_name => 'THIS_PROCEDURE_NAME'
                    , arg1 => itemtype
                    , arg2 => itemkey
                    , arg3 => TO_CHAR(actid)
                    , arg4 => funcmode);

    RAISE;
END;
    
```

# How pkg name and proc name get used in error stack

SQL> /

ITEM_KEY	RESULT	ACTIVITY STACK
1	ERROR	dj_s_make_error.proc_name(DJS, 1, 171717, RUN,ORA-01476: di vi sor is equal to zero) Wf_Engi ne_Uti l . Functi on_Cal l (dj_s_make_error, DJS, 1, 171717, RUN)
2	ERROR	.proc_name(DJS, 2, 171717, RUN,ORA-01476: di vi sor is equal to zero) Wf_Engi ne_Uti l . Functi on_Cal l (dj_s_make_error, DJS, 2, 171717, RUN)
3	ERROR	dj_s_make_error.(DJS, 3, 171717, RUN,ORA-01476: di vi sor is equal to zero) Wf_Engi ne_Uti l . Functi on_Cal l (dj_s_make_error, DJS, 3, 171717, RUN)

3 rows selected.

# Section 4

## Function Activities



## Coding for WF Function Activities

# Workflow Function Activities

- Often just called “functions”
- Implemented by procedures stored in the database
  - PLSQL
  - Java
- Usually return a “result”
- A Workflow Function is implemented by a PLSQL Procedure
  - In java it is a method

# Creating Function Activities

## Setup Steps:

- Create function activity within WF Builder
- Write underlying code to implement activity
- Ensure that the two are in synch
  - Function / procedure name
  - Result\_type

# Creating Function Activities

- **Function Name**
  - Name of the PLSQL procedure which implements the function
    - Usually: package.procedure
- **Result Type**
  - Ties to WF Lookup Type
  - Guides transition

The screenshot shows the 'Navigator Control Properties' dialog box with the 'Details' tab selected. The fields are as follows:

Field	Value
Internal Name	DJS_CALL
Display Name	Dans PLSQL Procedure Call
Description	
Icon	FUNCTION.ICD
Function Name	djs_plsql_values
Function Type	PL/SQL
Result Type	Yes/No
Cost	0.00

Buttons: Browse, Edit, OK, Cancel, Apply, Help

# PLSQL Approach

- Receive information from calling WF
  - Four inbound parameters
- Execute processing
  - Use Oracle's APIs
  - Query Database tables
  - Perform DML
  - Anything you would do in any PLSQL Program  
... except COMMIT!
- Pass information back to the calling node
  - Pass a value in the ResultOut parameter
  - Update values of item attributes

# PLSQL Call Signature

- Implements WF Function activities
- PLSQL Procedure
- Name of procedure matches value from “Function Name” in WF Builder
  - Or name of procedure within a package

```

PROCEDURE xxxx
(   itemtype           IN  VARCHAR2
,   itemkey            IN  VARCHAR2
,   actid              IN  NUMBER
,   funcmode           IN  VARCHAR2
,   resultout          OUT VARCHAR2 )
    
```



# PLSQL Call Signature

- Parameters:

- INBOUND

- itemType
- itemKey
- actid
- funcmode

- OUTBOUND

- resultOut

- Do I have to spell them this way?

```

PROCEDURE xxxx
(   itemType           IN  VARCHAR2
,   itemkey            IN  VARCHAR2
,   actid              IN  NUMBER
,   funcmode           IN  VARCHAR2
,   resultout          OUT VARCHAR2 )
    
```

# itemType/itemKey

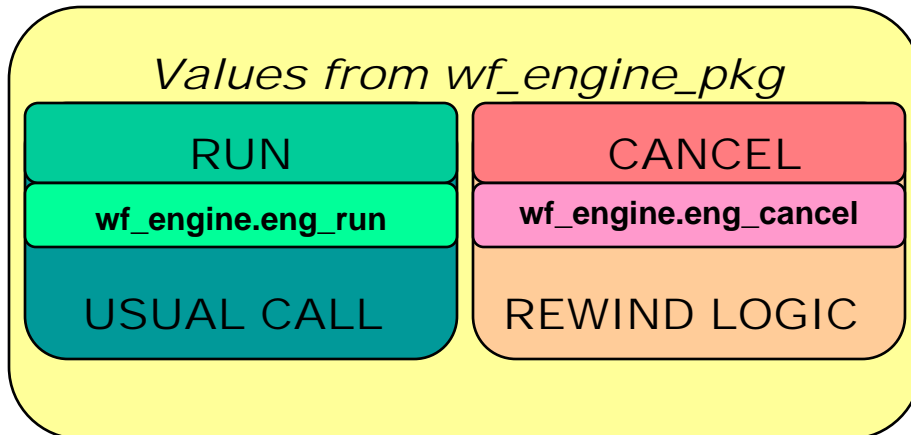
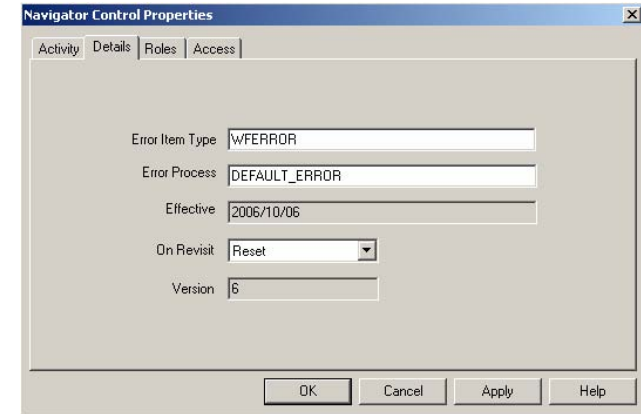
- Unique identifiers of a process instance
- Often, these are the only inbound parameters needed
  - API calls to get and set attribute values, eg.
  - wf\_standard.noop doesn't use any of the parameters

# actid

- Activity\_id
  - Instance\_id of calling node in wf\_process\_activities table
  - Ties to wf\_transitions
  - Joins to wf\_activity\_attr\_values
- Needed for Activity Attribute APIs
- Useful when you need to:
  - Determine which activity node made call
  - Determine name of calling process
  - Version of calling process

# funcmode

- Function mode
- Frequently spelled “funmode”
- Constants from wf\_engine pkg



<i>On revisit vs Funcmode</i>	
Reset	CANCEL <i>then</i> RUN
Loop	RUN
Ignore	<i>Subsequent calls ignored (OR logic)</i>

# resultOut

# 1=2

A Single  
Parameter

Workflow  
Engine  
parses to  
two  
values

Values are stored in  
wf\_item\_activity\_statuses

resultOut

Activity status

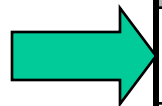
Activity\_status

Activity result

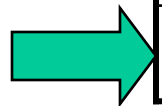
Activity\_result\_code

# resultOut – Activity Status

## *Constants from the wf\_engine package*



eng_completed	'COMPLETE'	Normal completion
eng_active	'ACTIVE'	Activity Running
eng_waiting	'WAITING'	Activity Waiting to Run
eng_notified	'NOTIFIED'	Notification Open
eng_suspended	'SUSPEND'	Activity Suspended
eng_deferred	'DEFERRED'	Activity Deferred
eng_error	'ERROR'	Complete with Error



- You'll probably only use: COMPLETE and ERROR
- wf\_engine.eng\_error will raise an error in process

## resultOut – Result\_Code

- Must be a valid member of the lookup defined as the result for the activity
  - Validity NOT enforced by Oracle
  - Match IS case sensitive – Upper case always
- Used by wf\_transitions
- If activity result not in lookup, then process will become #STUCK

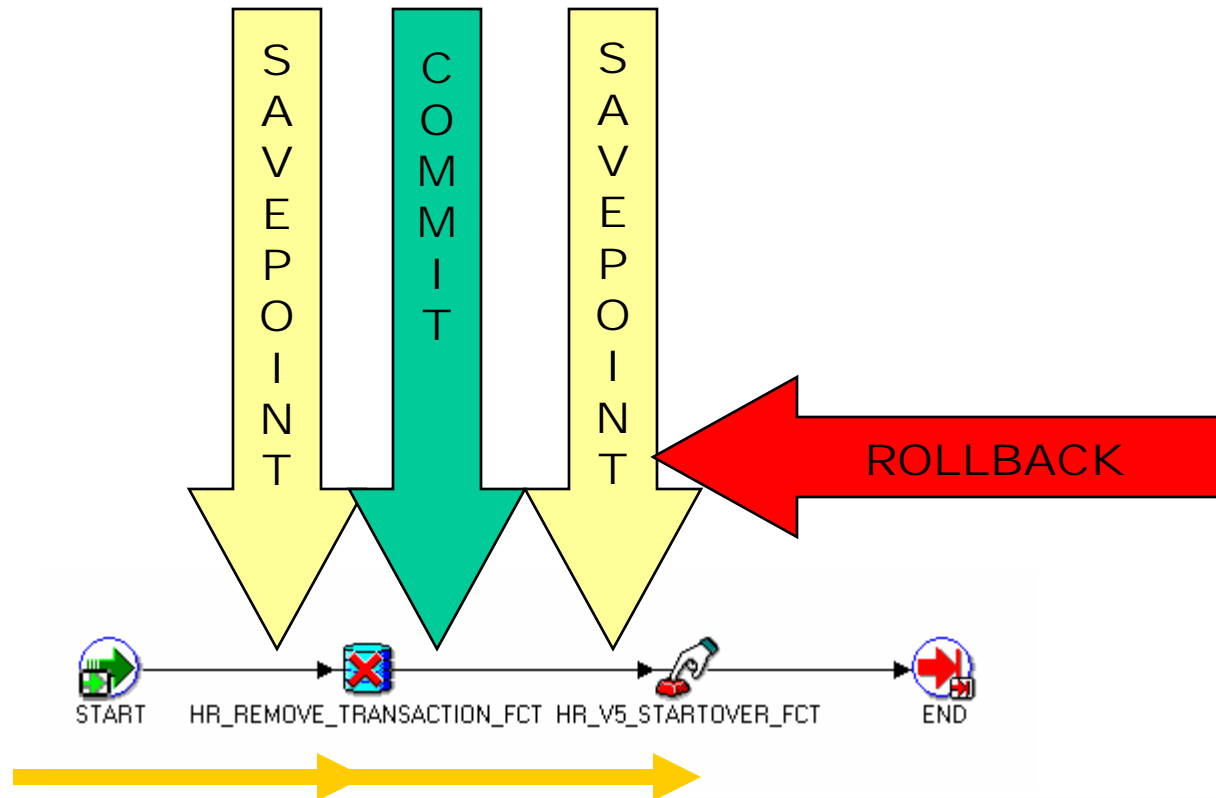
# Parsing resultOut

- Expected format
  - PROCESS\_STATUS:RESULT\_VALUE
- If not in correct format, wf\_engine still attempts to parse
  - Parse algorithm:
    1. Look for Colon :
    2. Check ResultOut vs known possible Status Codes
    3. Assume that ResultOut is Activity Result
- If no status\_code, default is COMPLETE



# When Do I COMMIT?

- Savepoint / Commit Illustration



# Other considerations

- Versioning
- Parameter Name Spelling
- Commit
  - Savepoint and Rollback OK
  - DDL = Implicit Commit

# Section 5

## Shut Up and Drive!



Writing Code to  
Implement Function Activities

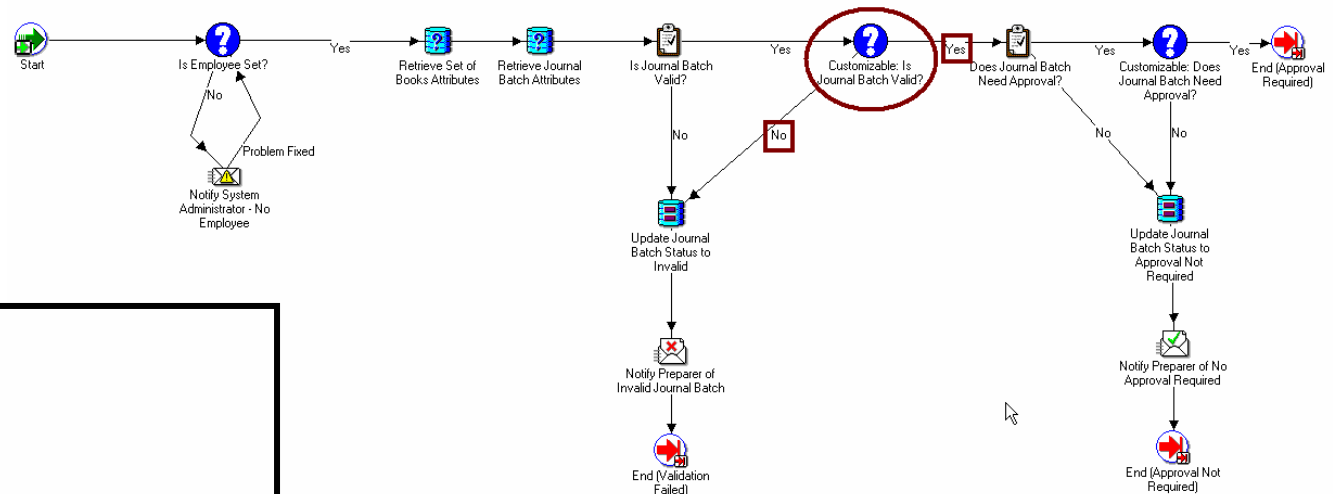
# Three Demonstrations with Function Activities

- GLBATCH
- Repeat Result
- Set Wait Date

# Customizing GLBATCH

- Download Package `gl_wf_customization_pkg`
  - `glwfcusb.pls`
  - Spool from `dba_source`
- Identify procedure(s) to customize
- Make changes to desired procedure
- Compile package with edits
- Don't forget to include the procedures that you did not customize!

# Customizing GLBATCH



**Journal Batch**

- + Attributes
- + Processes
- + Notifications
- Functions
  - Approve Journal Batch
  - Customizable: Does Journal Batch Need Approval?
  - Customizable: Is Journal Batch Valid?
  - Customizable: Is Preparer Authorized to Approve?
  - Customizable: Verify Authority
  - Does Journal Batch Need Approval?
  - Find Approver
  - Get The Manager of the Approver
  - Is Approver the Direct Manager?

# Customizing GLBATCH

```

1 PACKAGE BODY APPS.GL_WF_CUSTOMIZATION_PKG AS
2
3 PROCEDURE is_j_e_valid(i temtype      IN VARCHAR2,
4                       itemkey       IN VARCHAR2,
5                       actid         IN NUMBER,
6                       funcmode     IN VARCHAR2,
7                       result       OUT NOCOPY VARCHAR2 ) IS
8 BEGIN
9   IF ( funcmode = 'RUN' ) THEN
10    -- Additional code can be added here.
11    -- COMPLETE: Y (Workflow transition branch "Yes") indicates that
12    -- batch is valid.
13    -- COMPLETE: N (Workflow transition branch "No") indicates that
14    -- batch is not valid.
15    result := 'COMPLETE: Y';
16  ELSIF (funcmode = 'CANCEL') THEN
17    NULL;
18  END IF;
19 END is_j_e_valid;
20
21
22 PROCEDURE does_j_e_need_approval . . .
23
24 PROCEDURE can_preparer_approve . . .
25
26 PROCEDURE verify_authority . . .
27
28 END GL_WF_CUSTOMIZATION_PKG;

```



```

PROCEDURE is_j_e_valid(i temtype IN VARCHAR2,
                       itemkey IN VARCHAR2,
                       actid IN NUMBER,
                       funcmode IN VARCHAR2,
                       result OUT NOCOPY VARCHAR2 ) IS
BEGIN
  IF ( funcmode = 'RUN' ) THEN
    -- Additional code can be added here.
    -- COMPLETE: Y (Workflow transition branch "Yes") indicates
    -- that the journal batch is valid.
    -- COMPLETE: N (Workflow transition branch "No") indicates
    -- that the journal batch is not valid.
    result := 'COMPLETE: Y';
  ELSIF (funcmode = 'CANCEL') THEN
    NULL;
  END IF;
END is_j_e_valid;

```

# Customizing GLBATCH

```

3 PROCEDURE is_j_e_val id(i temtype      IN VARCHAR2,
4                        i temkey       IN VARCHAR2,
5                        act id         IN NUMBER,
6                        funcmode      IN VARCHAR2,
7                        resul t       OUT NOCOPY VARCHAR2 )
8 IS
9
10     l_batch_id         gl _j_e_batches.j_e_batch_id%TYPE;
11     l_bool ean_val id ty_check    BOOLEAN;
12
13 BEGIN
14     IF ( funcmode = ' RUN' ) THEN
15
16
17         l_batch_id := wf_engl ne.getltemAttrNumber ( i temtype, i temkey, ' BATCH_ID'
18
19
20         l_bool ean_val id ty_check := my_custom_val id ty_check (l_batch_id);
21
22         IF l_bool ean_val id ty_check THEN
23             resul t := ' COMPLETE: Y' ;
24         ELSE
25             resul t := ' COMPLETE: N' ;
26         END IF;
27
28     ELSIF (funcmode = ' CANCEL' ) THEN
29         NULL;
30     END IF;
31 END is_j_e_val id;

```

```

PROCEDURE is_j_e_val id(i temtype IN VARCHAR2,
                        i temkey IN VARCHAR2,
                        act id IN NUMBER,
                        funcmode IN VARCHAR2,

```

```

l_bool ean_val id ty_check :=
my_custom_val id ty_check(l_batch_id);

```

```

IF l_bool ean_val id ty_check THEN
    resul t := ' COMPLETE: Y' ;
ELSE
    resul t := ' COMPLETE: N' ;
END IF;

```

```

ELSIF (funcmode = ' CANCEL' ) THEN
    NULL;
END IF;

```

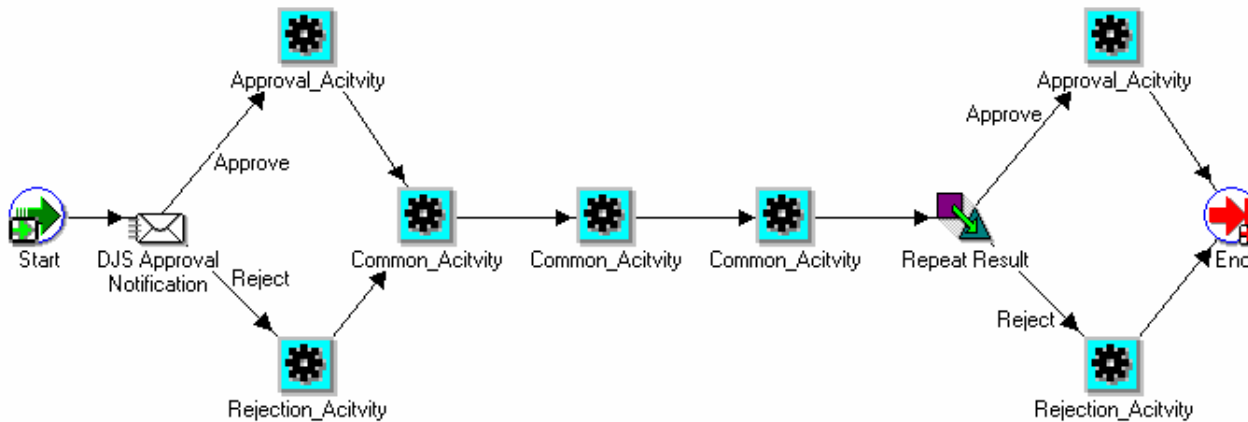
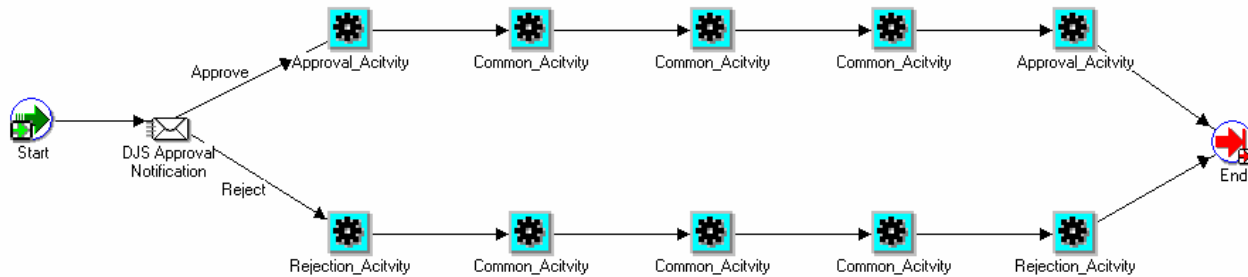
```

END is_j_e_val id;

```



# Repeat Result

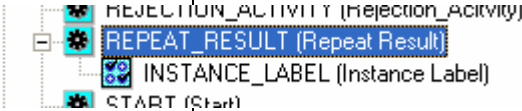


# Repeat Result

```

23
24 -- Read the value contained in the attribute
25 l_instance_label_attr := wf_engine.getActivityAttrText
26 (
27   itemtype => itemtype
28   , itemkey => itemkey
29   , actid   => actid
30   , aname   => 'INSTANCE_LABEL'
31   , ignore_notfound => TRUE
32 );
33
34 -- Get information about the process that the activity calling this is a part of
35 SELECT wpa.process_item_type
36        , wpa.process_name
37        , wpa.process_version
38 INTO l_process_item_type
39        , l_process_name
40        , l_process_version
41 FROM wf_process_activities wpa
42 WHERE wpa.instance_id = actid;
43
44 BEGIN
45
46 -- Then, get the instance_id of the activity whose value should be repeated
47 SELECT wpa.instance_id
48 INTO l_instance_id
49 FROM wf_process_activities wpa
50 WHERE wpa.process_item_type = l_process_item_type
51       AND wpa.process_name   = l_process_name
52       AND wpa.process_version = l_process_version
53       AND wpa.instance_label = l_instance_label_attr;
54
55 EXCEPTION
56 WHEN no_data_found THEN
57 -- Either the activity attribute does not exist or it contained a string that
58 -- is not an instance label in this process
59 RAISE e_activity_not_in_process;
60
61 END;
62
63 BEGIN
64 SELECT activity_result_code
65 INTO l_activity_result_code
66 FROM wf_item_activities
67 WHERE process_activity = l_instance_id
68       AND item_type    = itemtype
69       AND item_key     = itemkey
70       AND activity_status = wf_engine.completed;
71
72 EXCEPTION
73 WHEN no_data_found THEN
74 RAISE e_no_completed_activity_found;
75
76 END;
77
78 resultOut := wf_engine.completed || ':' || l_activity_result_code;
79
80 END IF; -- funcmode = wf_engine.eng_run

```



Navigator Control Properties

Function | Details | Roles | Access | Node | Node Attributes

Attribute

Name: Instance Label

Type: Constant

Value: DJS\_APPROVAL\_NTF

Name	Value Type	Value	Type	Description
Instance Label	Constant	DJS_APPROVAL_NTF	Text	Instance Label of ...

OK Cancel Apply Help

```

-- Read the value contained in the attribute
l_instance_label_attr :=
wf_engine.getActivityAttrText
(
  itemtype => itemtype
  , itemkey => itemkey
  , actid   => actid
  , aname   => 'INSTANCE_LABEL'
  , ignore_notfound => TRUE
);

```

# Repeat Result

```

23
24 -- Read the value contained in the attribute
25 l_instance_label_attr := wf_engine.getActivityAttrText
26 ( l_itemtype => l_itemtype
27 , l_itemkey => l_itemkey
28 , actid => actid
29 , aname => 'INSTANCE_LABEL'
30 , ignore_notfound => TRUE );
31
32 -- Get information about the process that the activity calling this is a part of
33 SELECT wpa.process_item_type
34 , wpa.process_name
35 , wpa.process_version
36 INTO l_process_item_type
37 , l_process_name
38 , l_process_version
39 FROM wf_process_activities wpa
40 WHERE wpa.instance_id = actid;
41
42 BEGIN
43
44 -- Then, get the instance_id of the activity whose value should be repeated
45 SELECT wpa.instance_id
46 INTO l_instance_id
47 FROM wf_process_activities wpa
48 WHERE wpa.process_item_type = l_process_item_type
49 AND wpa.process_name = l_process_name
50 AND wpa.process_version = l_process_version
51 AND wpa.instance_label = l_instance_label_attr;
52
53 EXCEPTION
54 WHEN no_data_found THEN
55 -- Either the activity attribute attribute does not exist or it contained a string that
56 -- is not an instance label in this process
57
58 RAISE e_activity_not_in_process;
59 END;
60
61 BEGIN
62 SELECT activity_result_code
63 INTO l_activity_result_code
64 FROM wf_item_activities
65 WHERE process_activity = l_instance_id
66 AND l_itemtype = l_itemtype
67 AND l_itemkey = l_itemkey
68 AND activity_status = wf_engine.eng_completed;
69
70 EXCEPTION
71 WHEN no_data_found THEN
72
73 RAISE e_no_completed_activity_found;
74 END;
75
76 resultOut := wf_engine.eng_completed || ':' || l_activity_result_code;
77
78 END IF; -- funcmode = wf_engine.eng_run
79

```

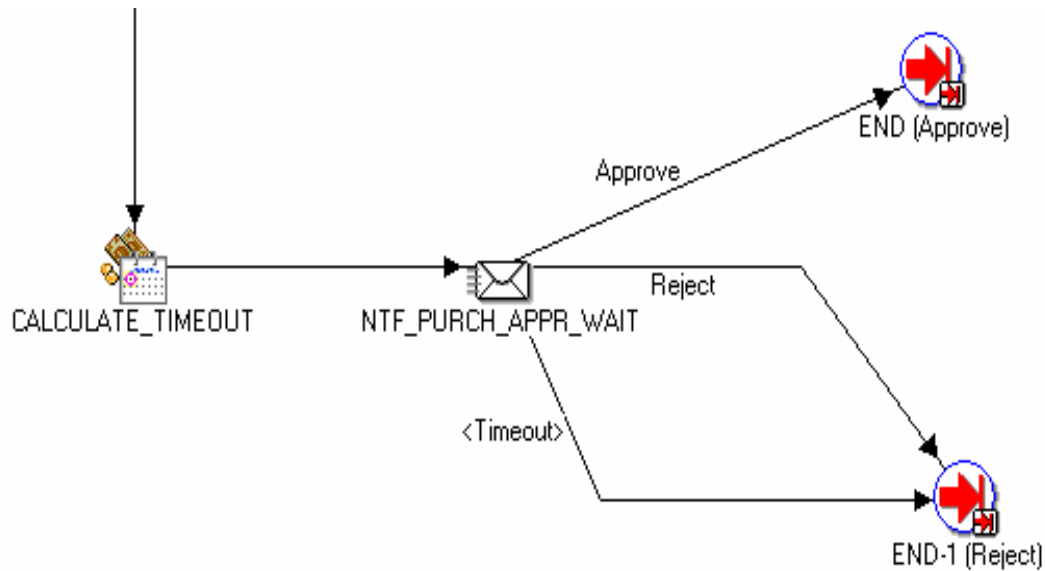
```

-- Get information about the process that
-- the activity calling this is a part of
BEFORE SELECT wpa.process_item_type
, wpa.process_name
, wpa.process_version
WHEN INTO l_process_item_type
SELECT l_process_name
IN l_process_version
FROM wf_process_activities wpa
WHERE wpa.instance_id = actid;
l_instance_label_attr
AND wpa.process_name = l_process_name
AND wpa.process_version =
l_process_version
AND wpa.instance_label =
l_instance_label_attr;

EXCEPTION
WHEN no_data_found THEN
-- Either the activity attribute attribute
does not exist or it contained a string that
-- is not an instance label in this process
RAISE e_activity_not_in_process;
END;

```

# Calculate Timeout Procedure



# Set Wait Date Procedure

```

1 PROCEDURE setWaitDate
2 ( itemtype      IN VARCHAR2
3 , itemkey       IN VARCHAR2
4 , actid         IN NUMBER
5 , funcmode      IN VARCHAR2
6 , resultout     OUT VARCHAR2 )
7
8 IS
9   l_ProcLaunchDate DATE;
10  l_OneMonthLater   DATE;
11 BEGIN
12   -- Get the process launch date for the item_type and item_key that
13   -- were passed in. Query the wf_items table directly.
14   SELECT wl.begin_date
15     INTO l_ProcLaunchDate
16    FROM wf_items wl
17   WHERE wl.item_type = itemtype
18         AND wl.item_key = itemkey;
19
20   -- Calculate the one month later date
21   l_OneMonthLater := TRUNC(ADD_MONTHS(l_ProcLaunchDate, 1));
22
23   -- make allowance for weekends
24   IF TO_CHAR(l_OneMonthLater, 'D') = 7 THEN -- Saturday
25     l_OneMonthLater := l_OneMonthLater +2;
26   ELSIF TO_CHAR(l_OneMonthLater, 'D') = 1 THEN -- Sunday
27     l_OneMonthLater := l_OneMonthLater +1;
28   END IF;
29
30   -- Write the calculated value to the attribute 'WaitUntilDate'
31   setItemAttrDate (
32     itemtype      => itemtype
33 , itemkey         => itemkey
34 , aname           => 'WAITUNTILDATE'
35 , avalue         => l_OneMonthLater);
36
37   -- Mark the activity status COMPLETE. No result code passed back
38   resultout := wf_engine.eng_completed;
39
40 EXCEPTION
41 WHEN OTHERS THEN
42   wf_core.context ('djs_wf_demo', 'setWaitDate'
43     , itemtype, itemkey, to_char(actid), funcmode
44     , substr(sqlerrm, 1, 100) )
45   -- Mark the activity status ERROR.
46   resultout := wf_engine.eng_error;
47
48 END;
```

```

PROCEDURE setWaitDate
( itemtype      IN VARCHAR2
, itemkey       IN VARCHAR2
, actid         IN NUMBER
, funcmode      IN VARCHAR2
, resultout     OUT VARCHAR2 )
```

**BEGIN**

```

-- Calculate the one month later date at 5:00 pm
-- Mark the activity status COMPLETE.
-- No result code passed back.
resultout := wf_engine.eng_completed;
```

```

END IF; -- if funcmode = wf_engine.eng_run
```

**EXCEPTION**

**WHEN OTHERS THEN**

```

-- Mark the activity status ERROR.
wf_core.context (itemtype, itemkey, TO_CHAR(actid), funcmode);
```

```

resultout := wf_engine.eng_error;
```

**END;**

# Section 6

## Other code



**Reset Preferences - A Concurrent Program  
Selector Functions  
Document Type Attributes**

# Reset Preferences

- Concurrent program
- Designed to overcome flaw in mailer logic
  - Failed notif results in a user pref of 'DISABLED'
- Finds all users with individual notif pref
  - Stored in fnd\_user\_preferences
- Removes those records and synchronizes
  - Two API calls
    - fnd\_preference.remove -> Private
    - fnd\_user\_pkg.user\_synch -> Public
  - Approved by Oracle support in OUR installation

# Reset Preferences

BEGIN

FOR prefs\_rec IN (SELECT \*

FROM appl sys. fnd\_user\_preferences  
WHERE preference\_name = 'MAILTYPE'  
AND module\_name = 'WF'  
AND user\_name != '-WF\_DEFAULT-'

) LOOP

BEGIN

v\_savepoint\_name := prefs\_rec.user\_name;  
SAVEPOINT I\_savepoint\_name;

I\_err\_loc := 'API calls on user ' || prefs\_rec.user\_name;

fnd\_preference.remove  
( p\_user\_name => prefs\_rec.user\_name  
, p\_module\_name => prefs\_rec.module\_name  
, p\_pref\_name => prefs\_rec.preference\_name  
);

fnd\_user\_pkg.user\_synch(prefs\_rec.user\_name);

I\_rec\_count := I\_rec\_count + 1;

```

1 PROCEDURE reset_notification_prefs
2   ( p_errbuff   OUT VARCHAR2
3   , p_retcode   OUT VARCHAR2)
4
5 AS
6
7   l_rec_count   NUMBER      :=0;
8   l_err_count   NUMBER      :=0;
9
10  l_savepoint_name VARCHAR2(25);
11  l_err_loc      VARCHAR2(100);
12
13 BEGIN
14
15  FOR prefs_rec IN (SELECT *
16                    FROM appl sys. fnd_user_preferences
17                    WHERE preference_name = 'MAILTYPE'
18                    AND module_name = 'WF'
19                    AND user_name != '-WF_DEFAULT-')
20  ) LOOP
21
22    BEGIN
23
24      v_savepoint_name := prefs_rec.user_name;
25
26      SAVEPOINT I_savepoint_name;
27
28      l_err_loc := 'API calls on user ' || prefs_rec.user_name;
29
30      fnd_preference.remove ( p_user_name => prefs_rec.user_name
31                            , p_module_name => prefs_rec.module_name
32                            , p_pref_name => prefs_rec.preference_name);
33
34      fnd_user_pkg.user_synch(prefs_rec.user_name);
35
36      l_rec_count := l_rec_count + 1;
37
38    EXCEPTION
39      WHEN OTHERS THEN
40
41        ROLLBACK TO I_savepoint_name;
42
43        l_err_count := l_err_count + 1;
44        fnd_file.put_line(fnd_file.log, 'Error at ' || l_err_loc || ' ');
45
46    END;
47  END LOOP;
48
49  COMMIT;
50
51  fnd_file.put_line(fnd_file.log, l_rec_count || ' user preference records ');
52
53
54  IF l_err_count = 0 THEN
55
56    p_errbuff := 'SUCCESSFUL COMPLETION. ' || l_rec_count || ' user preferences reset';
57    p_retcode := 0;
58
59  ELSE
60

```



# Selector Functions

- PLSQL Procedure
- Tells WF engine which process to launch when none is specified.
- Optional property of Item Type
- Procedure returns name of process
  - In “resultOut” parameter
- Same procedure used for callback
  - “command” parameter is ‘RUN’ for selector
- Example: generic\_selector
  - Returns name of only process in the item type

```

PROCEDURE xxxx
(
    item_type      IN      VARCHAR2
  , item_key       IN      VARCHAR2
  , activity_id    IN      NUMBER
  , command        IN      VARCHAR2
  , result_out     IN OUT  VARCHAR2 )
    
```

Define  
Selector Function  
Here

# Selector Function

```

1 PROCEDURE generic_selector
2 ( item_type IN VARCHAR2
3 , item_key IN VARCHAR2
4 , activity_id IN NUMBER
5 , command IN VARCHAR2
6 , result_out IN OUT VARCHAR2 )
7 IS
8
9 l_process_name wf_activities.name%TYPE;
10 l_process_count PLS_INTEGER := 0;
11
12 BEGIN
13 IF command = wf_engine.eng_run THEN
14
15     FOR proc_rec IN (SELECT name
16                     FROM wf_activities wa
17                     WHERE wa.item_type = item_type
18                       AND wa.type = 'PROCESS'
19                       AND runnable_flag = 'Y'
20                       AND SYSDATE BETWEEN wa.begin_date
21                       AND NVL(end_date, SYSDATE)
22                     ) LOOP
23
24         l_process_name := proc_rec.name;
25         l_process_count := l_process_count + 1;
26
27     END LOOP;
28
29     IF l_process_count = 1 THEN
30         -- Only one process in this item_type, so return its name
31
32         result_out := l_process_name;
33
34     ELSE
35
36         result_out := NULL;
37
38     END IF;
39
40 END IF;
41
42 END generic_selector;
```

```

PROCEDURE generic_selector
( item_type IN VARCHAR2
FOR proc_rec IN (SELECT name
                  FROM wf_activities wa
                  WHERE wa.item_type = item_type
                    AND wa.type = 'PROCESS'
                    AND runnable_flag = 'Y'
                    AND SYSDATE BETWEEN wa.begin_date
                    AND NVL(end_date, SYSDATE)
                  ) LOOP
    l_process_name := proc_rec.name;
    l_process_count := l_process_count + 1;
END LOOP;

IF l_process_count = 1 THEN
-- Only one process in this item_type, so return its name
result_out := l_process_name;
ELSE

result_out := NULL;
END IF;
```

# Why Use a Document Type Attribute?

## “Regular” attributes

- TEXT, NUMBER, DATE
- Stored in DB table
  - wf\_item\_attribute\_values
  - wf\_activity\_attr\_values
- Limited to 4000 chars
- <HTML> markup not permitted

## DOCUMENT attributes

- Populated by code
- Derived at runtime
- Virtually unlimited length
  - PLSQL VARCHAR2
  - PLSQL CLOB
- Ideal for <HTML> code with tables of unpredictable length

```
HTML Body

Here is a line from the Ordinary Attribute

Do <FONT COLOR="RED">NOT</FONT> PASS <B>GO</B> AND DO
<U>NOT</U> COLLECT $200

-----

And here is a line from the Document Attribute

Do NOT PASS GO AND DO NOT COLLECT $200

-----

+++ HTML Body +++
```

# Document Attribute Setup

1. Create a Document-type attribute for a message
2. Define a default value for the message attribute
  - Point to procedure as defined in step (4)
3. Include the attribute in the new message

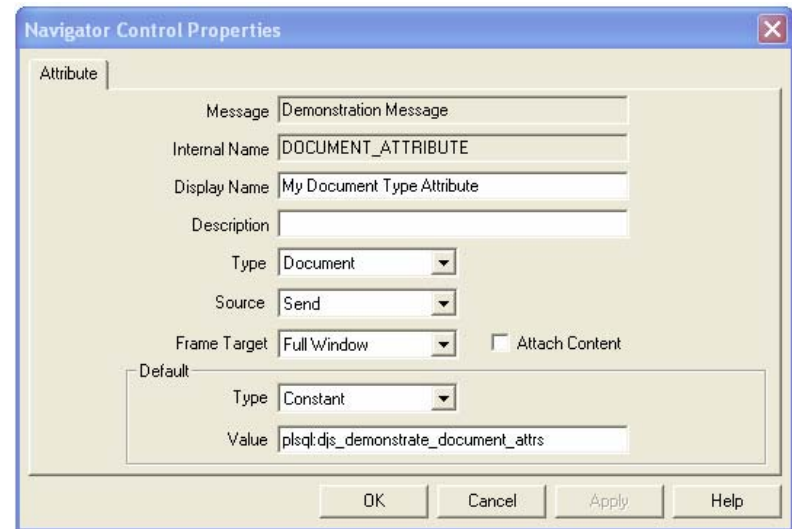
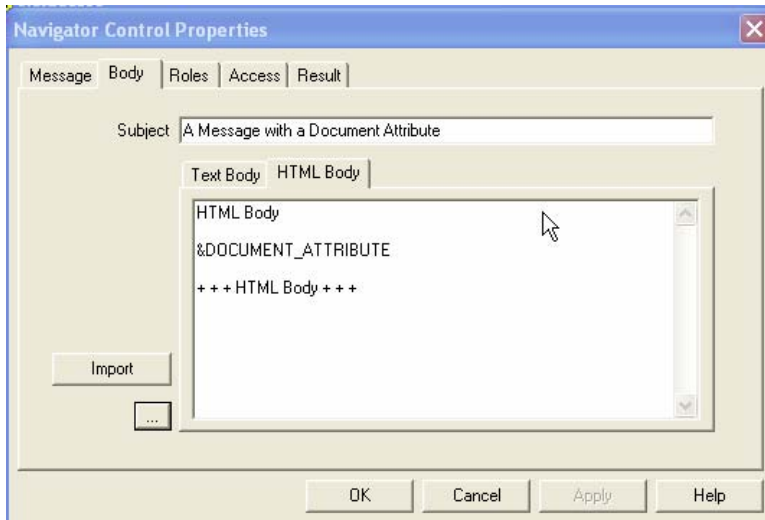
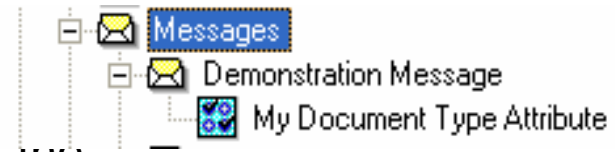
*Steps  
Performed in  
Workflow  
Builder*

4. Write a procedure to be called by the Workflow Engine to populate your attribute and compile

*Step  
Performed in  
Database*

# Document Attribute Setup

- Create a message attribute
  - Type: Document
- Set the attribute's default value to the name of the PLSQL procedure to be called:
  - pl sql : my\_package\_name. my\_proc\_name
  - pl sql cl ob: my\_package\_name. my\_proc\_name
- Include the attribute in message text



# Write a PLSQL procedure

```

PROCEDURE your_custom_procedure
(
  document_id          IN      VARCHAR2
,
  display_type        IN      VARCHAR2
,
  document            IN OUT NOCOPY VARCHAR2
,
  document_type       IN OUT NOCOPY VARCHAR2);

```

- document\_id: allows developer to pass in info
- display\_type: contains the notification style
- document: assign the entire stream of text to be contained in attribute
  - Can be VARCHAR2 or CLOB
  - IN OUT parameter
- document\_type: MIME type for output

# Document Attribute Code

```

1 PROCEDURE djs_demonstrate_document_attr
2   ( document_id          IN   VARCHAR2
3   , display_type        IN   VARCHAR2
4   , document            IN OUT NOCOPY VARCHAR2
5   , document_type       IN OUT NOCOPY VARCHAR2
6 IS
7
8   l_document_html VARCHAR2(32747);
9
10 BEGIN
11
12   l_document_html :=
13     '<H1>These are the values in the<BR>WF Standard
14     || '<BR>Comparison Lookup Type: </H1>'
15     || '<TABLE BORDER="1">';
16
17   FOR lookup_rec IN ( SELECT lookup_code, meaning
18                       FROM wf_lookups
19                       WHERE lookup_type = 'WFSTD_COMPARISON' ) LOOP
20
21     l_document_html := l_document_html
22       || '<TR><TD>' || lookup_rec.lookup_code
23       || '</TD><TD>' || lookup_rec.meaning
24       || '</TD></TR>';
25
26   END LOOP;
27
28   l_document_html := l_document_html
29     || '</TABLE>';
30
31   document := document
32     || l_document_html;
33
34 END djs_demonstrate_document_attr;

```

```

PROCEDURE djs_demonstrate_document_attr
l_document_html :=
  '<H1>These are the values in the<BR>WF Standard'
  || '<BR>Comparison Lookup Type: </H1>'
  || '<TABLE BORDER="1">';

FOR lookup_rec IN ( SELECT lookup_code, meaning
                    FROM wf_lookups
                    WHERE lookup_type = 'WFSTD_COMPARISON'
                    ) LOOP

  l_document_html := l_document_html
    || '<TR><TD>' || lookup_rec.lookup_code
    || '</TD><TD>' || lookup_rec.meaning
    || '</TD></TR>';

END LOOP;

l_document_html := l_document_html
  || '</TABLE>';

document := document
  || l_document_html;

```

# Document Attribute Code

```

1 PROCEDURE djs_demonstrate_document_attr
2   ( document_id          IN   VARCHAR2
3   , display_type        IN   VARCHAR2
4   , document            IN OUT NOCOPY VARCHAR2
5   , document_type       IN OUT NOCOPY VARCHAR2
6 IS
7
8   l_document_html VARCHAR2(32747);
9
10 BEGIN
11
12   l_document_html :=
13     '<H1>These are the values in the<BR>WF Standard
14     || '<BR>Comparison Lookup Type: </H1>'
15     || '<TABLE BORDER="1">';
16
17   FOR lookup_rec IN ( SELECT lookup_code, meaning
18                       FROM wf_lookups
19                       WHERE lookup_type = 'WFSTD_COMPARISON' ) LOOP
20
21     l_document_html := l_document_html
22       || '<TR><TD>' || lookup_rec.lookup_code
23       || '</TD><TD>' || lookup_rec.meaning
24       || '</TD></TR>';
25
26   END LOOP;
27
28   l_document_html := l_document_html
29     || '</TABLE>';
30
31   document := document
32     || l_document_html;
33
34 END djs_demonstrate_document_attr;

```

```

PROCEDURE djs_demonstrate_document_attr
l_document_html :=
  '<H1>These are the values in the<BR>WF Standard'
  || '<BR>Comparison Lookup Type: </H1>'
  || '<TABLE BORDER="1">';

FOR lookup_rec IN ( SELECT lookup_code, meaning
                    FROM wf_lookups
                    WHERE lookup_type = 'WFSTD_COMPARISON'
                    ) LOOP

  l_document_html := l_document_html
    || '<TR><TD>' || lookup_rec.lookup_code
    || '</TD><TD>' || lookup_rec.meaning
    || '</TD></TR>';

END LOOP;

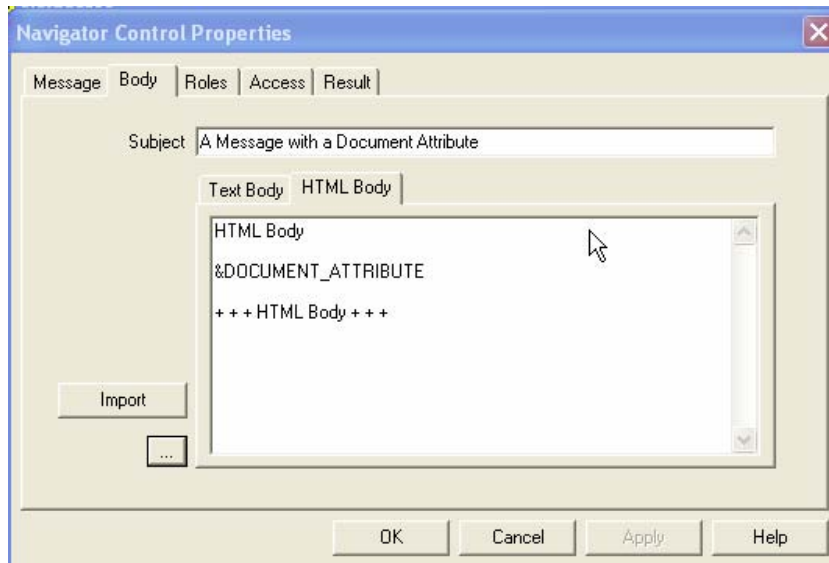
l_document_html := l_document_html
  || '</TABLE>';

document := document
  || l_document_html;

```



# Voilà!



HTML Body

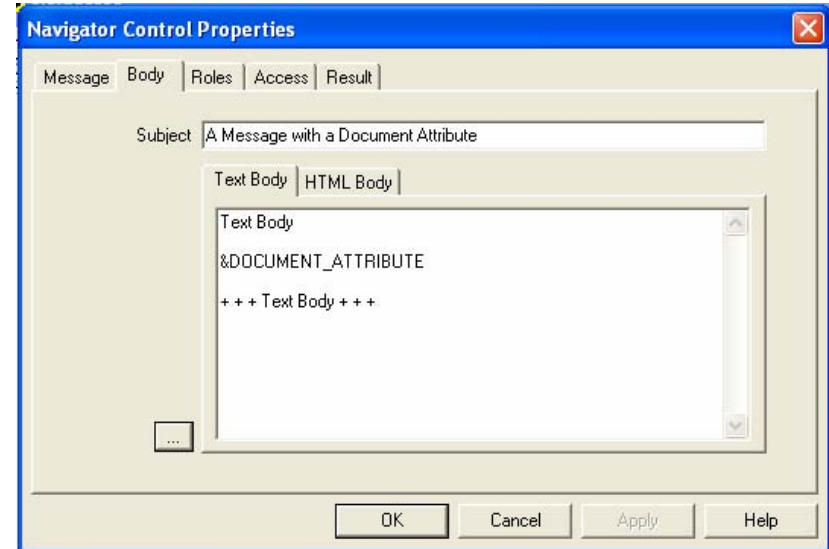
**These are the values in the  
WF Standard  
Comparison Lookup Type:**

EQ	Equal
GT	Greater Than
LT	Less Than
NULL	Null

+++ HTML Body +++

# Display\_type parameter

- In text-based notifications...
- HTML markup appears unescaped
- Solution: Display\_type parameter
- Values - Constants
  - wf\_notification.doc\_text := 'text/plain';
  - wf\_notification.doc\_html := 'text/html';
- You must write code for each value



Text Body

```
<H1>These are the values in the<BR>WF Standard<BR>Comparison Lookup
Type:</H1><TABLE BORDER="1"><TR><TD>EQ</TD><TD>Equal</TD></TR><TR><TD>GT</TD><TD>
Greater
Than</TD></TR><TR><TD>LT</TD><TD>Less
Than</TD></TR><TR><TD>NULL</TD><TD>Null</TD></TR></TABLE>
```

+ + + Text Body + + +

# Display\_type Code

```

1 PROCEDURE djs_demonstrate_document_attrs
2   ( document_id          IN   VARCHAR2
3   , display_type        IN   VARCHAR2
4   , document            IN OUT NOCOPY VARCHAR2
5   , document_type       IN OUT NOCOPY VARCHAR2)
6 IS
7
8   l_document_html VARCHAR2(32747);
9   l_document_text VARCHAR2(32747);
10
11 BEGIN
12
13   l_document_html :=
14     '<H1>These are the values in the<BR>WF Standard'
15     || '<BR>Comparison Lookup Type:</H1>'
16     || '<TABLE BORDER="1">';
17
18   l_document_text :=
19     'These are the values in the WF Standard '
20     || 'Comparison Lookup Type: || CHR(13) || CHR(10);
21
22   FOR l_lookup_rec IN ( SELECT l_lookup_code, meaning
23                         FROM wf_l_lookups
24                         WHERE l_lookup_type = 'WFSTD_COMPARISON' ) LOOP
25
26     l_document_html := l_document_html
27       || '<TR><TD>' || l_lookup_rec.l_lookup_code
28       || '</TD><TD>' || l_lookup_rec.meaning
29       || '</TD></TR>';
30
31     l_document_text := l_document_text || CHR(13) || CHR(10)
32       || RPAD(l_lookup_rec.l_lookup_code, 6)
33       || l_lookup_rec.meaning;
34
35   END LOOP;
36
37   l_document_html := l_document_html
38     || '</TABLE>';
39
40   IF display_type = wf_notification.doc_html THEN
41
42     document := document
43       || l_document_html;
44
45   ELSIF display_type = wf_notification.doc_text THEN
46
47     document := document
48       || l_document_text;
49
50   END IF;
51
52 END djs_demonstrate_document_attrs;

```

```

l_document_text :=
  'These are the values in the WF Standard '
  || 'Comparison Lookup Type: ' || CHR(13) || CHR(10);

```

```

l_document_text :=
  l_document_text || CHR(13) || CHR(10)
  || RPAD(l_lookup_rec.l_lookup_code, 6)
  || l_lookup_rec.meaning;

```

```

IF display_type = wf_notification.doc_html THEN

```

```

  document := document
  || l_document_html;

```

```

ELSIF display_type = wf_notification.doc_text THEN

```

```

  document := document
  || l_document_text;

```

```

END IF;

```

# Text based email...

```

1 PROCEDURE djs_demonstrate_document_attrs
2   ( document_id          IN   VARCHAR2
3   , display_type        IN   VARCHAR2
4   , document            IN OUT NOCOPY VARCHAR2
5   , document_type       IN OUT NOCOPY VARCHAR2)
6 IS
7
8   l_document_html VARCHAR2(32747);
9   l_document_text VARCHAR2(32747);
10
11 BEGIN
12
13   l_document_html :=
14     '<H1>These are the values in the<BR>WF Standard'
15     || '<BR>Comparison Lookup Type:</H1>'
16     || '<TABLE BORDER="1">';
17
18   l_document_text :=
19     'These are the values in the WF Standard '
20     'Comparison Lookup Type: || CHR(13) || CHR(10);
21
22   FOR l_lookup_rec IN ( SELECT l_lookup_code, meaning
23                         FROM wf_lookups
24                         WHERE l_lookup_type = 'WFSTD_COMPARISON' ) LOOP
25
26     l_document_html := l_document_html
27       || '<TR><TD>' || l_lookup_rec.l_lookup_code
28       || '</TD><TD>' || l_lookup_rec.meaning
29       || '</TD></TR>';
30
31     l_document_text := l_document_text || CHR(13) || CHR(10)
32       || RPAD(l_lookup_rec.l_lookup_code, 6)
33       || l_lookup_rec.meaning;
34
35   END LOOP;
36
37   l_document_html := l_document_html
38     || '</TABLE>';
39
40   IF display_type = wf_notification.doc_html THEN
41     document := document
42       || l_document_html;
43
44   ELSIF display_type = wf_notification.doc_text THEN
45     document := document
46       || l_document_text;
47
48   END IF;
49
50
51 END djs_demonstrate_document_attrs;

```

## Text Body

These are the values in the WF  
Standard Comparison Lookup Type:

EQ Equal

GT Greater Than

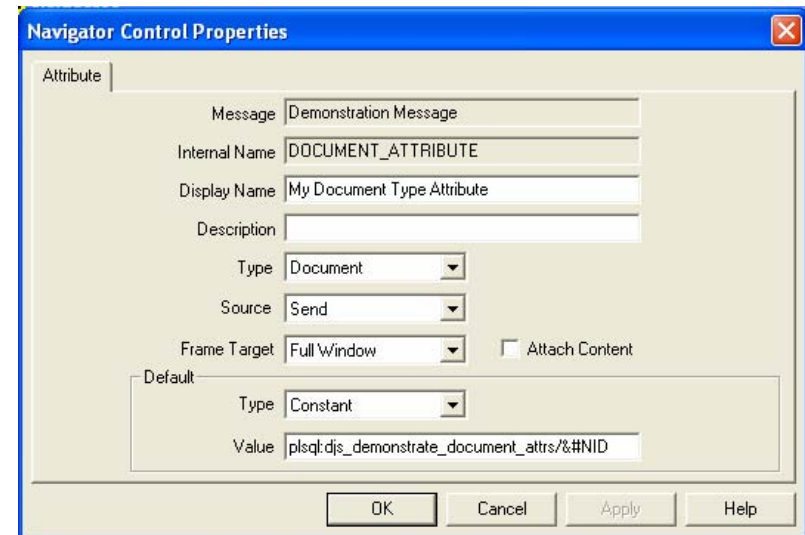
LT Less Than

NULL Null

+ + + Text Body + + +

# Document\_id parameter

- One inbound parameter: document\_id
- Append to the end of procedure name
  - pl sql : my\_package\_name. my\_proc\_name/whatever
- Use to pass and derive specific information about the process instance:
  - Pass text strings
  - Item attributes
    - &#NID for notification\_id
    - Then, derive item\_key, item\_type



# Document Attribute Code

```

1 1 PROCEDURE djs_demonstrate_document_attrs
2 CREATE OR REPLACE PROCEDURE djs_demonstrate_document_attrs
3 ( document_id          IN   VARCHAR2
4 , display_type        IN   VARCHAR2
5 , document            IN OUT NOCOPY VARCHAR2
6 , document_type      IN OUT NOCOPY VARCHAR2)
IS
    l_item_type wf_items.item_type%TYPE;
    l_item_key  wf_items.item_key%TYPE;
BEGIN
    SELECT wias.item_type
    , wias.item_key
    INTO l_item_type
    , l_item_key
    FROM applsys.wf_item_activity_statuses wias
    WHERE wias.notification_id = document_id ;

    document := document
    || '<TABLE BORDER="1">'
    || '<TR><TD>Item Type is ' || l_item_type
    || '</TD><TD>Item Key is ' || l_item_key
    || '</TD></TR></TABLE>';
END;
```

HTML Body

Item Type is DJS_DEMO	Item Key is 164110
-----------------------	--------------------

+++ HTML Body +++

IS

```

l_item_type wf_items.item_type%TYPE;
l_item_key  wf_items.item_key%TYPE;
```

BEGIN

```

SELECT wias.item_type
, wias.item_key
INTO l_item_type
, l_item_key
FROM applsys.wf_item_activity_statuses wias
WHERE wias.notification_id = document_id ;
```

document := document

```

|| '<TABLE BORDER="1">'
|| '<TR><TD>Item Type is ' || l_item_type
|| '</TD><TD>Item Key is ' || l_item_key
|| '</TD></TR></TABLE>';
```

END;

# Document Type Attributes

## Issues:

- PLSQL not version controlled
  - Values derived at runtime - on the fly
- Handle Document parameter correctly: IN OUT
- Changes to code require a mailer bounce

# Mailer Restart Required

ORA-04068: existing state of packages has been discarded

ORA-04061: existing state of package "<package\_name>. <proc>" has been invalidated

- When using Document Type attributes with Notification Mailer
- Mailer runs in a continuous session
- See Metalink Note 303260.1
- Other errors may require a restart of ALL Workflow Services



# One final tip: Debug w/ Anonymous Block

```

DECLARE
  l_resultOut  VARCHAR2(200);
BEGIN
  my_workflow_proc ( itemtype => 'DJS_DEMO'
                    , itemkey  => '123456'
                    , actid    => 81005
                    , funcmode => 'RUN'
                    , resultOut => l_resultOut);
  dbms_output.put_line(l_out);
END;

DECLARE
  l_out          VARCHAR2(32767) := '';
  l_mime_type    VARCHAR2(50)    := 'text/plain';
BEGIN
  my_package.my_proc ( document => '1206521'
                     , display_type => 'text/html'
                     , document    => l_out
                     , document_type => l_mime_type );
  dbms_output.put_line(v_out);
END;

```

# Section 7 It's a Wrap!



## Conclusion and Wrap-Up

# More Info on Metalink

- Workflow FAQ General  
Metalink Document 187735.1
- Workflow FAQ (More mailer questions)  
Metalink Document 48666.1
- Workflow FAQs and Patch Information  
Metalink Document 225453.1
- Frequently Asked Questions on Purging of Oracle Workflow Data  
Metalink Document 277124.1

# More Info on Metalink

- White Paper :Getting Started with Workflow - Standalone 2.6.3  
Metalink Document 266612.1
- How to Create a New Workflow  
Metalink Document 47711.1
- Desupport Notice for Workflow Cartridge  
Metalink Document 391546.1
- Workflow Tables Data Growth Issues  
Metalink Document 298550.1

# PDF Documentation

For Version 2.6.3 (11.5.9)

- Oracle Workflow Administrators Guide (Part Number B10283-01)
- Oracle Workflow API Reference (Part Number B10286-01)
- Oracle Workflow Developer's Guide (Part Number B10284-01)
- Oracle Workflow Users Guide (Part Number B-10285-01)
  
- Download these items from OTN
  - <http://www.oracle.com/technology/software/index.html>
  
- Oracle Workflow Product Documentation  
(includes links to more PDF documentation)  
Metalink Document 67183.1

# Fora

- Oracle Workflow Forum (on OTN)
  - <http://forums.oracle.com/forums/forum.jspa?forumID=72>
- Workflow FAQ Forum – Matt Seale
  - <http://smforum.workflowfaq.com/>

# Thank-you!

- Please complete session survey
- Dan Stober
  - dan.stober@imail.org
  - Reference this presentation in subject line

# Legal Stuff

The content in this presentation was created by and is owned by Daniel Stober. You may use this presentation as a reference for your own understanding of the material contained herein. The material may not be presented by anyone other than its creator, nor can it used as the sole or substantial basis of another presentation without the expressed permission of its creator. It may be referenced with proper attribution.

The material contained herein, is based on research and empirical testing which the author believes to be accurate as of the time of the presentation, However, the author makes no warranty as to the accuracy. Responsibility for verifying assertions presented herein rest solely with the user of the information.