11i/2.6 Implement Oracle Workflow

Volume 2 - Student Guide

D13401GC10

Edition 1.0

February 2002

D34104

Author

Clara Jaeckel

Technical Contributors and Reviewers

Mark Craig, Bill Curtis, Leta Davis, Kevin Hudson, Adam Laro-Bashford, Helena Martinovic

This book was published using: oracle*tutor*

Table of Contents

Preface

Profile

Before You Begin This Course

Before you begin this course, you should have the following qualifications:

- Thorough knowledge of Oracle8*i* or Oracle9*i* Database Server and Oracle9*i* Application Server technology
- If you plan to use Oracle Workflow with Oracle E-Business Suite, thorough knowledge of Oracle E-Business Suite
- Working knowledge of XML

Prerequisites

- *Introduction to Oracle9i: SQL*
- *Oracle9i: Program with PL/SQL*

How This Course Is Organized

11i/2.6 Implement Oracle Workflow is an instructor-led course featuring lecture and hands-on exercises. Online demonstrations and written practice sessions reinforce the concepts and skills introduced.

Related Publications

Oracle Publications

Title	**Part Number**
Oracle Workflow Guide Release 2.6.2	*A95625*

Additional Publications

- System release bulletins
- Installation and user's guides
- *read.me* files
- *Oracle Magazine*

Typographic Conventions

Typographic Conventions in Text

Convention	Element	Example
Bold italic	Glossary term (if there is a glossary)	The ***algorithm*** inserts the new key.
Caps and lowercase	Buttons, check boxes, triggers, windows	Click the Executable button. Select the Can't Delete Card check box. Assign a When-Validate-Item trigger to the ORD block. Open the Master Schedule window.
Courier new, case sensitive (default is lowercase)	Code output, directory names, filenames, passwords, pathnames, URLs, user input, usernames	Code output: `debug.set ('I", 300);` Directory: `bin` (DOS), `$FMHOME` (UNIX) Filename: Locate the `init.ora` file. Password: User `tiger` as your password. Pathname: Open `c:\my_docs\projects` URL: Go to `http://www.oracle.com` User input: Enter `300` Username: Log on as `scott`
Initial cap	Graphics labels (unless the term is a proper noun)	Customer address (*but* Oracle Payables)
Italic	Emphasized words and phrases, titles of books and courses, variables	Do *not* save changes to the database. For further information, see *Oracle7 Server SQL Language Reference Manual.* Enter `user_id@us.oracle.com`, where *user_id* is the name of the user.
Quotation marks	Interface elements with long names that have only initial caps; lesson and chapter titles in cross-references	Select "Include a reusable module component" and click Finish. This subject is covered in Unit II, Lesson 3, "Working with Objects."
Uppercase	SQL column names, commands, functions, schemas, table names	Use the SELECT command to view information stored in the LAST_NAME column of the EMP table.

Convention	Element	Example
Arrow	Menu paths	Select File—> Save.

Brackets	Key names	Press [Enter].
Commas	Key sequences	Press and release keys one at a time: [Alternate], [F], [D]
Plus signs	Key combinations	Press and hold these keys simultaneously: [Ctrl]+[Alt]+[Del]

Typographic Conventions in Code

Convention	Element	Example
Caps and lowercase	Oracle Forms triggers	`When-Validate-Item`

Lowercase	Column names, table names	`SELECT last_name`
		`FROM s_emp;`

	Passwords	`DROP USER scott`
		`IDENTIFIED BY tiger;`

	PL/SQL objects	`OG_ACTIVATE_LAYER`
		`  (OG_GET_LAYER`
		`('prod_pie_layer'))`

Lowercase italic	Syntax variables	`CREATE ROLE` *`role`*

Uppercase	SQL commands and functions	`SELECT userid`
		`FROM emp;`

Typographic Conventions in Navigation Paths

This course uses simplified navigation paths, such as the following example, to direct you through Oracle Applications.

(N) Invoice > Entry > Invoice Batches Summary (M) Query > Find (B) Approve

This simplified path translates to the following:

1. (N) From the Navigator window, select Invoice > Entry > Invoice Batches Summary.

2. (M) From the menu, select Query > Find.

3. (B) Click the Approve button.

Notations :

(N) = Navigator

(M) = Menu

(T) = Tab

(I) = Icon

(H) = Hyperlink

(B) = Button

Typographical Conventions in Help System Paths

This course uses a “navigation path” convention to represent actions you perform to find pertinent information in the Oracle Applications Help System.

The following help navigation path, for example—

(Help) General Ledger > Journals > Enter Journals

—represents the following sequence of actions:

1. In the navigation frame of the help system window, expand the General Ledger entry.
2. Under the General Ledger entry, expand Journals.
3. Under Journals, select Enter Journals.
4. Review the Enter Journals topic that appears in the document frame of the help system window.

Getting Help

Oracle Applications provides you with a complete online help facility.

Whenever you need assistance, simply choose an item from the Help menu to pinpoint the type of information you want.

To display help for a current window:

1. Choose Window Help from the Help menu, click the Help button on the toolbar, or hold down the Control key and type 'h'.

   A web browser window appears, containing search and navigation frames on the left, and a frame that displays help documents on the right.

   The document frame provides information on the window containing the cursor. The navigation frame displays the top-level topics for your responsibility, arranged in a tree control.

2. If the document frame contains a list of topics associated with the window, click on a topic of interest to display more detailed information.
3. You can navigate to other topics of interest in the help system, or choose Close from your web browser's File menu to close help.

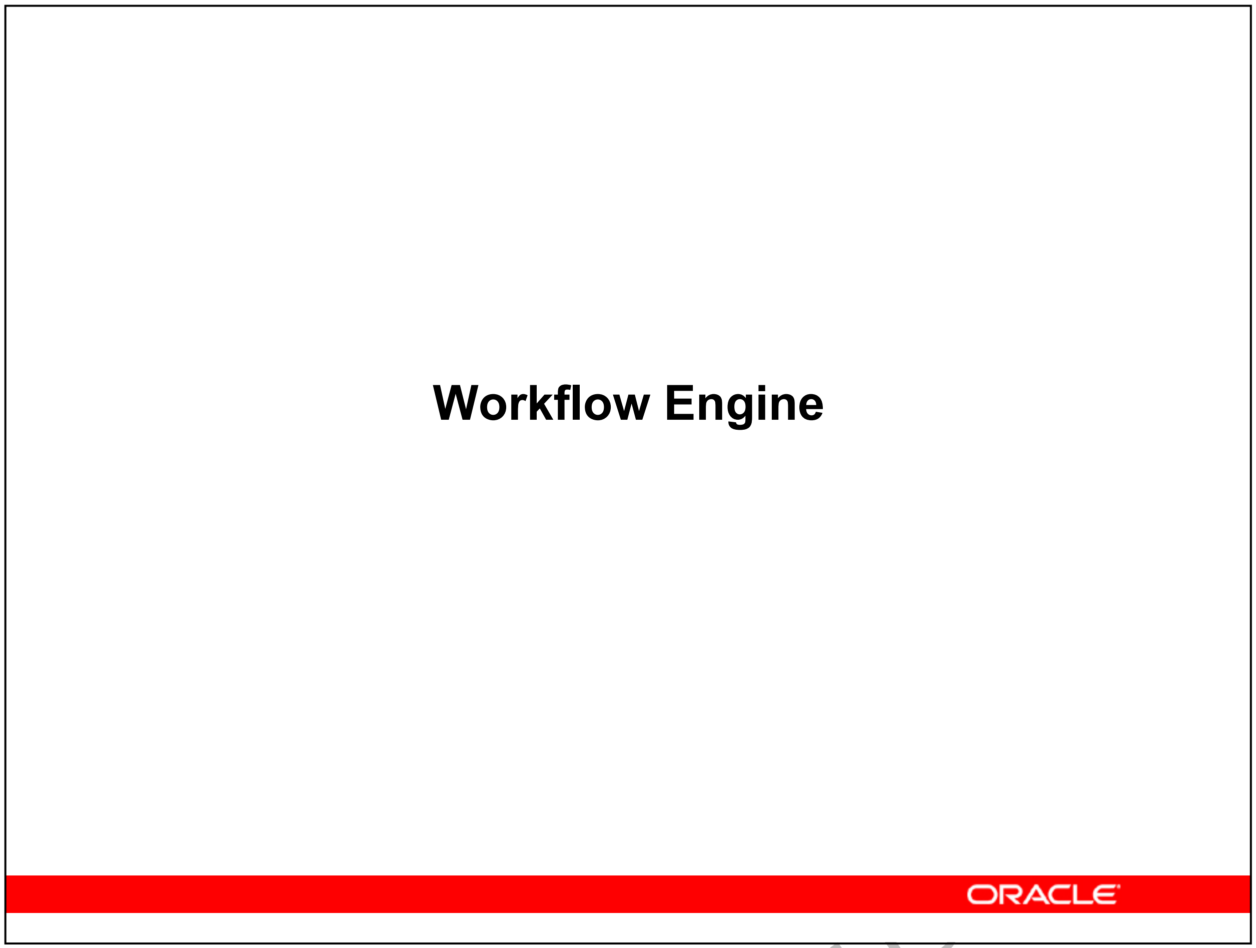
Searching for Help

You can perform a search to find the Oracle Applications help information you want. Simply enter your query in the text field located in the top-left frame of the browser window when viewing help, then click the adjacent Find button.

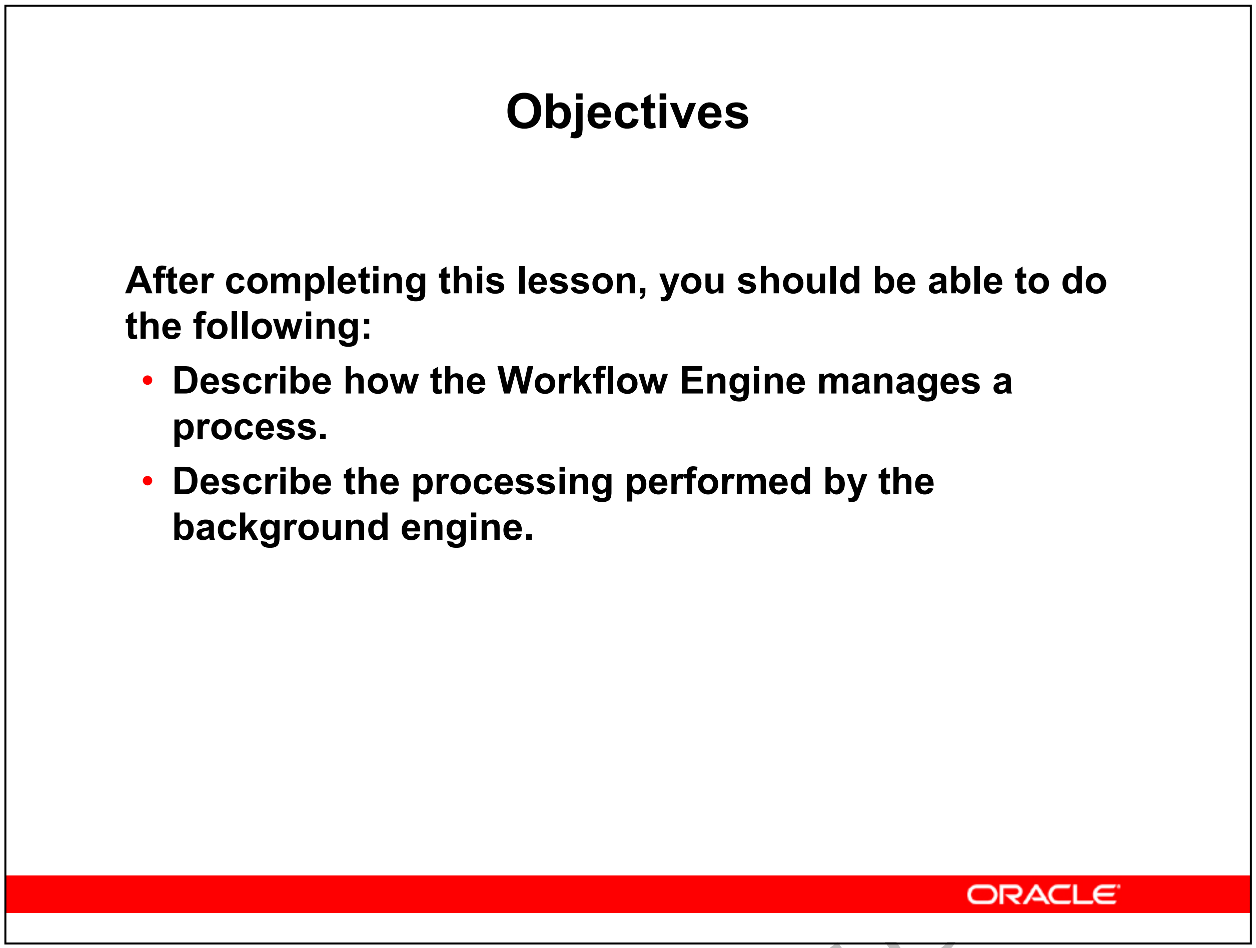
A list of titles, ranked by relevance and linked to the documents in question, is returned from your search in the right-hand document frame. Click on whichever title seems to best answer your needs to display the complete document in this frame. If the document doesn't fully answer your questions, use your browser's Back button to return to the list of titles and try another.

Workflow Engine

Chapter 14



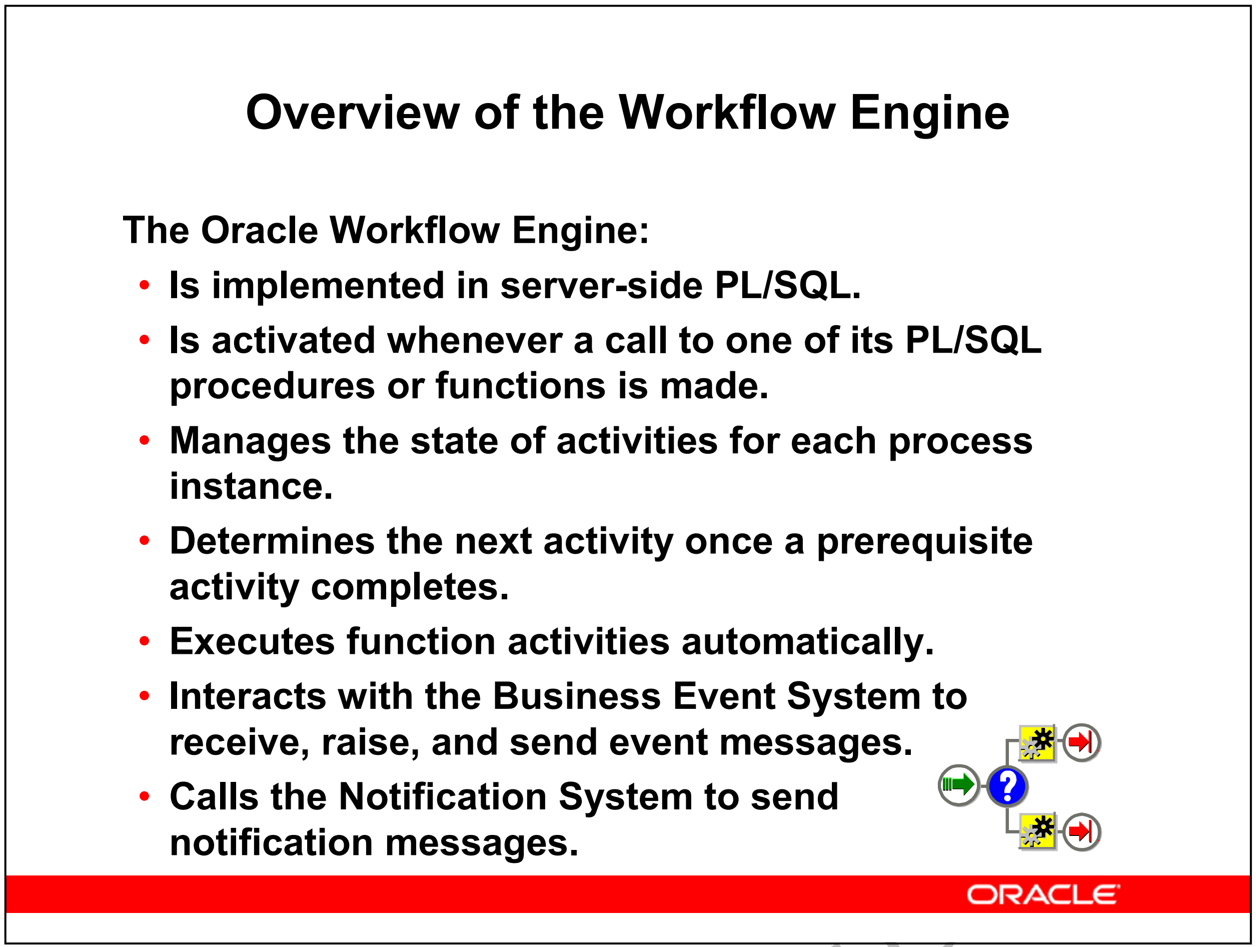
Workflow Engine



Objectives

After completing this lesson, you should be able to do the following:

- Describe how the Workflow Engine manages a process.
- Describe the processing performed by the background engine.



Overview of the Workflow Engine

The Oracle Workflow Engine:

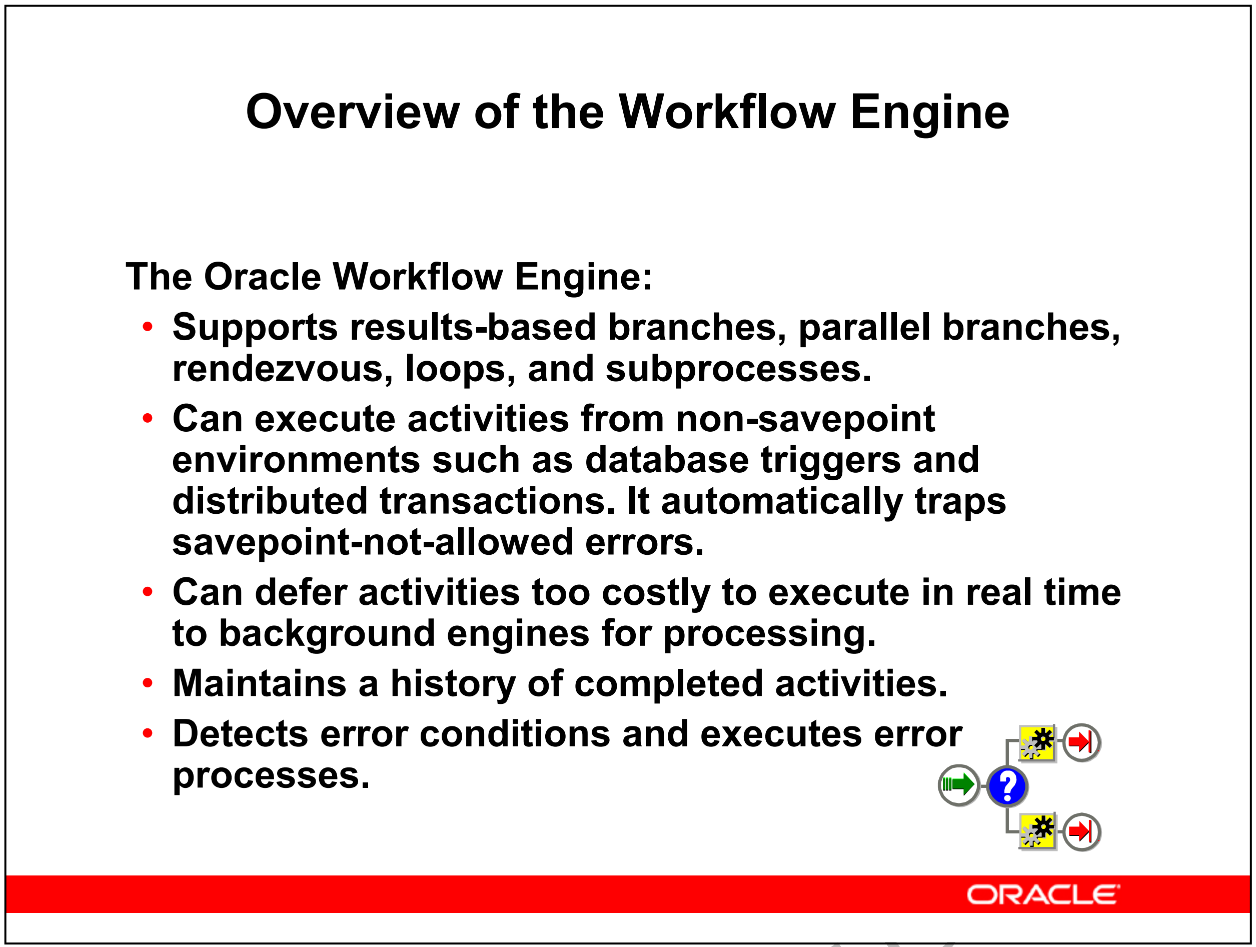
- **Is implemented in server-side PL/SQL.**
- **Is activated whenever a call to one of its PL/SQL procedures or functions is made.**
- **Manages the state of activities for each process instance.**
- **Determines the next activity once a prerequisite activity completes.**
- **Executes function activities automatically.**
- **Interacts with the Business Event System to receive, raise, and send event messages.**
- **Calls the Notification System to send notification messages.**

Overview of the Workflow Engine

The Notification System is also implemented in server-side PL/SQL and can interface with the Workflow web agent or the Notification Mailer program to deliver notifications to end users.

The Workflow Engine and the Business Event System can function independently of each other. However, you can now achieve the most powerful and flexible processing by using the Workflow Engine and the Business Event System together to execute cross-system processes for e-business integration.

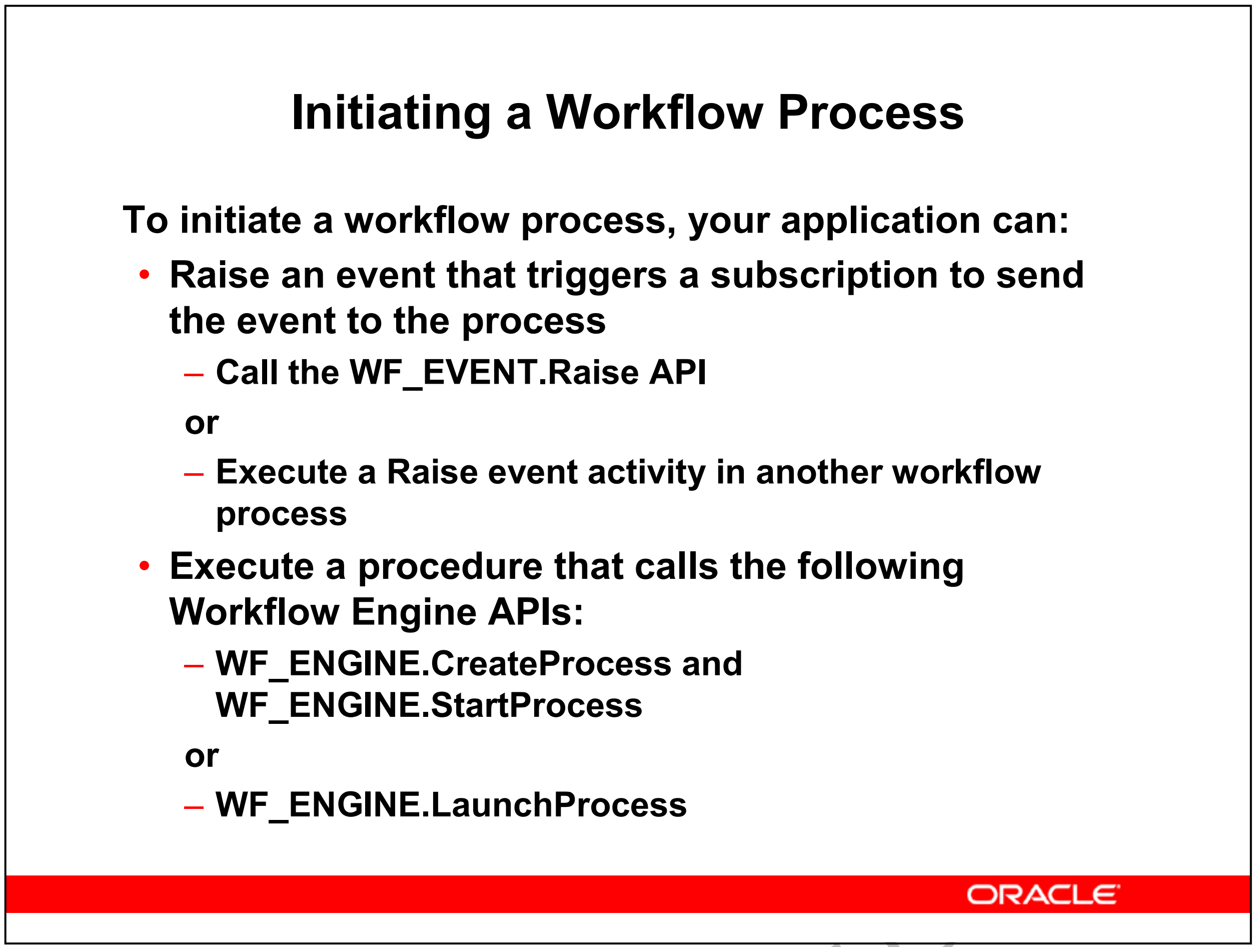
For more information, refer to the Oracle Workflow APIs chapter in the *Oracle Workflow Guide*.



Overview of the Workflow Engine

The Oracle Workflow Engine:

- **Supports results-based branches, parallel branches, rendezvous, loops, and subprocesses.**
- **Can execute activities from non-savepoint environments such as database triggers and distributed transactions. It automatically traps savepoint-not-allowed errors.**
- **Can defer activities too costly to execute in real time to background engines for processing.**
- **Maintains a history of completed activities.**
- **Detects error conditions and executes error processes.**



Initiating a Workflow Process

To initiate a workflow process, your application can:

- **Raise an event that triggers a subscription to send the event to the process**
  - **Call the WF_EVENT.Raise API**

  or

  - **Execute a Raise event activity in another workflow process**
- **Execute a procedure that calls the following Workflow Engine APIs:**
  - **WF_ENGINE.CreateProcess and WF_ENGINE.StartProcess**

  or

  - **WF_ENGINE.LaunchProcess**

Initiating a Workflow Process

- If you use an event to initiate a workflow process, you must define a subscription that sends the event message to the workflow process when the event is raised. You can also define a subscription to send an event to start a workflow process when the event message is received from an external source. In both cases, the process must begin with a Receive event activity that is marked as a Start activity node to receive the event.
- If you call the Workflow Engine APIs to initiate a workflow process, use the CreateProcess and StartProcess APIs if you want to perform additional tasks, such as setting item attributes, after creating and before starting the process. If you do not need to perform any additional tasks, you can use the LaunchProcess API, which is a wrapper combining the CreateProcess and StartProcess APIs.

  The procedure that executes the Workflow Engine APIs to initiate a process must identify the item type and item key of the process for these APIs. The item type and item key passed to these APIs uniquely identify an item and must be passed to subsequent API calls for each specific process. The process must begin with the standard Start activity or with another function, notification, or process activity, marked as a Start activity node.

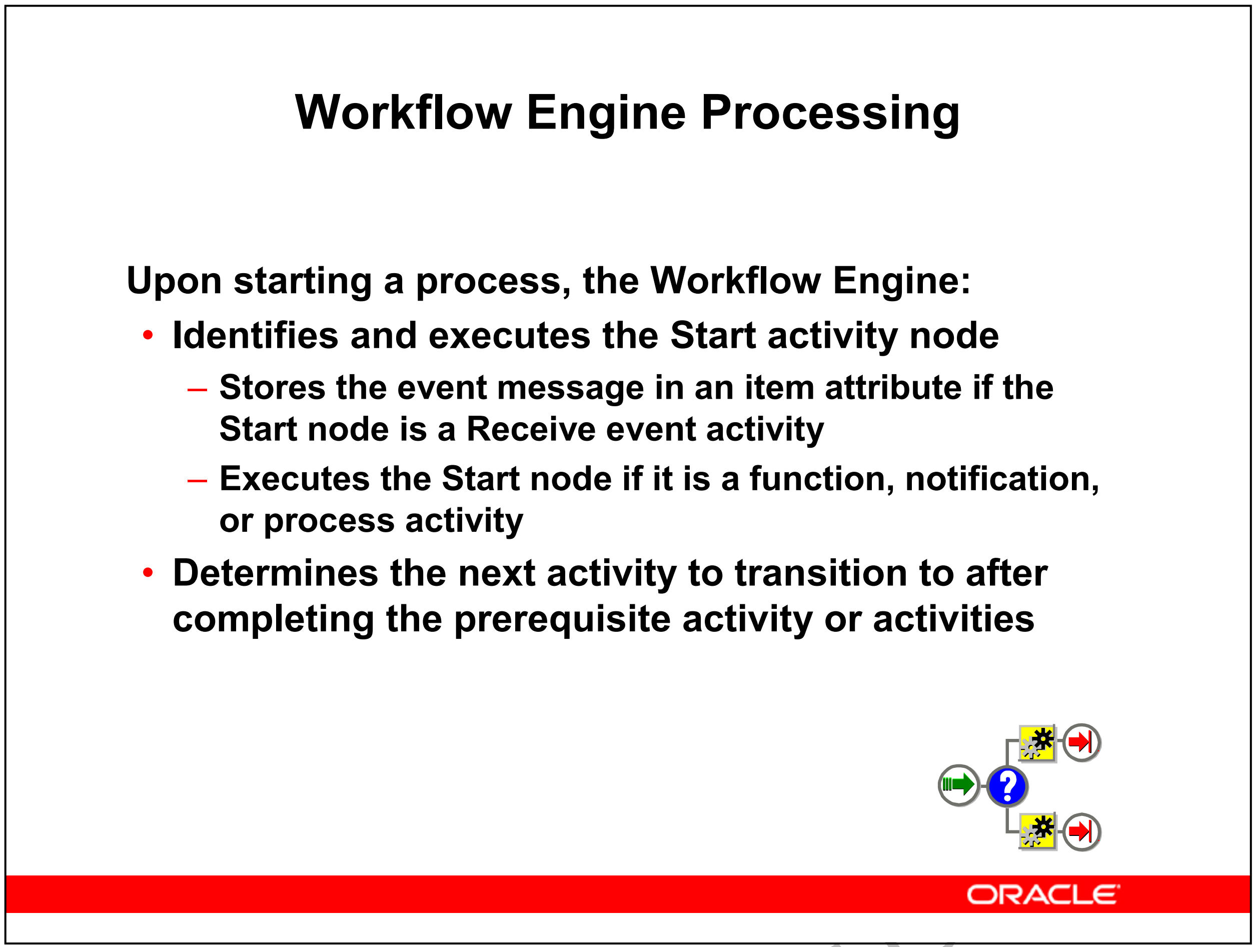
- Oracle Workflow also includes a Launch Processes web page for initiating workflow processes. The Launch Processes link is available from the Workflow home page in standalone Oracle Workflow or from a workflow administrator responsibility in embedded Oracle Workflow. It is intended for use by the workflow administrator to test workflow processes in a development environment.

For more information, refer to the Workflow Engine APIs section in the *Oracle Workflow Guide*.

Example

The following example shows how to initiate a process using the CreateProcess, SetItemUserKey, SetItemAttribute, SetItemOwner, and StartProcess APIs.

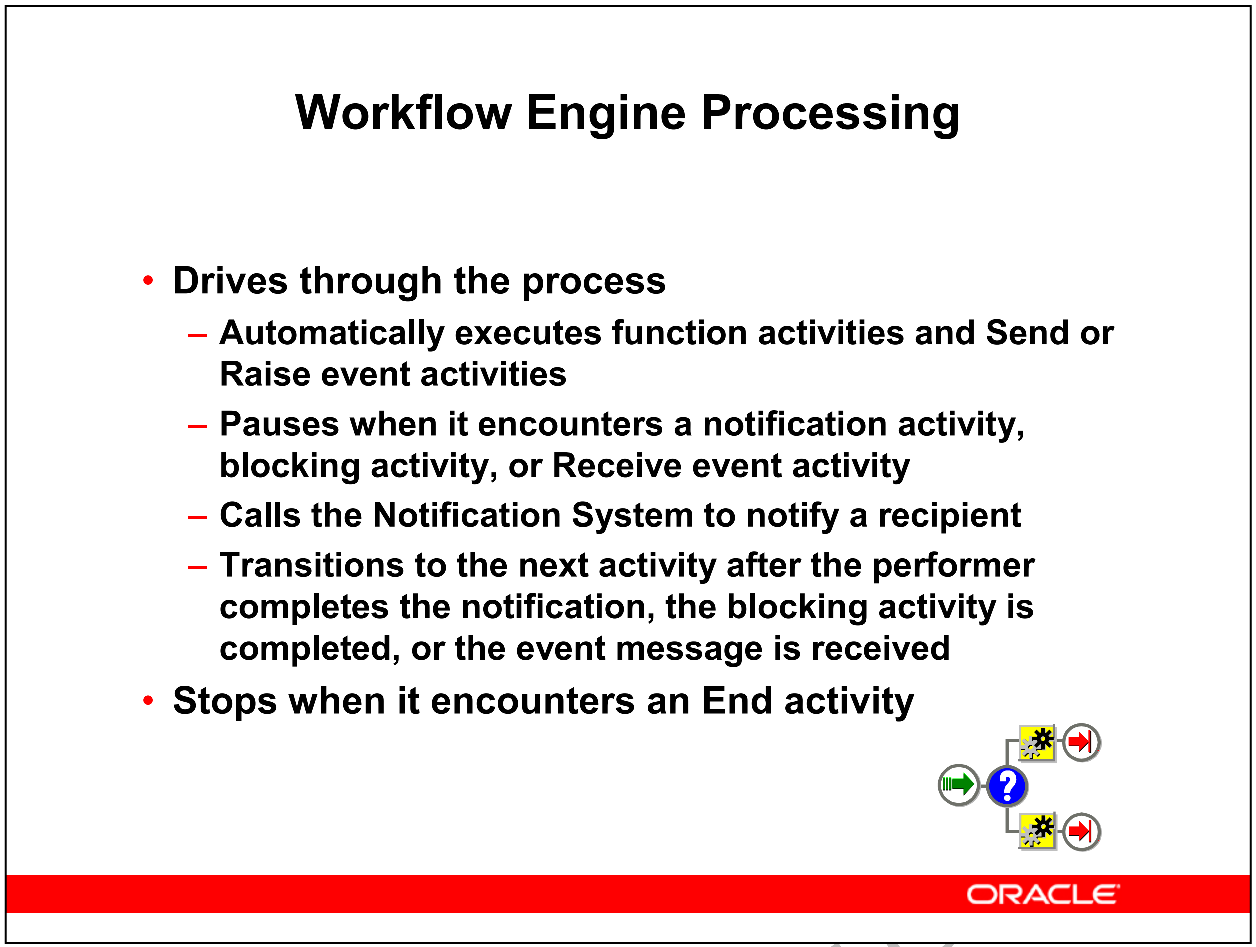
```
wf_engine.CreateProcess( ItemType => ItemType,
                         ItemKey  => ItemKey,
                         process  => WorkflowProcess );

wf_engine.SetItemUserKey ( ItemType => ItemType,
                           ItemKey  => ItemKey,
                           UserKey  => ItemUserKey);
--
--
-- Initialize workflow item attributes
--
wf_engine.SetItemAttrText ( itemtype => itemtype,
                            itemkey  => itemkey,
                            aname    => 'REQUISITION_NUMBER',
                            avalue   => RequisitionNumber);
--
wf_engine.SetItemAttrNumber ( itemtype => itemtype,
                              itemkey  => itemkey,
                              aname    => 'REQUISITION_AMOUNT',
                              avalue   => RequisitionAmount );
--

...

--
wf_engine.SetItemOwner ( itemtype => itemtype,
                         itemkey  => itemkey,
                         owner    => ProcessOwner );
--
wf_engine.StartProcess( itemtype => itemtype,
                        itemkey  => itemkey );
```



Workflow Engine Processing

Upon starting a process, the Workflow Engine:

- Identifies and executes the Start activity node
  - Stores the event message in an item attribute if the Start node is a Receive event activity
  - Executes the Start node if it is a function, notification, or process activity
- Determines the next activity to transition to after completing the prerequisite activity or activities



Workflow Engine Processing

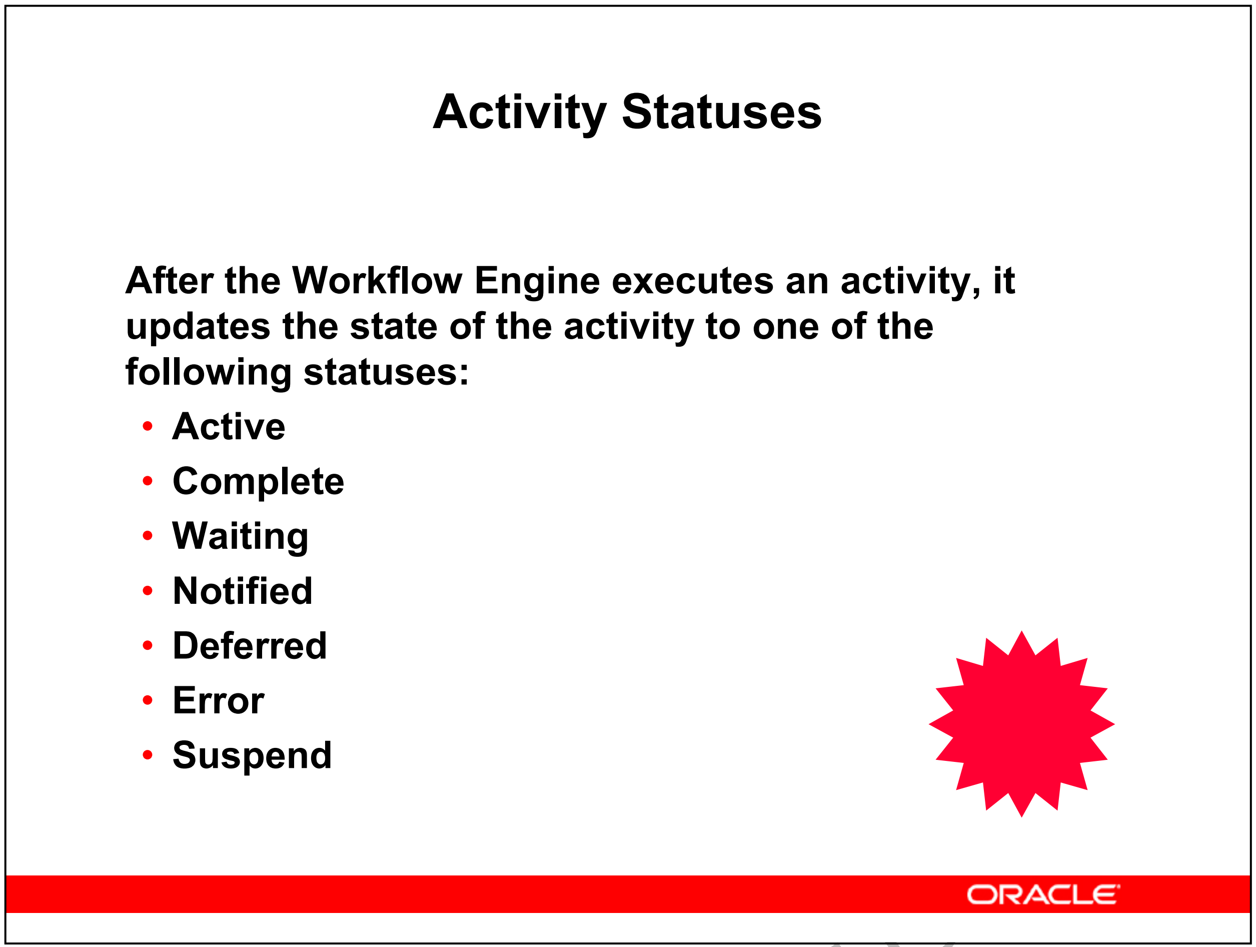
- Drives through the process
  - Automatically executes function activities and Send or Raise event activities
  - Pauses when it encounters a notification activity, blocking activity, or Receive event activity
  - Calls the Notification System to notify a recipient
  - Transitions to the next activity after the performer completes the notification, the blocking activity is completed, or the event message is received
- Stops when it encounters an End activity

Workflow Engine Processing

The Workflow Engine sets a savepoint for each completed function activity, and saves all consecutively completed function activities as part of one commit cycle. If an error occurs during a commit, the database can roll back to an appropriate savepoint.

The Workflow Engine never issues a commit, as it is the responsibility of the calling application to commit. When the Workflow Engine encounters a blocking activity such as a response-required notification, it sets the activity to the appropriate status, such as NOTIFIED, and returns control to the calling application. That application can then issue a commit.

Note: A notification activity that sends an FYI notification message will be automatically completed by the Workflow Engine. Only notifications that prompt for a response will cause the Workflow Engine to pause and wait for a response from the notification recipient.



Activity Statuses

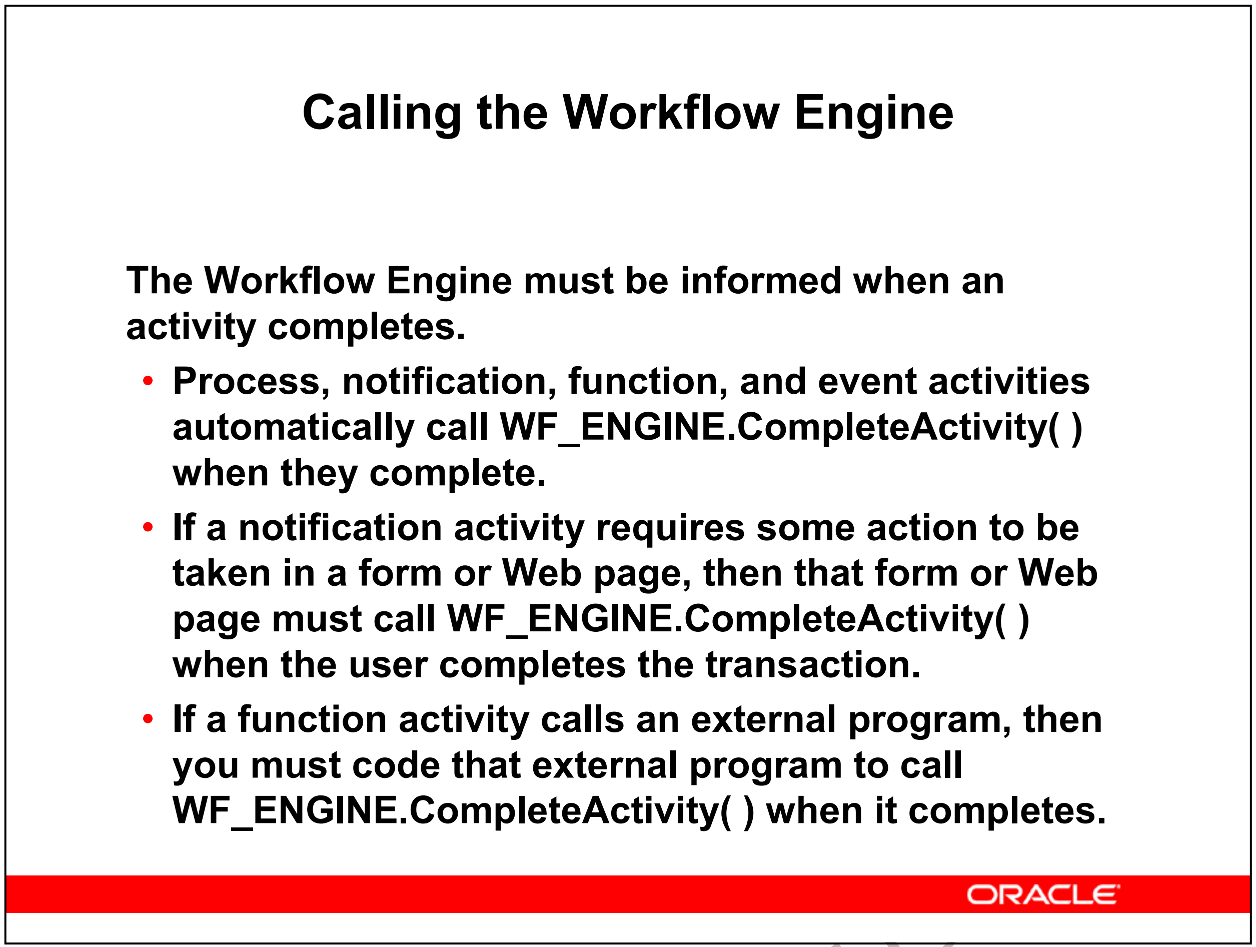
After the Workflow Engine executes an activity, it updates the state of the activity to one of the following statuses:

- **Active**
- **Complete**
- **Waiting**
- **Notified**
- **Deferred**
- **Error**
- **Suspend**

ORACLE

Activity Statuses

- Active: The activity is currently being executed. The topmost process will be active until the entire process completes.
- Complete: The activity executed successfully.
- Waiting: The activity is waiting for dependencies to complete. An example is the AND activity, where one of its incoming activity transitions is complete but is still waiting for another required incoming activity transition to complete before it can be marked as complete.
- Notified: The activity is waiting for a response from a notification, for an event message, or for an external program to complete and call the Workflow Engine.
- Deferred: The activity is deferred to a background engine for execution.
- Error: The activity has encountered an error during execution.
- Suspend: The activity is suspended from further execution.



Calling the Workflow Engine

The Workflow Engine must be informed when an activity completes.

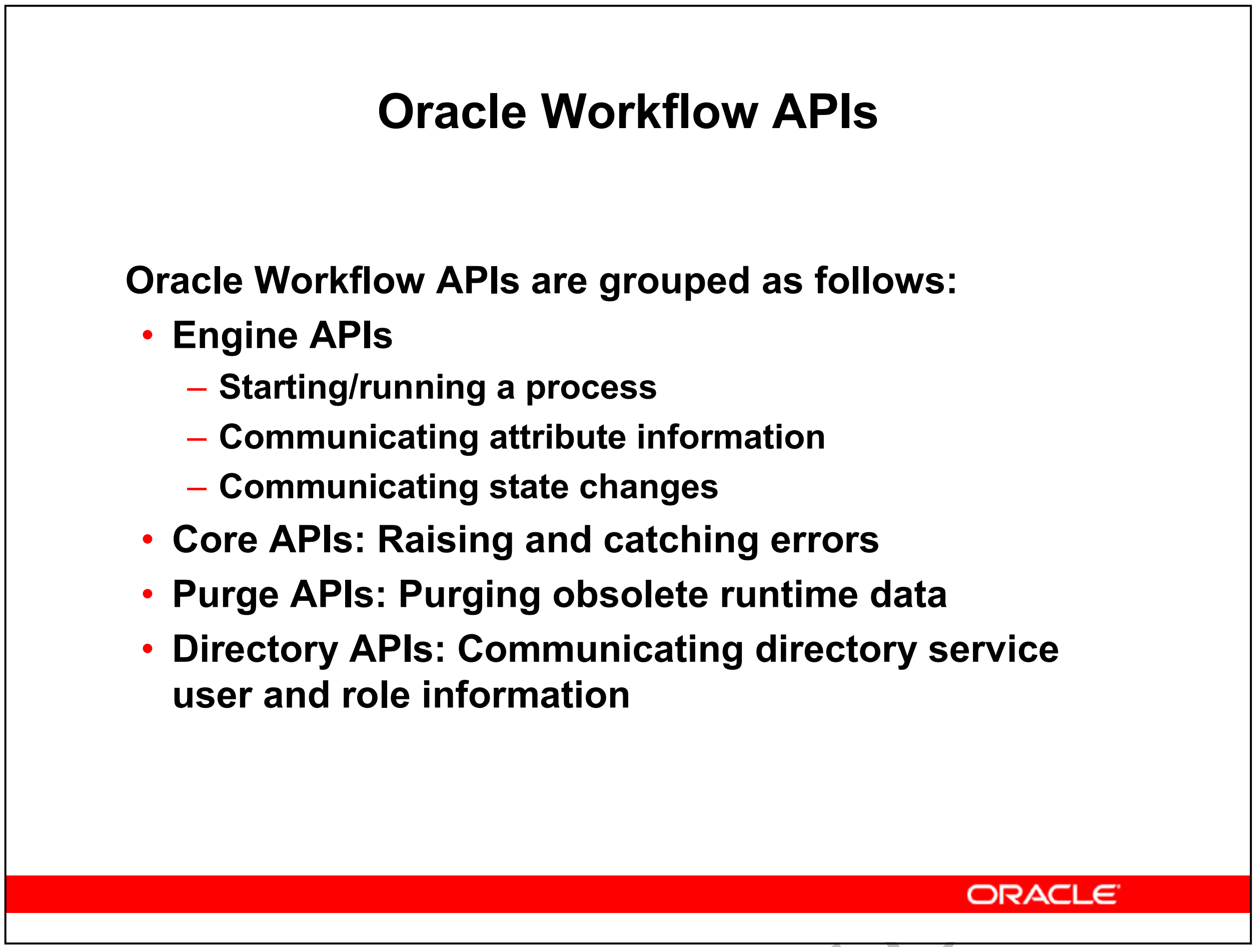
- **Process, notification, function, and event activities automatically call WF_ENGINE.CompleteActivity( ) when they complete.**
- **If a notification activity requires some action to be taken in a form or Web page, then that form or Web page must call WF_ENGINE.CompleteActivity( ) when the user completes the transaction.**
- **If a function activity calls an external program, then you must code that external program to call WF_ENGINE.CompleteActivity( ) when it completes.**

ORACLE

Calling the Workflow Engine

You can add a standard Block activity after any function activity that calls an external program. The Block activity will pause the process until the external program completes and makes a call to WF_ENGINE.CompleteActivity( ).

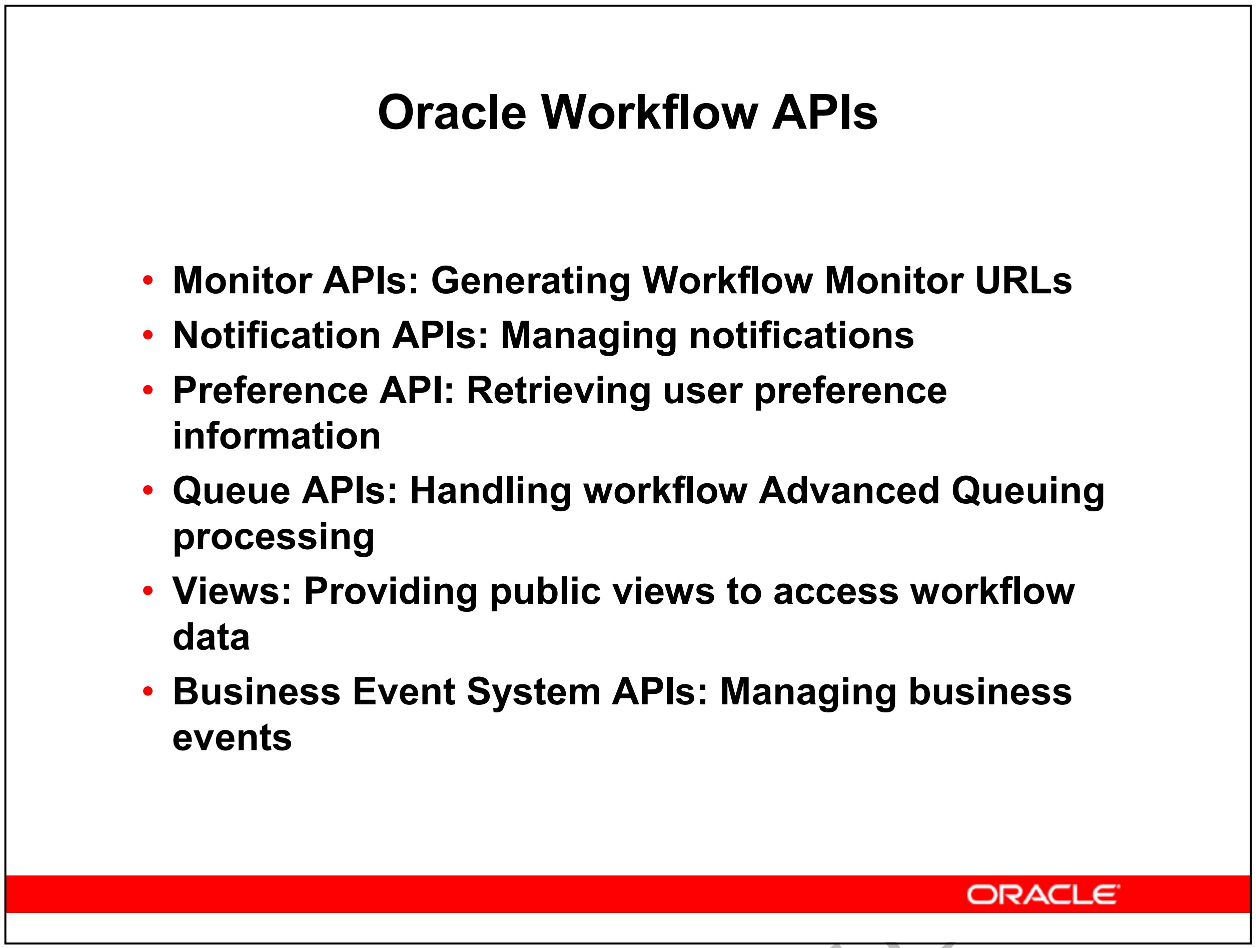
For more information, refer to the Oracle Workflow APIs chapter in the *Oracle Workflow Guide*.



Oracle Workflow APIs

Oracle Workflow APIs are grouped as follows:

- Engine APIs
  - Starting/running a process
  - Communicating attribute information
  - Communicating state changes
- Core APIs: Raising and catching errors
- Purge APIs: Purging obsolete runtime data
- Directory APIs: Communicating directory service user and role information



Oracle Workflow APIs

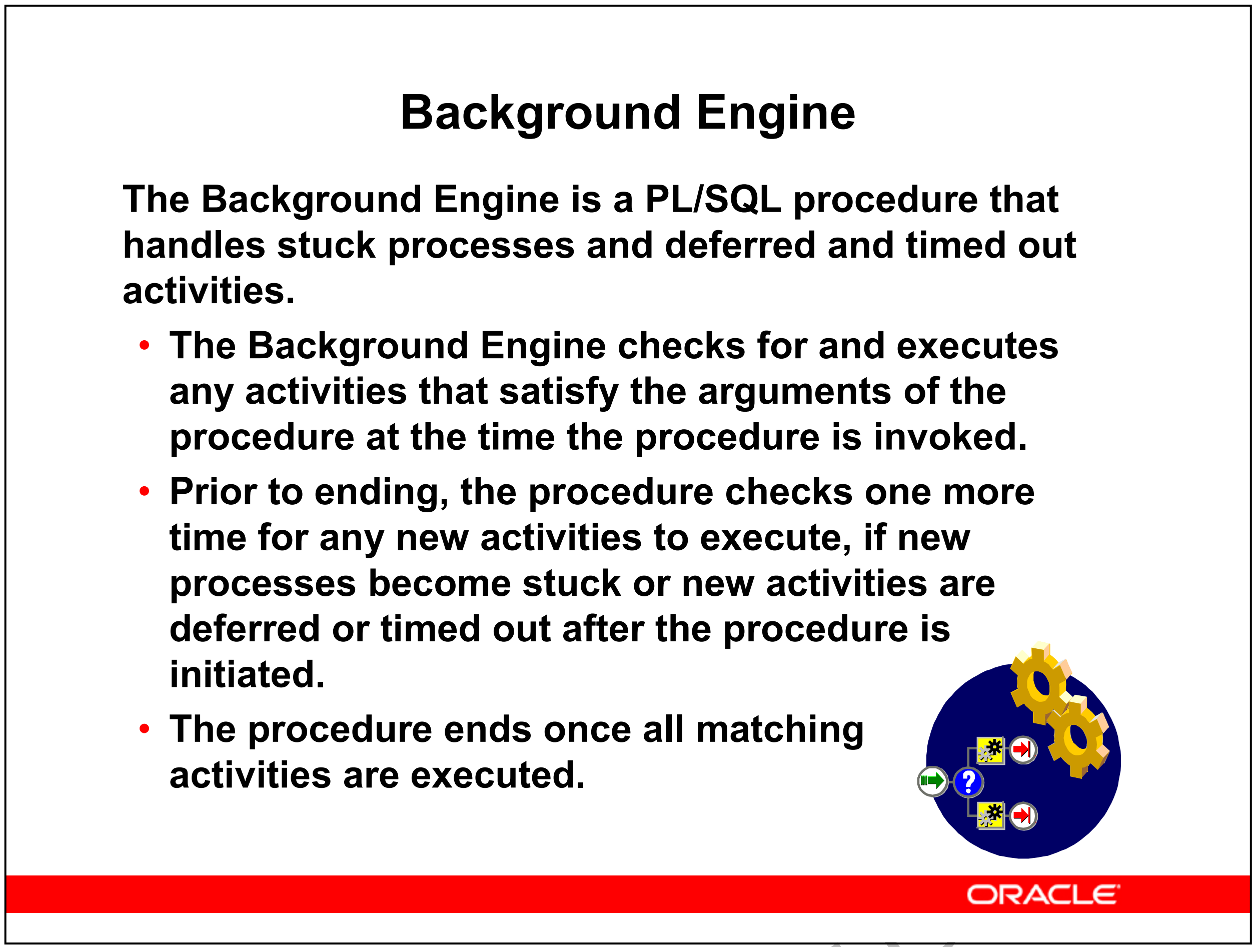
- Monitor APIs: Generating Workflow Monitor URLs
- Notification APIs: Managing notifications
- Preference API: Retrieving user preference information
- Queue APIs: Handling workflow Advanced Queuing processing
- Views: Providing public views to access workflow data
- Business Event System APIs: Managing business events

Oracle Workflow APIs

The public views are installed in the APPS account for the version of Oracle Workflow embedded in Oracle E-Business Suite.

Note: Oracle Workflow also includes document management APIs. Document management functionality will be supported in a future release.

For more information, refer to the Oracle Workflow APIs chapter in the *Oracle Workflow Guide*.



Background Engine

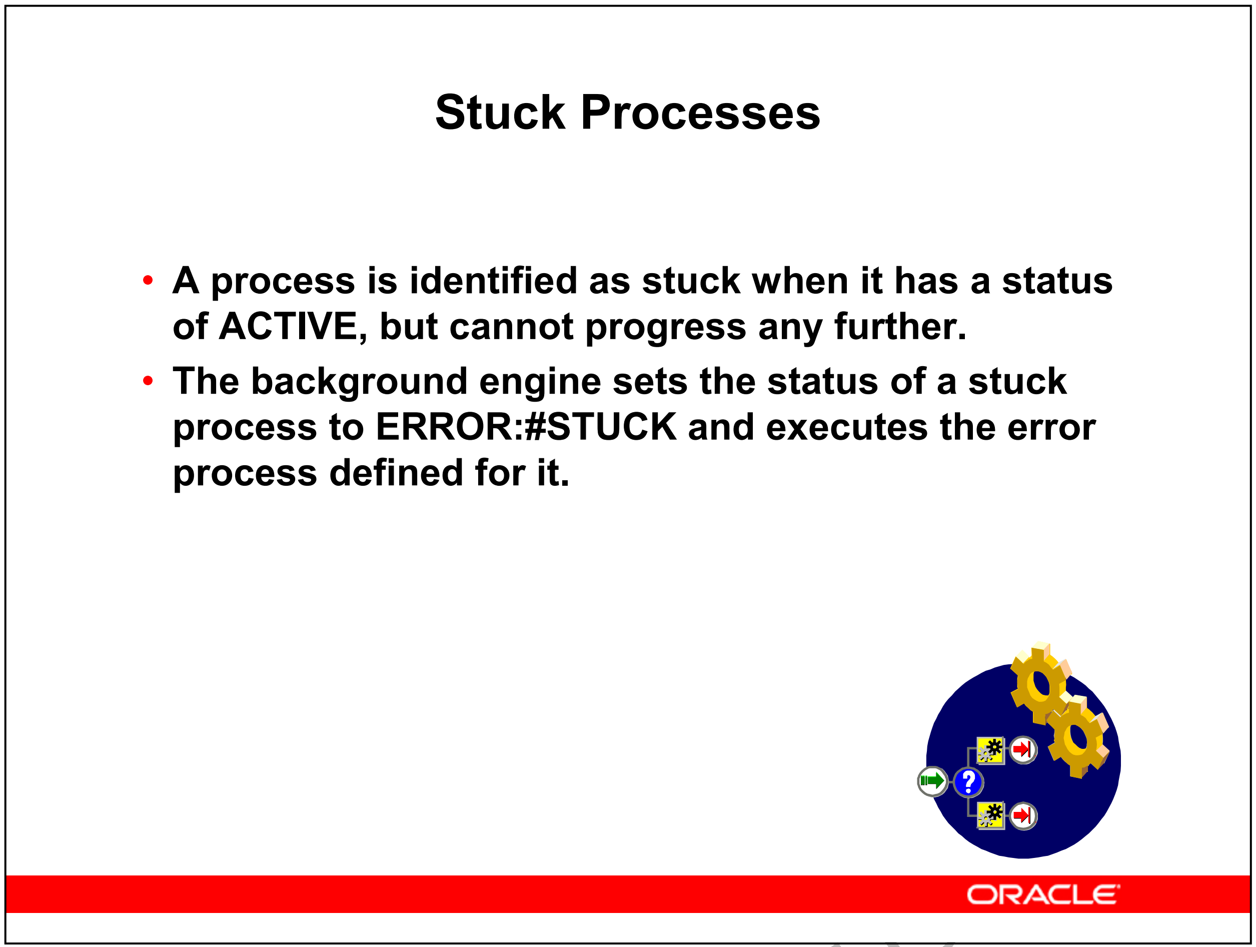
The Background Engine is a PL/SQL procedure that handles stuck processes and deferred and timed out activities.

- **The Background Engine checks for and executes any activities that satisfy the arguments of the procedure at the time the procedure is invoked.**
- **Prior to ending, the procedure checks one more time for any new activities to execute, if new processes become stuck or new activities are deferred or timed out after the procedure is initiated.**
- **The procedure ends once all matching activities are executed.**

Background Engine

- If you are using the standalone version of Oracle Workflow, then use the WF_ENGINE.BACKGROUND() API to start up a background engine. Sample scripts that repeatedly run the background engine are provided with the standalone version of Oracle Workflow.
- If you are using the version of Oracle Workflow embedded in Oracle E-Business Suite, start a background engine by submitting the Background Process concurrent program using the Submit Requests form.

Generally, you should run a separate background engine to check for stuck processes at less frequent intervals than the background engine that you run for deferred or timed out activities, normally not more often than once a day. Run the background engine to check for stuck processes when the load on the system is low.



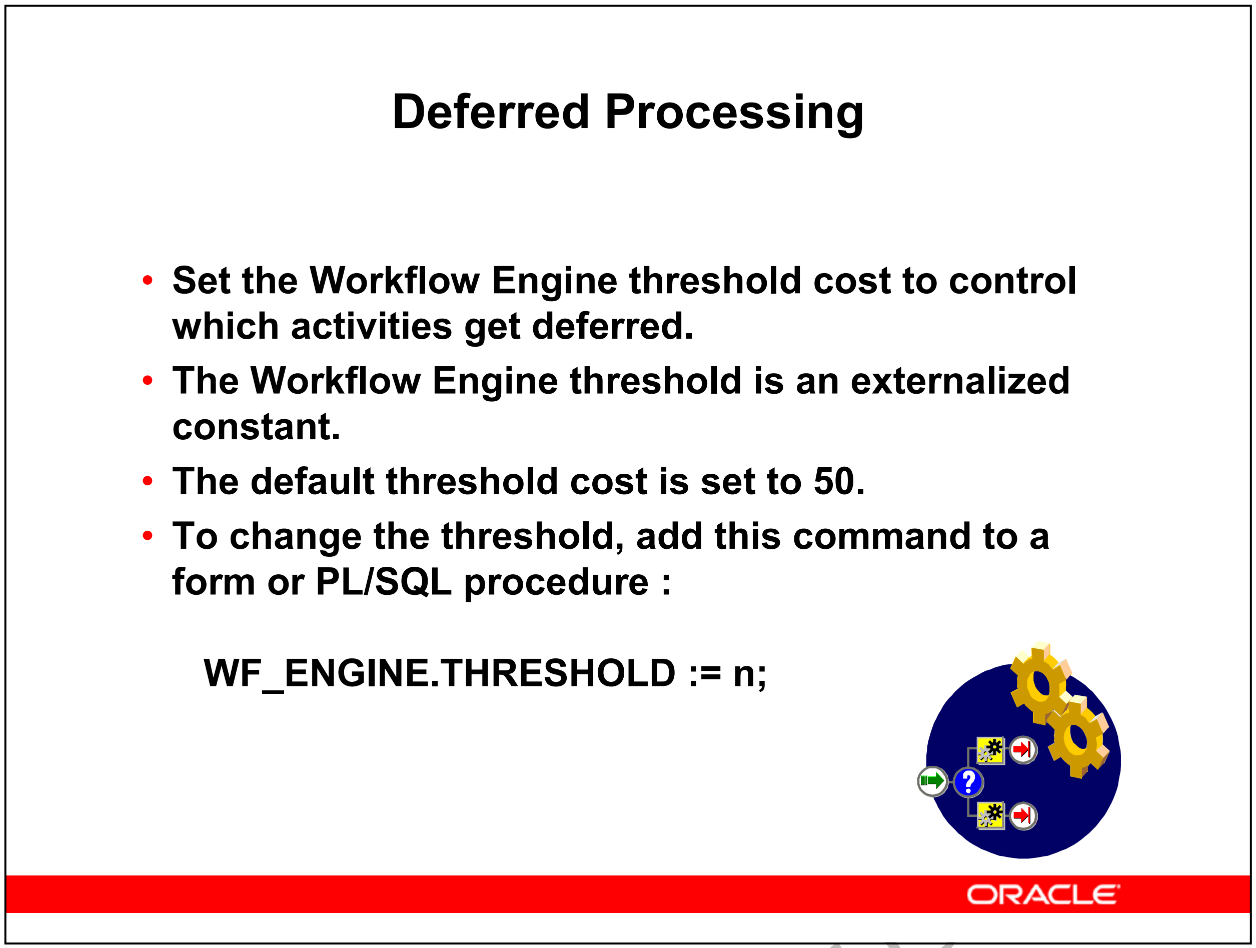
Stuck Processes

- A process is identified as stuck when it has a status of ACTIVE, but cannot progress any further.
- The background engine sets the status of a stuck process to ERROR:#STUCK and executes the error process defined for it.

Stuck Processes

For example, a process could become stuck when:

- A thread within a process leads to an activity that is not defined as an End activity but has no other activity modeled after it, and no other activity is active.
- A process with only one thread loops back, but the pivot activity of the loop has the On Revisit property set to Ignore.
- An activity returns a result for which no eligible transition exists. For instance, if the function for a function activity returns an unexpected result value, and no default transition is modeled after that activity, the process cannot continue.



Deferred Processing

- Set the Workflow Engine threshold cost to control which activities get deferred.
- The Workflow Engine threshold is an externalized constant.
- The default threshold cost is set to 50.
- To change the threshold, add this command to a form or PL/SQL procedure :

WF_ENGINE.THRESHOLD := n;

ORACLE

Deferred Processing

A function activity is deferred to a background engine for processing if its cost exceeds the threshold cost of the Workflow Engine.

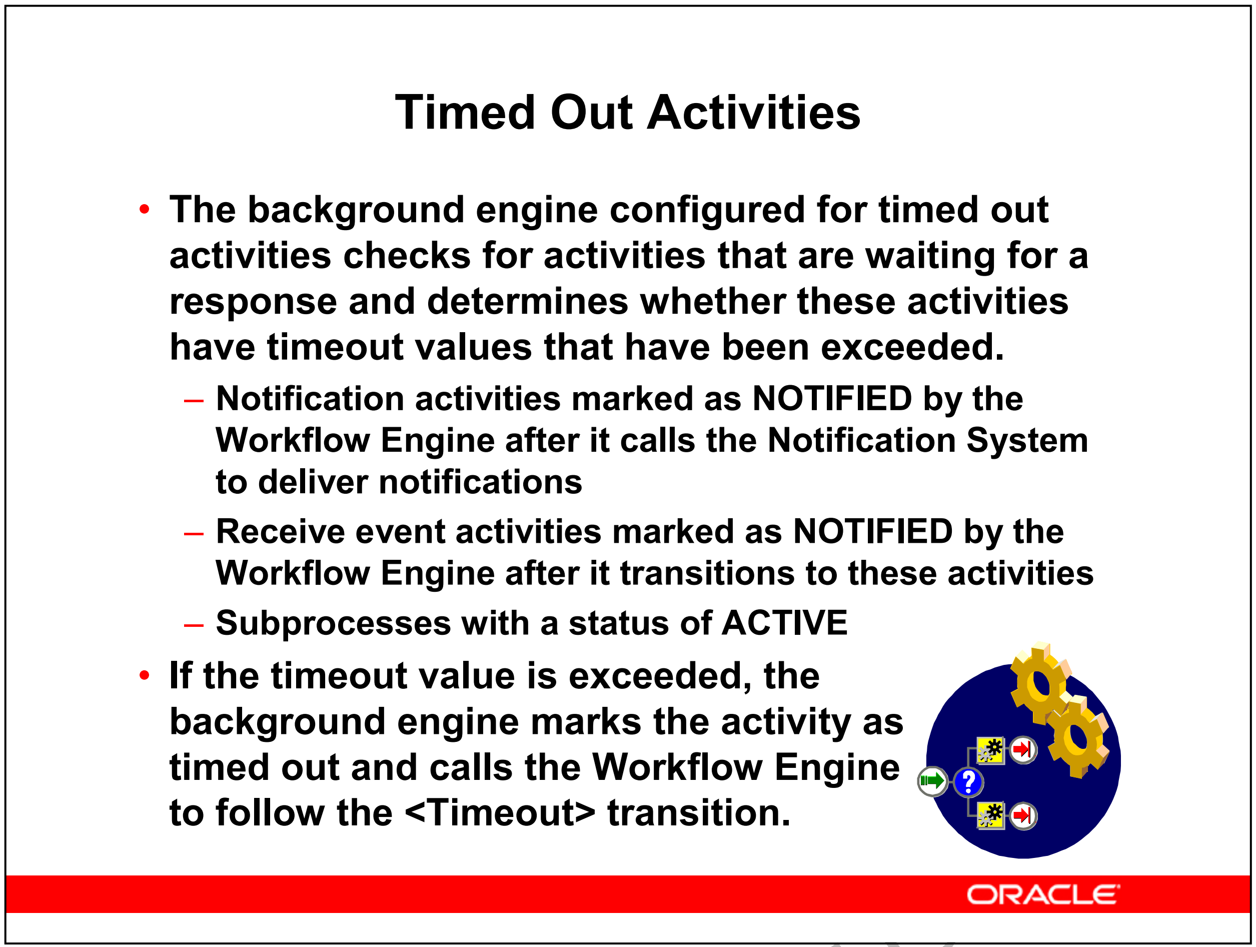
The Workflow Engine integrates with Oracle Advanced Queuing to carry out deferred processing. When the Workflow Engine encounters an activity for deferred processing, a message is written to a separate "deferred" queue. The Background Engine consumes and processes the deferred queue messages, executing and completing the deferred activity.

You can defer a process as soon as it is launched by lowering the Workflow Engine threshold level. For example, if you want to launch a workflow process using the Workflow Engine APIs from a database trigger, you must defer the process immediately in order to avoid issuing savepoints, which are not allowed in a database trigger.

The following example shows how to defer a process immediately upon launching it.

```
-- Set engine to defer everything to the background engine for this session
-- This provides faster user response time, at the expense of
-- delaying workflow progress until the background engine runs.
save_threshold := wf_engine.threshold;
```

```
wf_engine.threshold := -1;

-- Launch the process
wf_engine.CreateProcess(...);
wf_engine.SetItemAttr...(...);
wf_engine.StartProcess(...);

-- Reset the threshold
wf_engine.threshold:=save_threshold;

exception
    -- Ensure threshold is reset
    wf_engine.threshold := save_threshold;
```



Timed Out Activities

- The background engine configured for timed out activities checks for activities that are waiting for a response and determines whether these activities have timeout values that have been exceeded.
  – Notification activities marked as NOTIFIED by the Workflow Engine after it calls the Notification System to deliver notifications
  – Receive event activities marked as NOTIFIED by the Workflow Engine after it transitions to these activities
  – Subprocesses with a status of ACTIVE
- If the timeout value is exceeded, the background engine marks the activity as timed out and calls the Workflow Engine to follow the <Timeout> transition.

Timed Out Activities

You can fix the due date of an activity at design time by setting the timeout value to a relative time in days, hours, and minutes. You can also set the timeout value to an item attribute and specify it when you initiate an instance of the process.

If you are using an item attribute to set the timeout value, then the item attribute must be set to type number or type date. If the item attribute is of type number, the value entered must be in the unit of minutes. If the item attribute is of type date, the value entered must be a date in the format DD-MON-RR HH24:MI:SS.

Note: If a timed out activity does not have a <Timeout> transition modeled, the Workflow Engine will try to find an error process to execute.

Practice - Implementing Timeout Processing

Overview

In this practice, you will implement timeout processing in the Vacation Proposal workflow process you created in the Creating a Workflow Process practice.

- Define timeout processing to transition back to the same notification until a response is received.
- Run a background engine to process timed out activities.
- Define timeout processing to transition to a loop counter and on the third loop exit to approve the proposal automatically.
- Run a background engine again to process timed out activities.

Note: Because many students access the system and create objects during this course, you need a way to distinguish between the objects created by you and by your classmates. Therefore, you will be assigned a terminal number by your instructor. Use this number as a prefix wherever you see *XX* included in the name of something you are defining. In this way, you can ensure that the definitions you create are unique.

Assumptions

- The instructor will provide you with the connect string for the class database and the username and password of the Oracle Workflow database account.
- The instructor will provide you with the username and password of a user with workflow administrator privileges. The workflow administrator is defined in the Global Workflow Preferences page.
- For standalone Oracle Workflow, the instructor will provide you with the URL for the Oracle Workflow home page. The URL is <*webagent*>/wfa_html.home, where <*webagent*> is the base URL of the web agent configured for Oracle Workflow in your Web server.
- For Oracle Workflow embedded in Oracle E-Business Suite, the instructor will provide you with the name of a Workflow administrator responsibility. The username you use to log in should have this responsibility assigned to it.
- The instructor will provide you with the names of users that you can assign as the requestor and approver in the Vacation Proposal process.

Tasks

1. Define timeout processing to transition back to the same notification until a response is received.

Solution:

1. Start the Oracle Workflow Builder.

2. From the File menu, choose Open to open the wfvac*XX*.wft data store you defined in the Creating a Workflow Process practice.

3. Open the process diagram window for the Vacation Proposal process and open the property pages for the Vacation Proposal notification activity node.

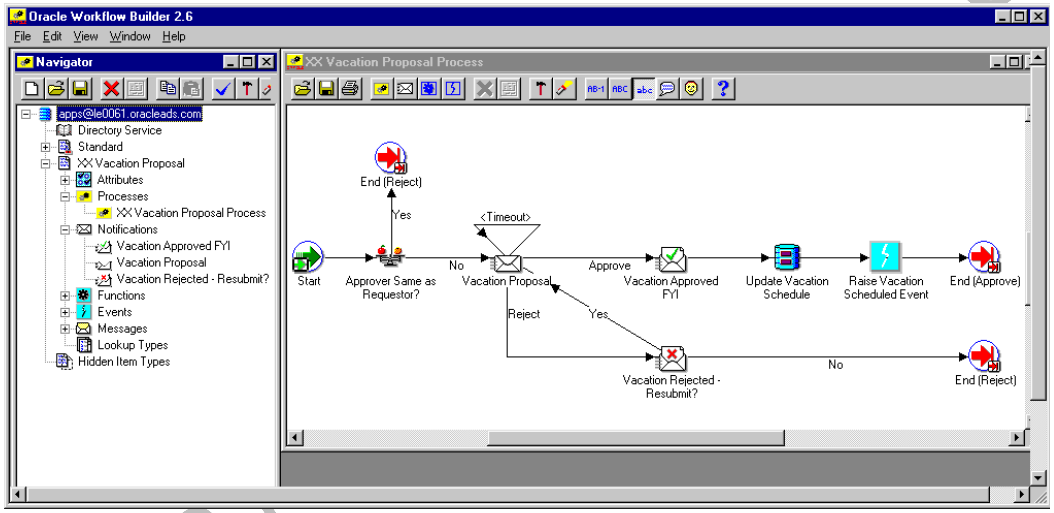
4. Choose the Node tab. In the Timeout region, select Relative Time as the type. For the value, enter a short duration such as 1 minute for testing purposes. Choose OK.

5. Create a timeout transition from the Vacation Proposal notification back to itself. To do so, select the Vacation Proposal node and hold down the right mouse button. Drag the cursor away from the notification and then back to the notification, and then release the right mouse button. Select <Timeout> from the transition results menu.

6. In the Navigator window, click the Verify button to verify your workflow.

7. From the File menu, choose Save to save your work to your workflow definition file.

8. From the File menu, choose Save As and save your item type to the class database, using the database username, password, and connect string provided by the instructor.



9. Use a web browser to connect to the Oracle Workflow home page with the URL provided by the instructor for standalone Oracle Workflow, or to a Workflow administrator responsibility provided by the instructor for Oracle Workflow embedded in Oracle E-Business Suite. Log in as a user with workflow administrator privileges.

10. Use the Launch Processes page to launch your workflow process. You can use the Notifications Worklist to view the notifications sent by the process and use the Workflow Monitor to review the status of the process. Verify your work by reviewing the Vacation Proposal notification in the Notification Details web page. The message header should now display both a sent date and a due date. The interval between these times should be equal to the relative time value you entered for the notification timeout. To test your work, do not respond to the Vacation Proposal notification, but allow the notification activity to time out instead by waiting for the timeout interval you specified to elapse.

2. Run a background engine to process timed out activities.

Solution:

11. Run a background engine for your item type to process the timed out activity and allow the process to continue. In Oracle E-Business Suite, you can use the Workflow Background Process concurrent program to run a background engine. You can also run a background engine using the WF_ENGINE.Background API.

12. To run the Workflow Background Process concurrent program, log in as the system administrator and open the Oracle E-Business Suite Submit Requests form. Choose to submit a single request and select the Workflow Background Process as the request to run. Enter the following parameters:
    - Item Type: <your item type display name>
    - Process Deferred: No
    - Process Timeout: Yes
    - Process Stuck: No

13. To run the WF_ENGINE.Background API, open SQL*Plus and enter the following command:

    SQL> Exec WF_ENGINE.Background('WFVAC*XX*', NULL, NULL, FALSE, TRUE, FALSE);

14. Review the new notification and check the sent and due dates to verify that this notification was sent after the process followed the timeout transition. (The first notification is canceled and will no longer appear in the Worklist.)

3. Define timeout processing to transition to a loop counter and on the third loop exit to approve the proposal automatically.

Solution:

15. Start the Oracle Workflow Builder, open the wfvac*XX*.wft data store, and open the process diagram window for the Vacation Proposal process.

16. Delete the transition from the Vacation Proposal node to itself so that you can implement another type of timeout processing.

17. Select the Loop Counter function activity in the Standard item type. To locate the Loop Counter activity, you can choose Find from the Edit menu or press Ctrl+F. Enter 'Loop Counter' in the Search Text field, select the Display Name and Function check boxes, and click the Search button. The Loop Counter activity will be automatically selected. Close the Find window.

18. Drag and drop the Loop Counter activity into the Vacation Proposal process diagram, positioning it above the Vacation Proposal notification node.

19. Open the property pages for the Loop Counter node and choose the Node Attributes tab to set the number of times the Workflow Engine should execute the loop. Select the Loop Limit attribute. Select the type Constant and enter the value 2 for the attribute. Choose OK.

20. Draw a transition from the Vacation Proposal node to the Loop Counter node and select <Timeout> from the transition results menu.

21. Create a vertex point in the transition between the Vacation Proposal node and the Loop Counter node.

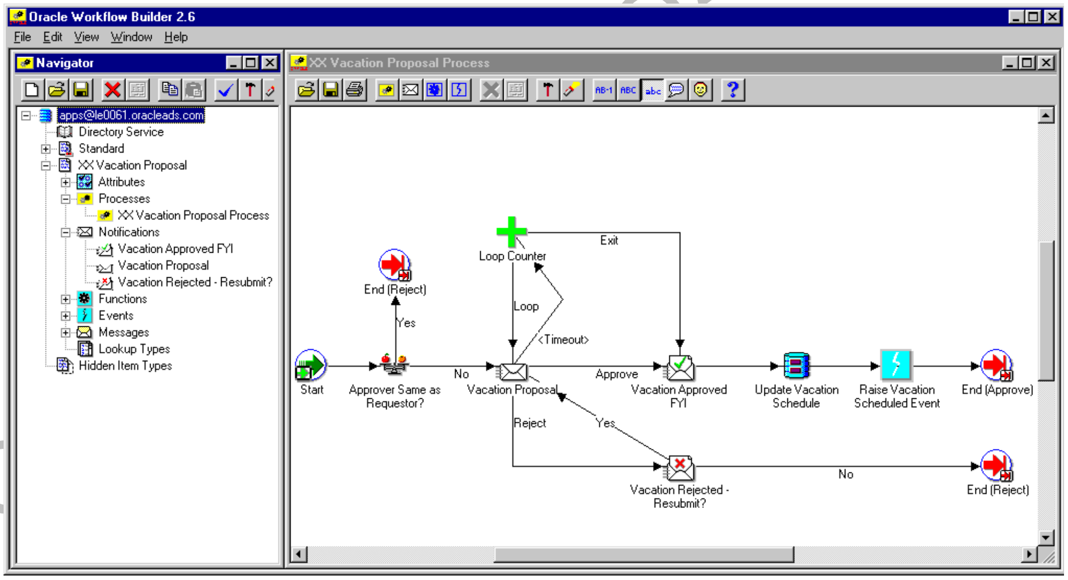
22. Draw a transition from the Loop Counter node to the Vacation Proposal node and select Loop from the transition results menu.

23. Draw a transition from the Loop Counter node to the Vacation Approved FYI node and select Exit from the transition results menu.

24. In the Navigator window, click the Verify button to verify your workflow.

25. From the File menu, choose Save to save your work to your workflow definition file.

26. From the File menu, choose Save As and save your item type to the class database, using the database username, password, and connect string provided by the instructor.



27. Use a web browser to connect to the Oracle Workflow home page with the URL provided by the instructor for standalone Oracle Workflow, or to a Workflow administrator responsibility provided by the instructor for Oracle Workflow embedded in Oracle E-Business Suite. Log in as a user with workflow administrator privileges.

28. Use the Launch Processes page to launch your workflow process. You can use the Notifications Worklist to view the notifications sent by the process and use the Workflow Monitor to review the status of the process. To test your work, do not respond to the Vacation Proposal notification, but allow the notification activity to time out instead by waiting for the timeout interval you specified to elapse.

4. Run a background engine to process timed out activities.

Solution:

29. Run a background engine for your item type to process the timed out activity and allow the process to continue. In Oracle E-Business Suite, you can use the Workflow Background Process concurrent program to run a background engine. You can also run a background engine using the WF_ENGINE.Background API. Run the background engine repeatedly to process timed out activities for each execution of the loop.

30. To run the Workflow Background Process concurrent program, log in as the system administrator and open the Oracle E-Business Suite Submit Requests form. Choose to submit a single request and select the Workflow Background Process as the request to run. Enter the following parameters:
    - Item Type: <your item type display name>
    - Process Deferred: No
    - Process Timeout: Yes
    - Process Stuck: No

31. To run the WF_ENGINE.Background API, open SQL*Plus and enter the following command:

    SQL> Exec WF_ENGINE.Background('WFVAC*XX*', NULL, NULL, FALSE, TRUE, FALSE);

32. Use the Workflow Monitor to review the status of the process and verify that the process followed the timeout transition to execute the Loop Counter activity.

33. Allow the Vacation Proposal notification activity to time out again. Then run a background engine again to process the timed out activity.

34. Use the Workflow Monitor to review the status of the process and verify that the process followed the timeout transition to execute the Loop Counter activity a second time.

35. Allow the Vacation Proposal notification activity to time out again. Then run a background engine again to process the timed out activity.

36. Use the Workflow Monitor to review the status of the process and verify that the process followed the timeout transition to execute the Loop Counter activity a third time and then followed the exit transition to execute the Vacation Approved FYI activity and the rest of the process.

Practice - Implementing Deferred Processing

Overview

In this practice, you will implement deferred processing for a function activity in the Vacation Proposal workflow process you created in the Creating a Workflow Process practice.

- Set the function activity cost above the engine threshold to defer the activity when the process is executed.
- Run a background engine to process deferred activities.

Note: Because many students access the system and create objects during this course, you need a way to distinguish between the objects created by you and by your classmates. Therefore, you will be assigned a terminal number by your instructor. Use this number as a prefix wherever you see *XX* included in the name of something you are defining. In this way, you can ensure that the definitions you create are unique.

Assumptions

- The instructor will provide you with the connect string for the class database and the username and password of the Oracle Workflow database account.
- The instructor will provide you with the username and password of a user with workflow administrator privileges. The workflow administrator is defined in the Global Workflow Preferences page.
- For standalone Oracle Workflow, the instructor will provide you with the URL for the Oracle Workflow home page. The URL is <*webagent*>/wfa_html.home, where <*webagent*> is the base URL of the web agent configured for Oracle Workflow in your Web server.
- For Oracle Workflow embedded in Oracle E-Business Suite, the instructor will provide you with the name of a Workflow administrator responsibility. The username you use to log in should have this responsibility assigned to it.
- The instructor will provide you with the names of users that you can assign as the requestor and approver in the Vacation Proposal process.

Tasks

1. Set the function activity cost above the engine threshold to defer the activity when the process is executed.

   Solution:

   1. Start the Oracle Workflow Builder.

2. From the File menu, choose Open to open the wfvac*XX*.wft data store you defined in the Creating a Workflow Process practice.

3. Open the property pages for the Update Vacation Schedule function activity. Change the cost for the activity to 100. Since this cost is above the default engine threshold of 50, the activity will be deferred when the process is executed.

4. In the Navigator window, click the Verify button to verify your workflow.

5. From the File menu, choose Save to save your work to your workflow definition file.

6. From the File menu, choose Save As and save your item type to the class database, using the database username, password, and connect string provided by the instructor.

7. Use a web browser to connect to the Oracle Workflow home page with the URL provided by the instructor for standalone Oracle Workflow, or to a Workflow administrator responsibility provided by the instructor for Oracle Workflow embedded in Oracle E-Business Suite. Log in as a user with workflow administrator privileges.

8. Use the Launch Processes page to launch your workflow process. You can use the Notifications Worklist to view the notifications sent by the process and use the Workflow Monitor to review the status of the process.

9. To test your work, log in as the approver and approve the Vacation Proposal notification.

10. Log in as the requestor and use the Workflow Monitor to view the diagram for the process. The process should be stopped at the Update Vacation Schedule function activity.

2. Run a background engine to process deferred activities.

Solution:

11. Run a background engine for your item type to process the deferred activity and allow the process to continue. In Oracle E-Business Suite, you can use the Workflow Background Process concurrent program to run a background engine. You can also run a background engine using the WF_ENGINE.Background API.

12. To run the Workflow Background Process concurrent program, log in as the system administrator and open the Oracle E-Business Suite Submit Requests form. Choose to submit a single request and select the Workflow Background Process as the request to run. Enter the following parameters:
    – Item Type: <your item type display name>
    – Process Deferred: Yes
    – Process Timeout: No
    – Process Stuck: No

13. To run the WF_ENGINE.Background API, open SQL*Plus and enter the following command:

SQL> Exec WF_ENGINE.Background('WFVAC*XX*', NULL, NULL, TRUE, FALSE, FALSE);

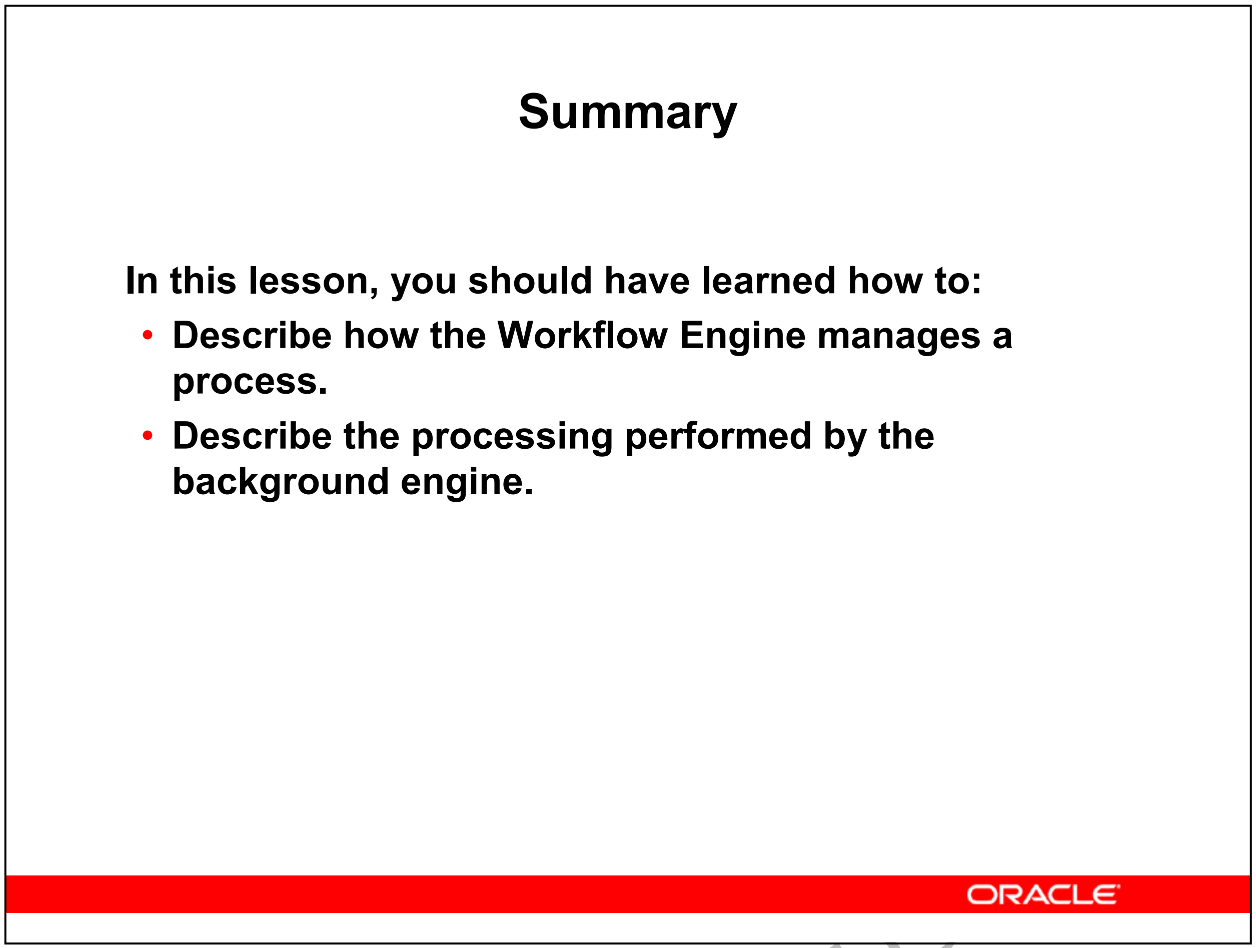
14. Log in as the requestor and use the Workflow Monitor to view the diagram for the process. The process should now have completed.

15. To remove the deferred processing so that you will not need to run the background engine in later practices, start the Oracle Workflow Builder, open the wfvac*XX*.wft data store, open the property pages for the Update Vacation Schedule function activity, and change the cost for the activity back to 0.

16. In the Navigator window, click the Verify button to verify your workflow.

17. From the File menu, choose Save to save your work to your workflow definition file.

18. From the File menu, choose Save As and save your item type to the class database, using the database username, password, and connect string provided by the instructor.



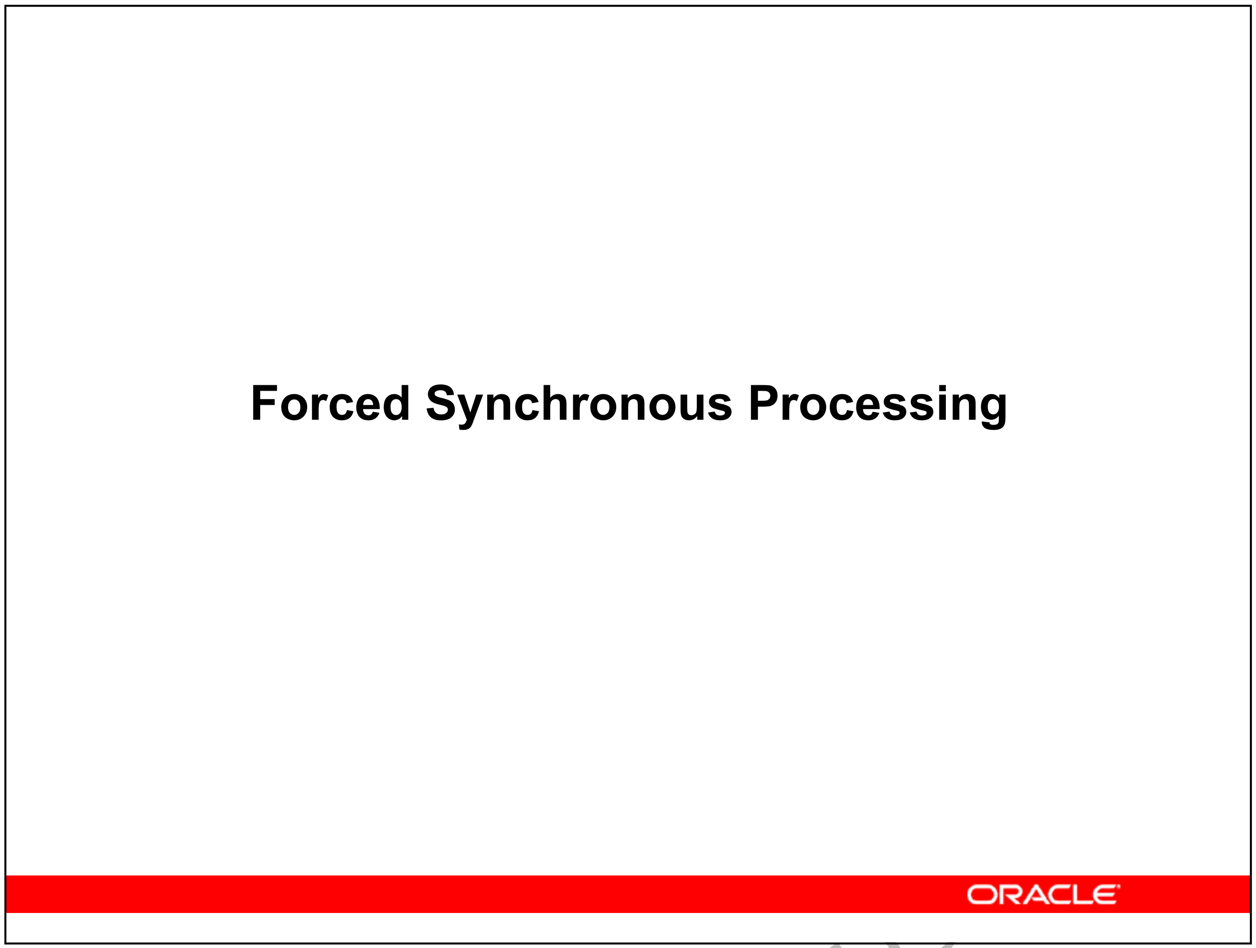
Summary

In this lesson, you should have learned how to:

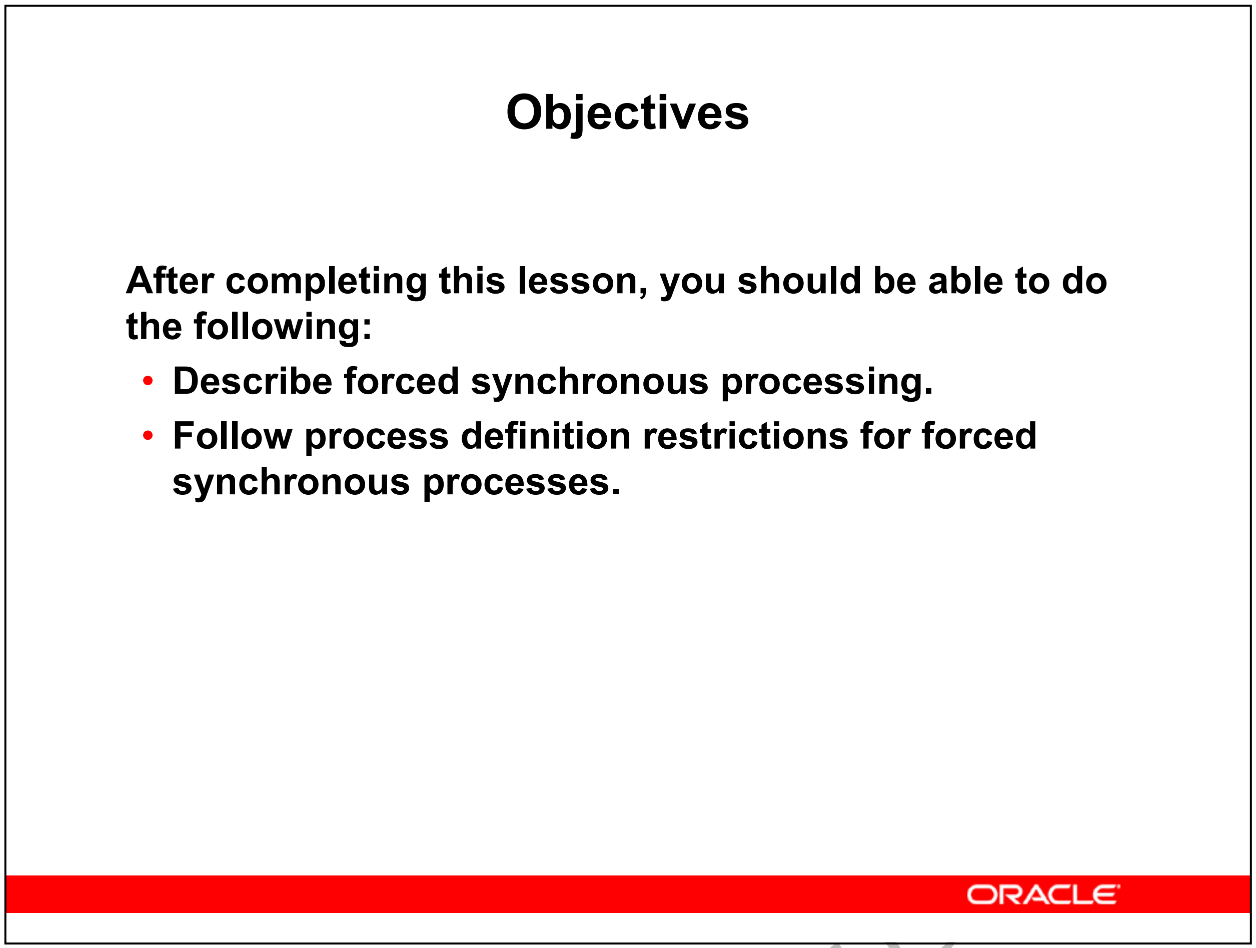
- **Describe how the Workflow Engine manages a process.**
- **Describe the processing performed by the background engine.**

Forced Synchronous Processing

Chapter 15



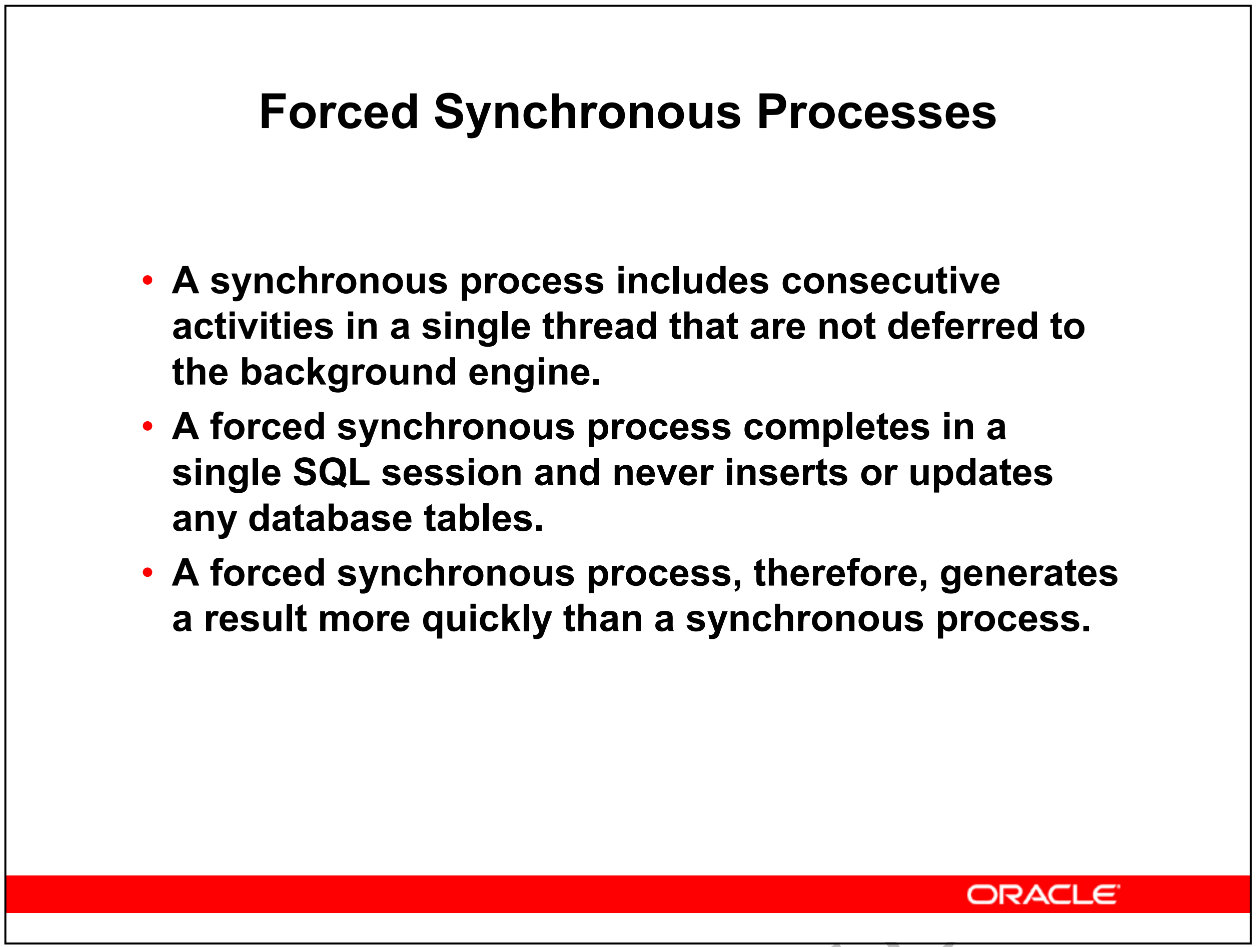
Forced Synchronous Processing



Objectives

After completing this lesson, you should be able to do the following:

- Describe forced synchronous processing.
- Follow process definition restrictions for forced synchronous processes.



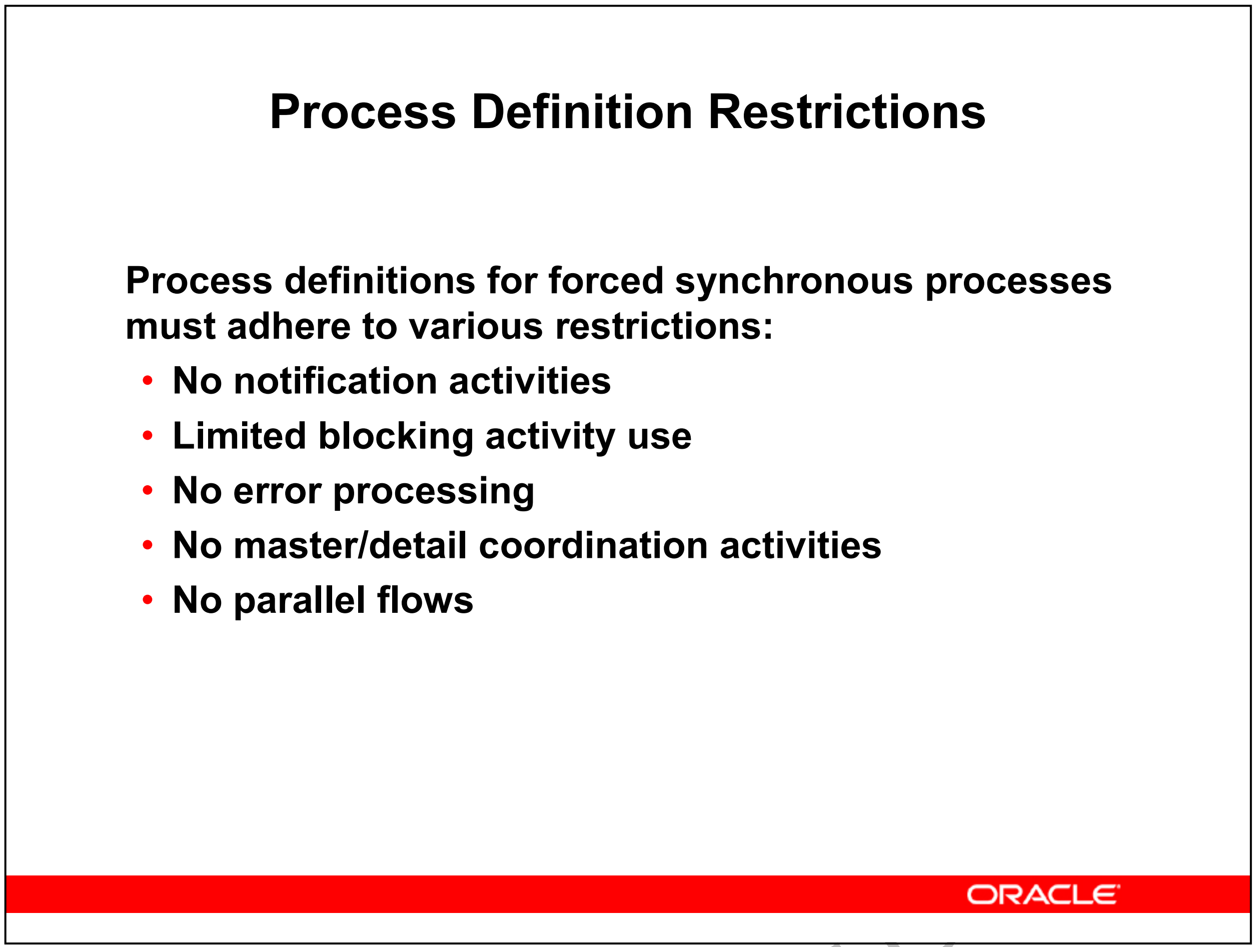
Forced Synchronous Processes

- A synchronous process includes consecutive activities in a single thread that are not deferred to the background engine.
- A forced synchronous process completes in a single SQL session and never inserts or updates any database tables.
- A forced synchronous process, therefore, generates a result more quickly than a synchronous process.

Forced Synchronous Processing

To create a forced synchronous process, you must set the item key of your process to #SYNCH, or to wf_engine.eng_synch, which returns the #SYNCH constant, when you call the necessary WF_ENGINE APIs. Because a forced synchronous process never writes to the database, using a non-unique item key such as #SYNCH is not an issue.

Also because a forced synchronous process never writes to the database, it completes faster than a normal synchronous process. However, for the same reason, the process cannot be viewed from the Workflow Monitor, and no auditing is available for the process.



Process Definition Restrictions

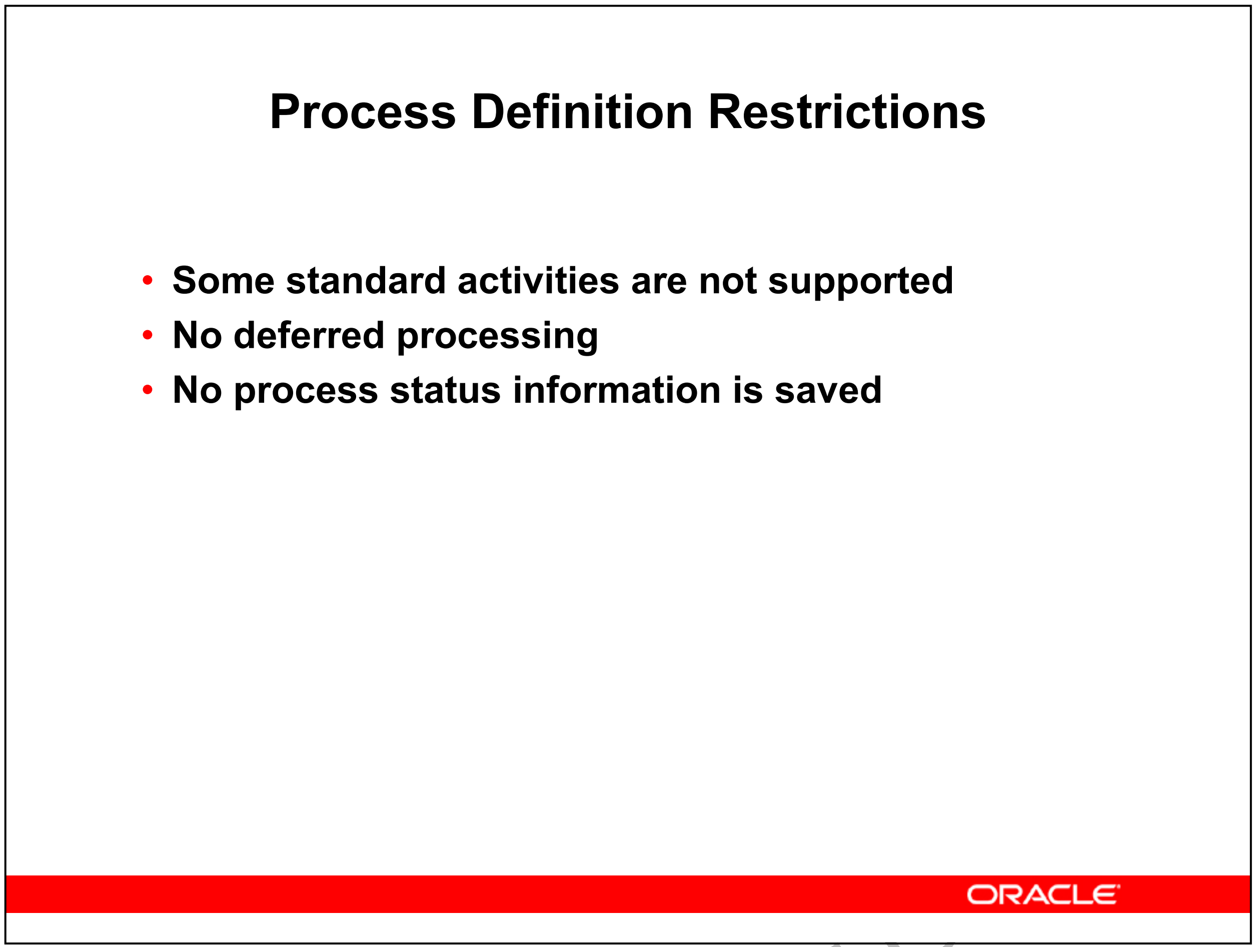
Process definitions for forced synchronous processes must adhere to various restrictions:

- **No notification activities**
- **Limited blocking activity use**
- **No error processing**
- **No master/detail coordination activities**
- **No parallel flows**

ORACLE

Process Definition Restrictions

- No notification activities are allowed.
- Limited blocking-type activities are allowed. A process can block and restart with a call to WF_ENGINE.CompleteActivity only if the blocking and restarting activities:
  - Occur in the same database session
  - Contain no intervening calls to Oracle Workflow
  - Contain no intervening commits
- No error processes can be assigned to the process or the process's activities.
- Each function activity behaves as if On Revisit is set to Loop, and is run in non-canceling mode, regardless of its actual On Revisit setting. Loops are allowed in the process.
- No master/detail coordination activities are allowed.
- No parallel flows are allowed in the process, as transitions from each activity must have a distinct result. This also means that no <Any> transitions are allowed, because they cause parallel flows.

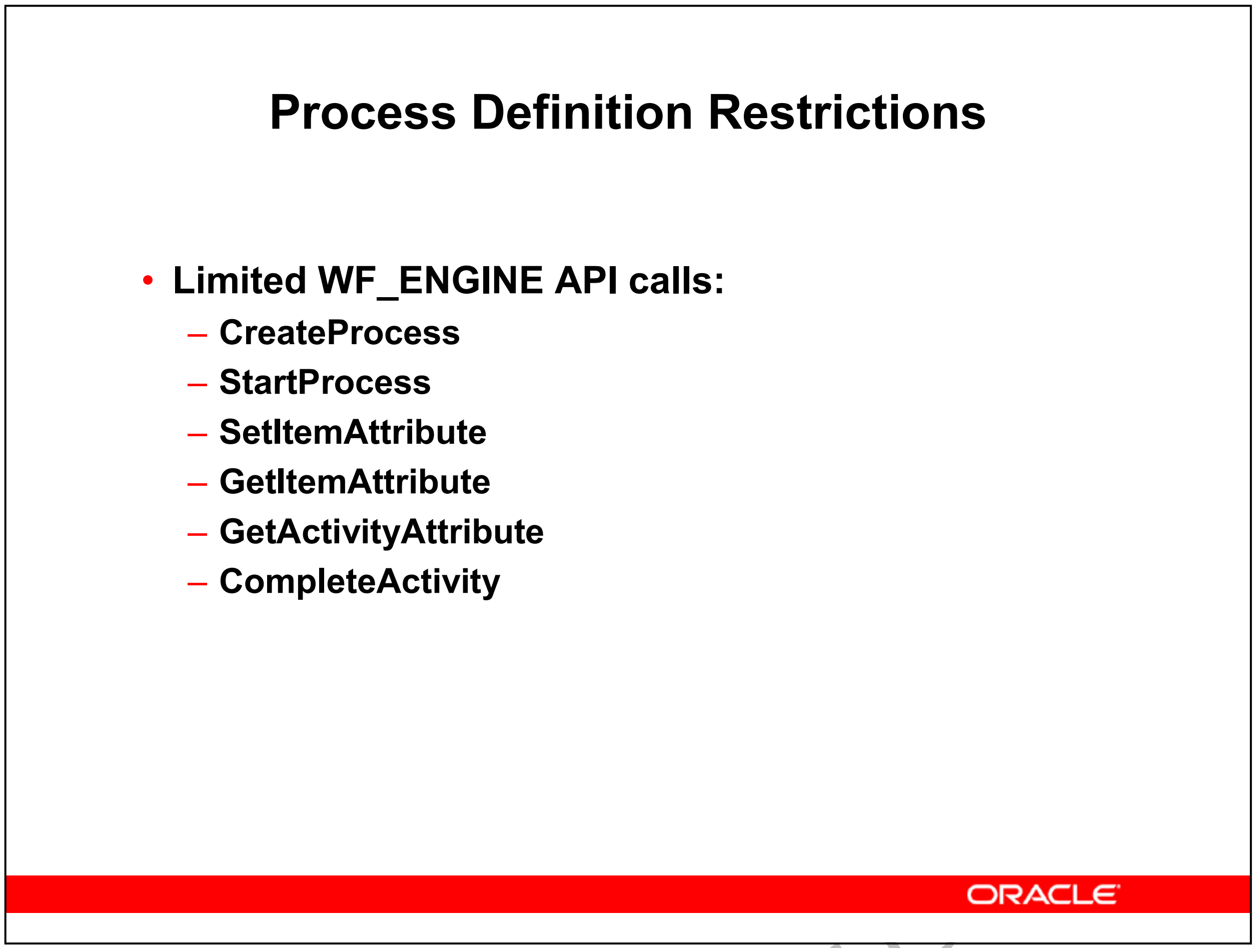


Process Definition Restrictions

- Some standard activities are not supported
- No deferred processing
- No process status information is saved

Process Definition Restrictions

- None of the following Standard activities are allowed:
  - And
  - Block (restricted by the conditions stated for limited blocking)
  - Defer Thread
  - Wait
  - Continue Flow/Wait for Flow
  - Role Resolution
  - Voting
  - Compare Execution Time
  - Notify
- No use of the background engine is allowed; that is, activities are never deferred.
- No data is ever written to the Oracle Workflow tables, and as a result:
  - The process cannot be viewed from the Workflow Monitor.
  - No auditing is available for the process.



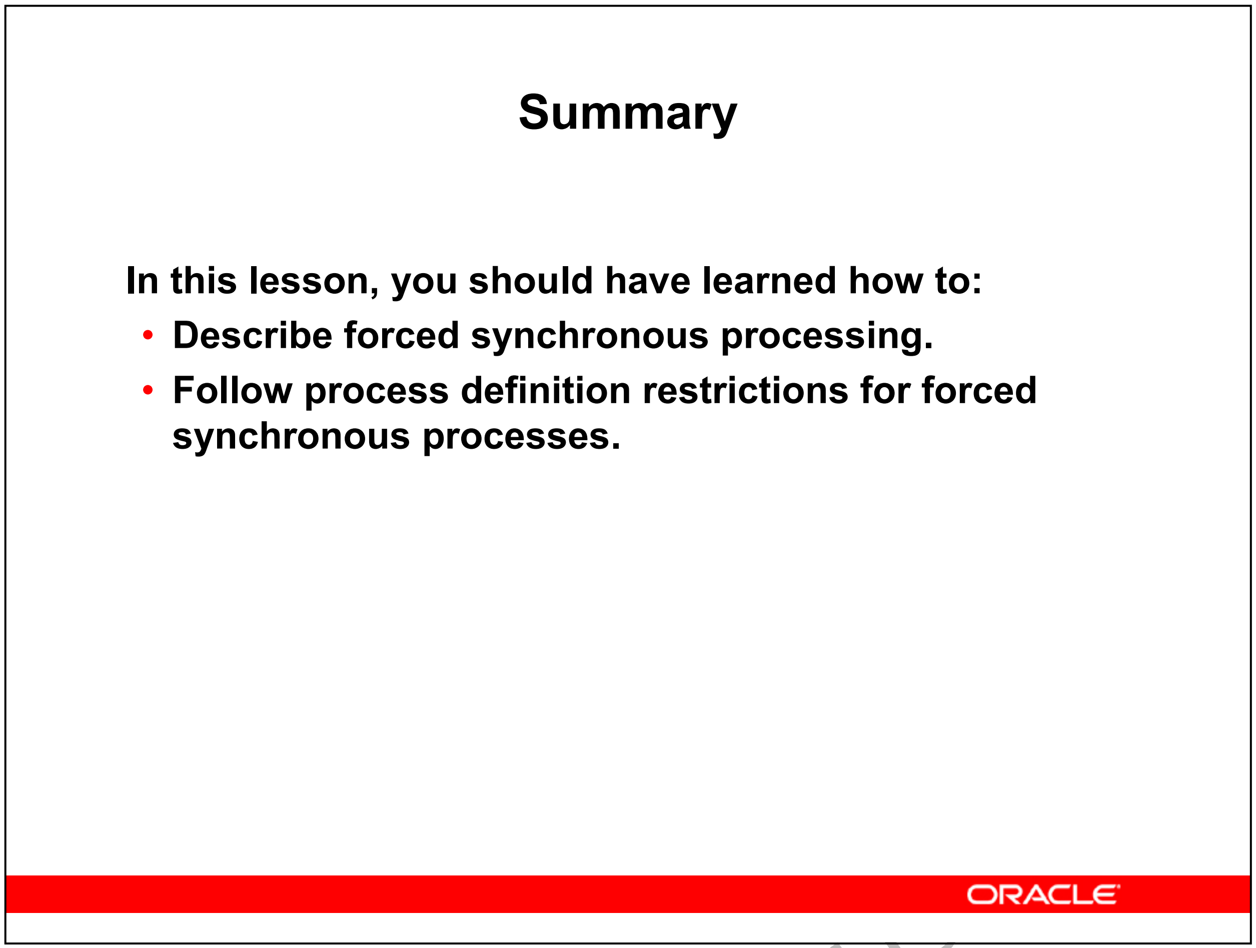
Process Definition Restrictions

- **Limited WF_ENGINE API calls:**
  - **CreateProcess**
  - **StartProcess**
  - **SetItemAttribute**
  - **GetItemAttribute**
  - **GetActivityAttribute**
  - **CompleteActivity**

Process Definition Restrictions

- Only the following WF_ENGINE API calls are allowed to be made, and in all cases, the item key supplied to these APIs must be specified as #SYNCH or wf_engine.eng_synch:
  - WF_ENGINE.CreateProcess
  - WF_ENGINE.StartProcess
  - WF_ENGINE.GetItemAttribute
  - WF_ENGINE.SetItemAttribute
  - WF_ENGINE.GetActivityAttribute
  - WF_ENGINE.CompleteActivity (for the limited usage of blocking-type activities)
- WF_ENGINE API calls for any item except the current synchronous item are not allowed.

Summary



Summary

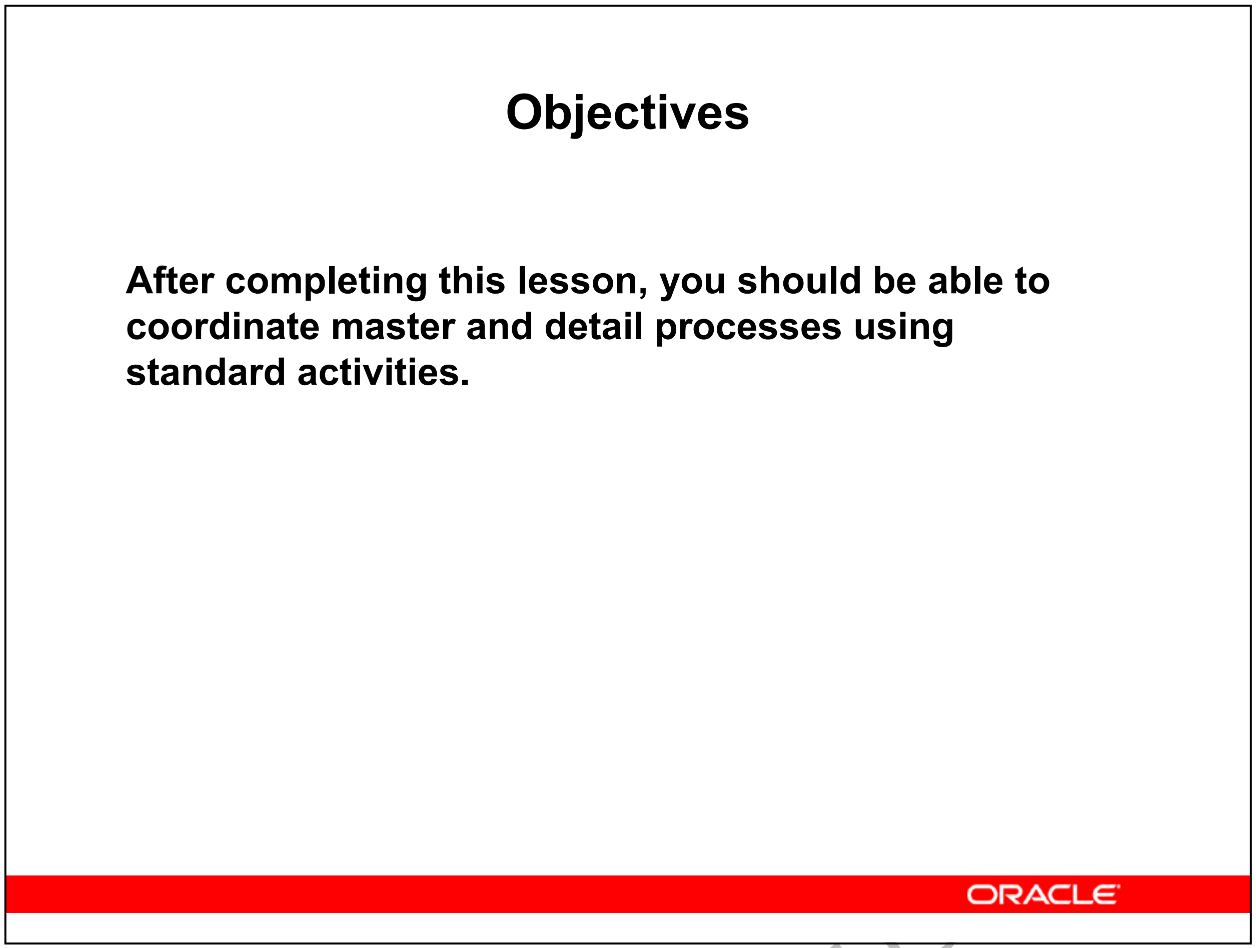
In this lesson, you should have learned how to:

- Describe forced synchronous processing.
- Follow process definition restrictions for forced synchronous processes.

Master/Detail Coordination Activities

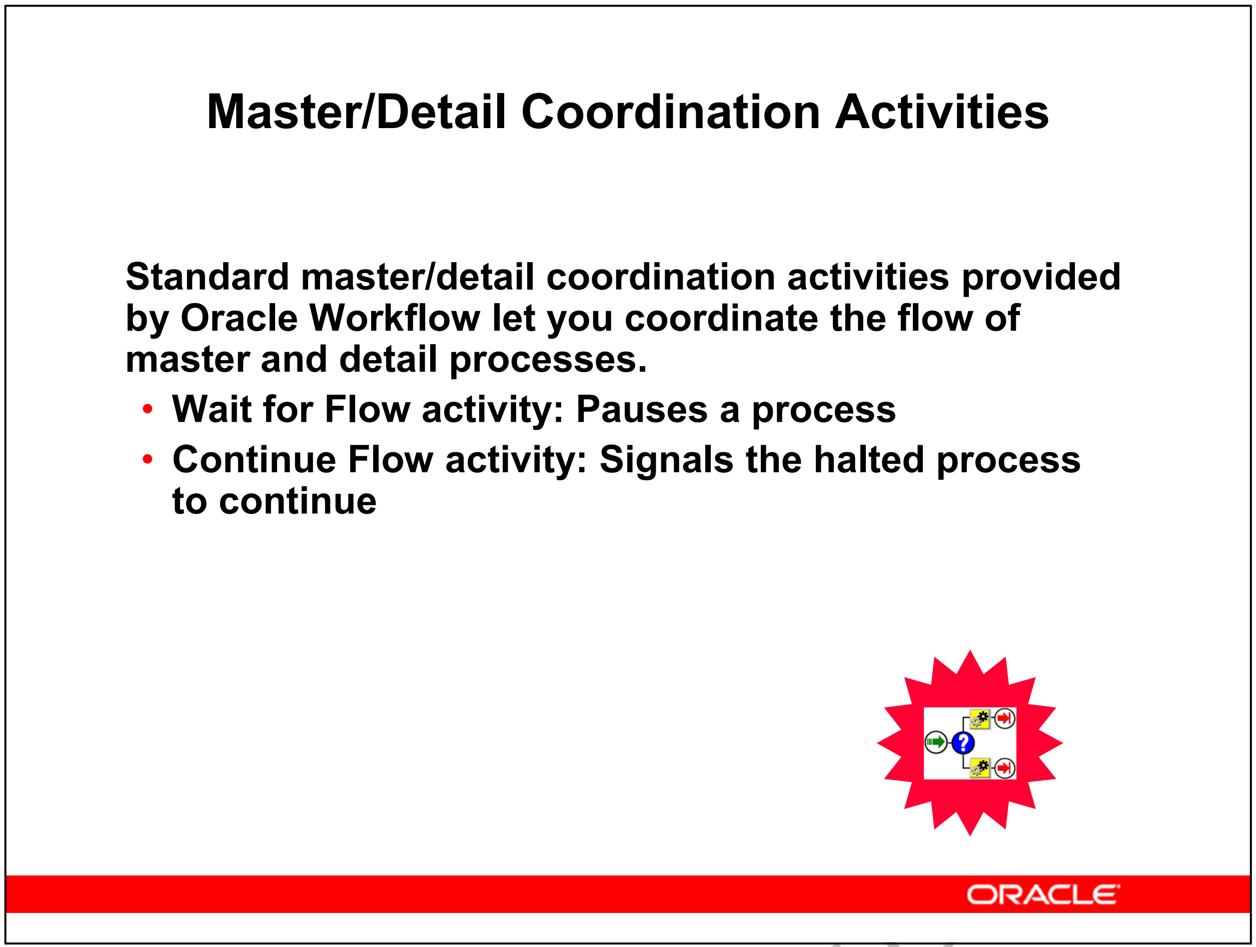
Chapter 16

Master/Detail Coordination Activities



Objectives

After completing this lesson, you should be able to coordinate master and detail processes using standard activities.



Master/Detail Coordination Activities

Standard master/detail coordination activities provided by Oracle Workflow let you coordinate the flow of master and detail processes.

- Wait for Flow activity: Pauses a process
- Continue Flow activity: Signals the halted process to continue

Master/Detail Coordination Activities

When you spawn a detail process from a master process, you are in effect creating a separate process with its own unique item type and item key. You define the master/detail relationship between the two processes by calling the Workflow Engine SetItemParent API after you call the CreateProcess API and before you call the StartProcess API when you create the detail process.

Two activities are used to coordinate the flow in the master and detail processes. One activity is placed in the master process, the other in the detail process. Each of the activities contains two attributes used to identify the coordinating activity in the other process.

Wait for Flow Activity

Place this activity in a master or detail process to pause the flow until the other corresponding detail or master process completes a specified activity. This activity calls the PL/SQL procedure WF_STANDARD.WAITFORFLOW.

The Wait for Flow activity contains two attributes:

- Continuation Flow: Specify whether this activity is waiting for a corresponding ‘Master’ or ‘Detail’ process to complete.

- Continuation Activity: Specify the label of the activity node that must complete in the corresponding process before the current process continues. The default value is CONTINUEFLOW.

Continue Flow Activity

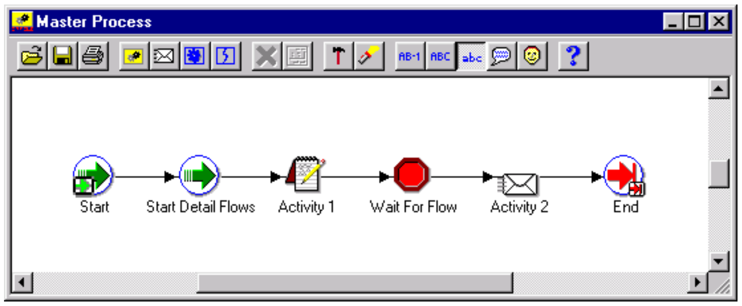
Use this activity to mark the position in the corresponding detail or master process where, upon completion, the halted process is to continue. This activity calls the PL/SQL procedure WF_STANDARD.CONTINUEFLOW.

The Continue Flow activity contains two attributes:

- Waiting Flow: Specify whether the halted process which is waiting for this activity to complete is a ‘Master’ or ‘Detail’ flow.
- Waiting Activity: Specify the label of the activity node in the halted process that is waiting for this activity to complete.

For more information, refer to the Predefined Workflow Activities chapter in the *Oracle Workflow Guide*.

Master Process Example

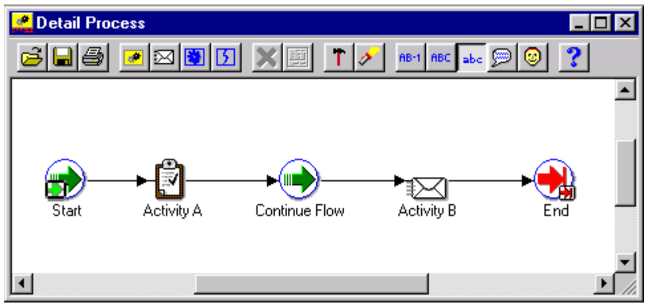


ORACLE

Master Process Example

In the master process above, the Start Detail Flows activity initiates several detail processes. The master process then completes Activity 1 before it pauses at the Wait for Flow activity. The Wait for Flow activity is defined to wait for all its detail processes to complete a Continue Flow activity before allowing the master process to transition to Activity 2.

Detail Process Example

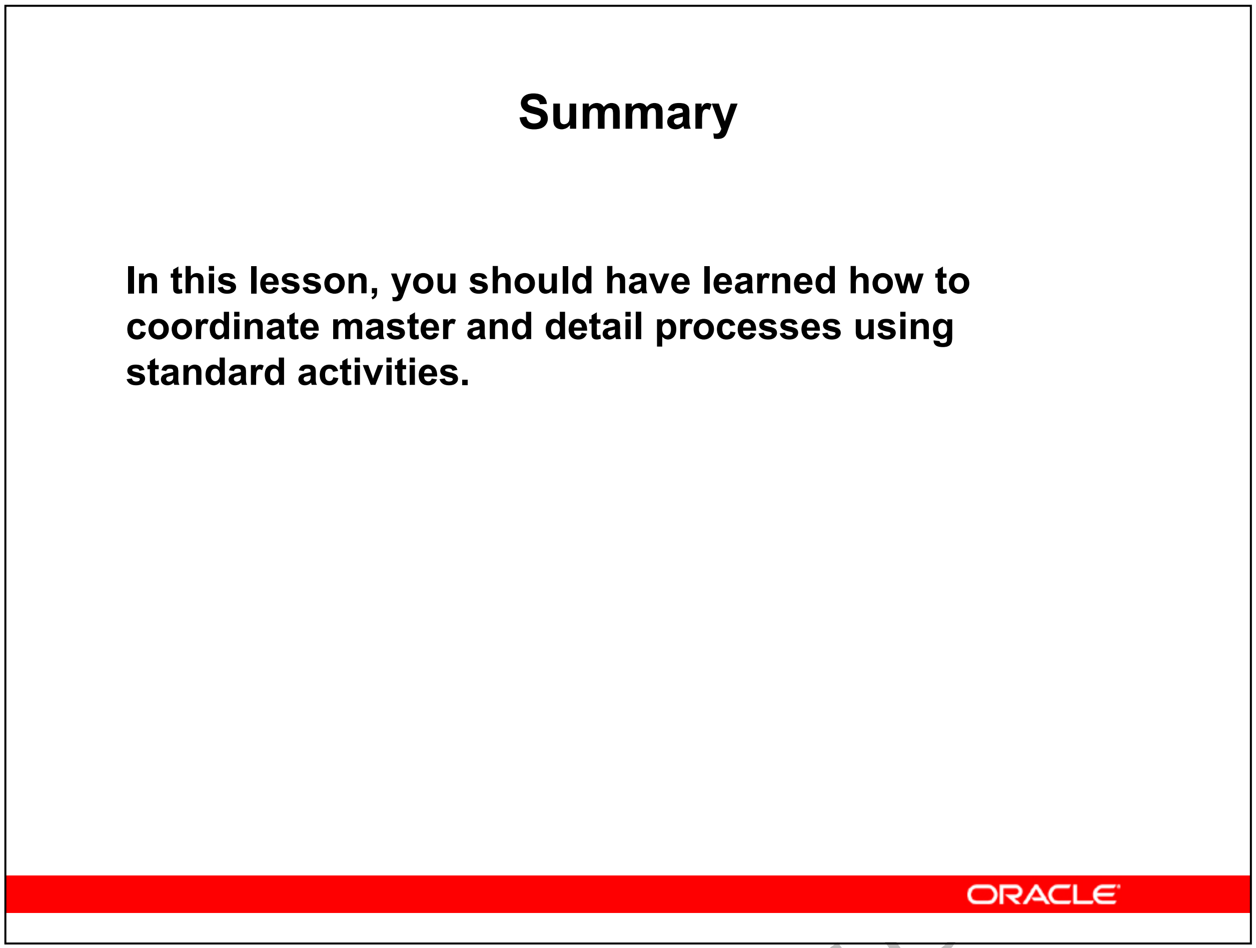


ORACLE

Detail Process Example

When a detail process begins, it completes Activity A. When the process reaches the Continue Flow activity, it signals to the Workflow Engine that the master process can now continue from the Wait for Flow activity. The detail process itself then transitions to Activity B.

Summary



Summary

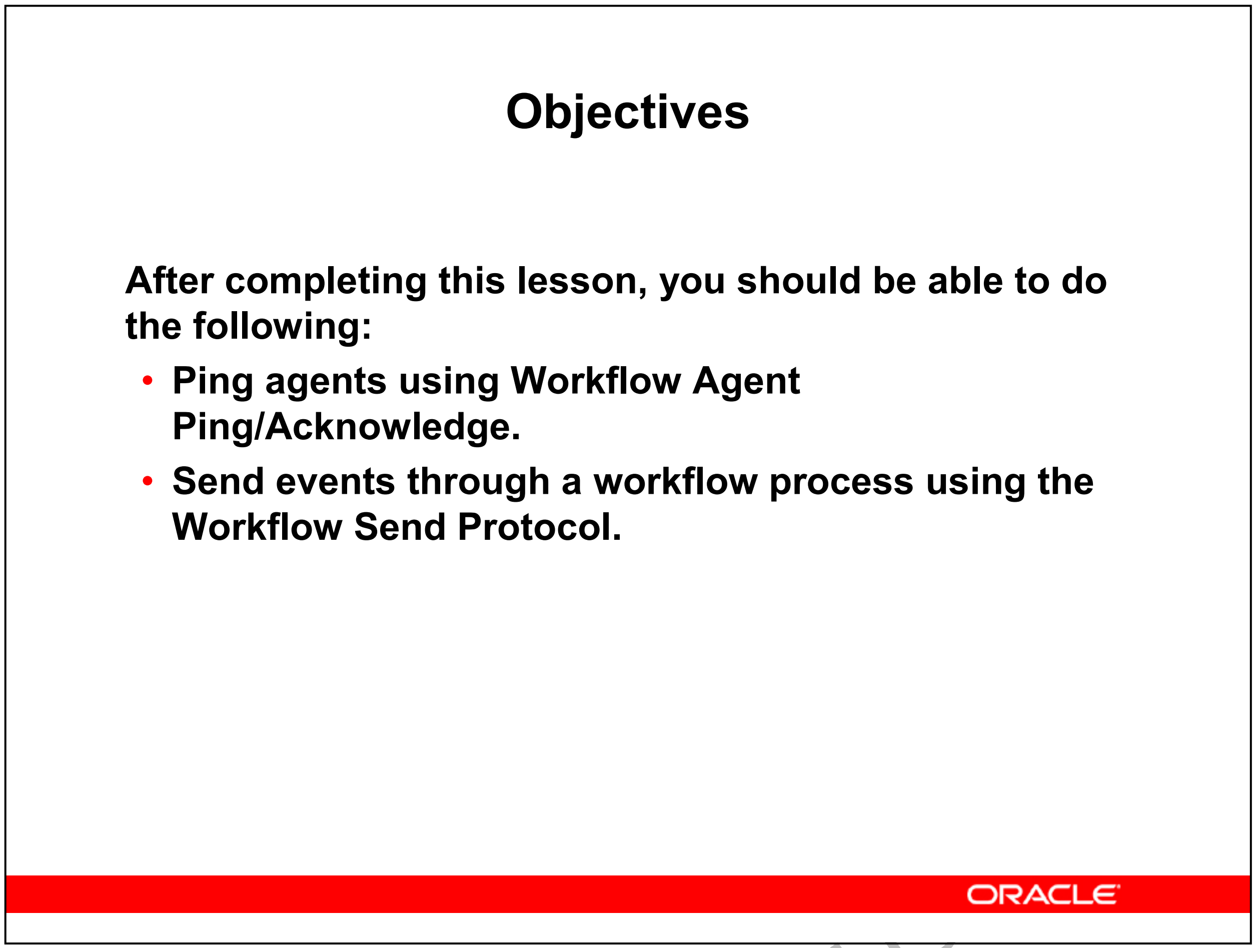
In this lesson, you should have learned how to coordinate master and detail processes using standard activities.

Sample Business Event-Based Workflow Processes

Chapter 17

Sample Business Event-Based Workflow Processes

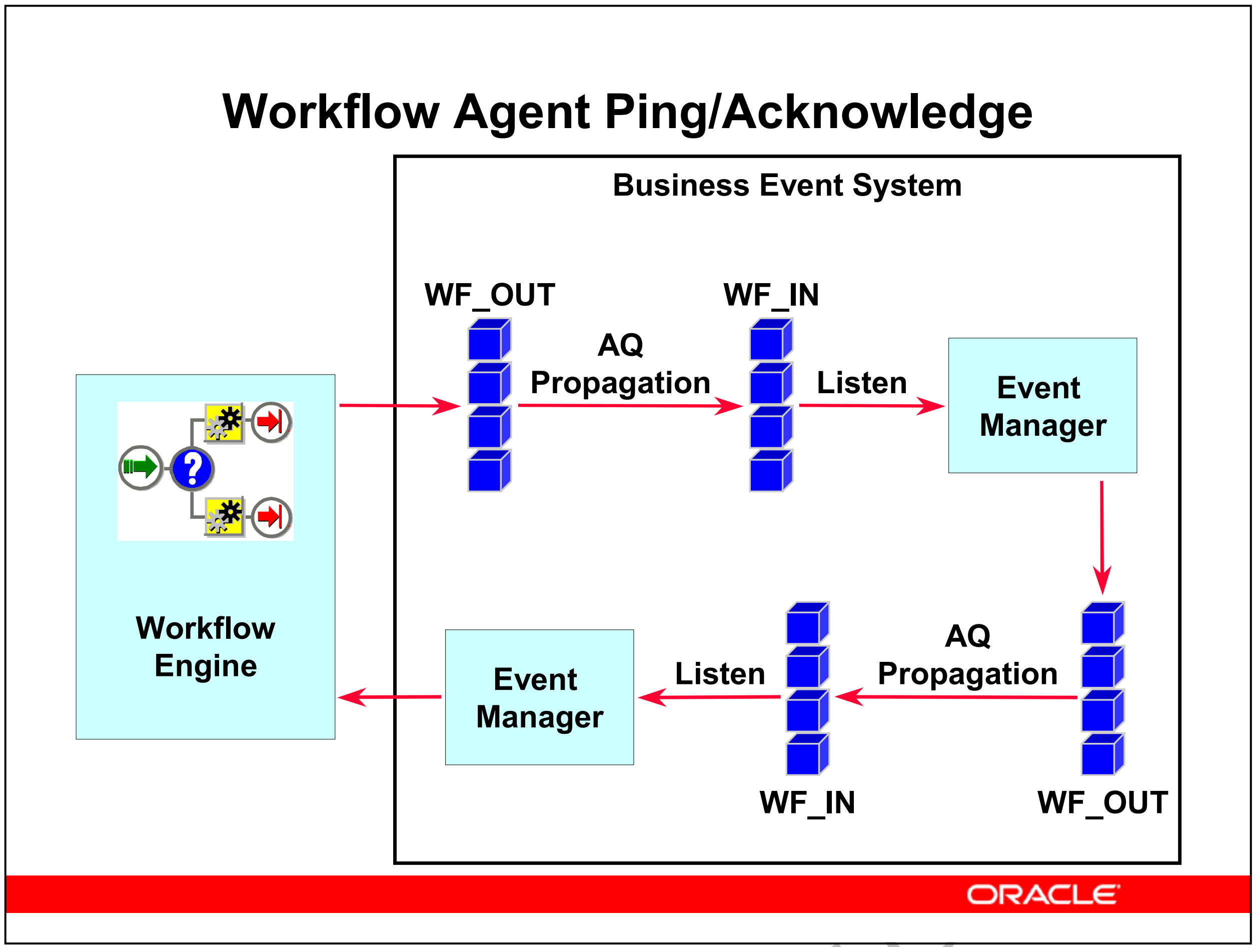
Objectives



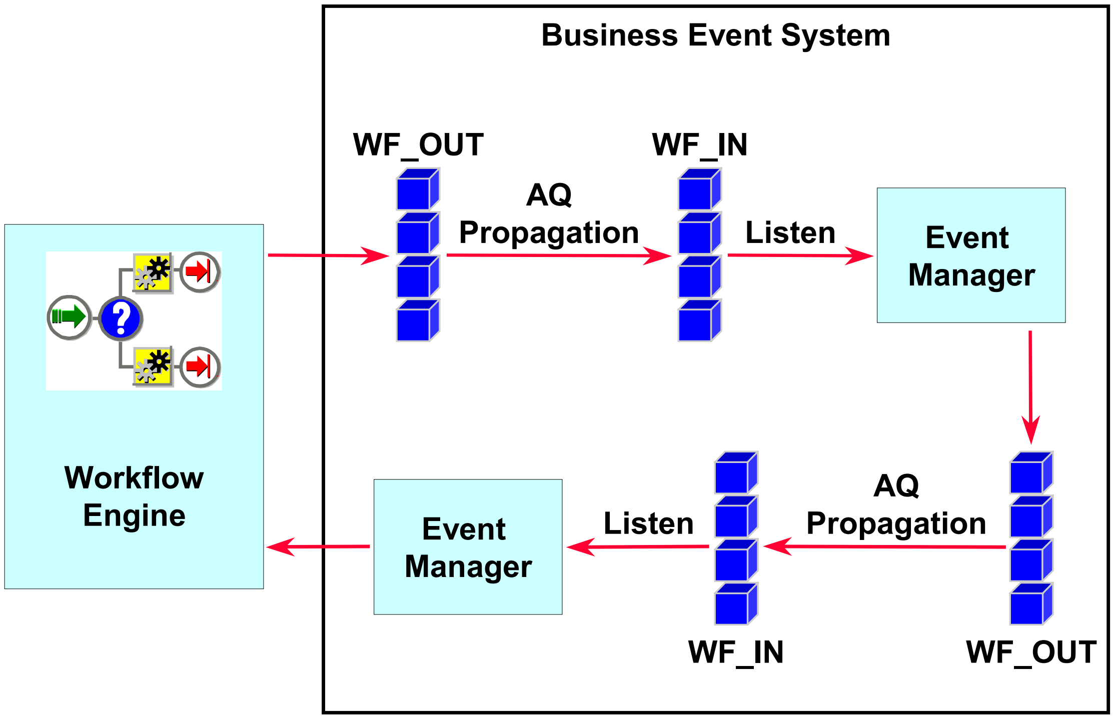
Objectives

After completing this lesson, you should be able to do the following:

- **Ping agents using Workflow Agent Ping/Acknowledge.**
- **Send events through a workflow process using the Workflow Send Protocol.**

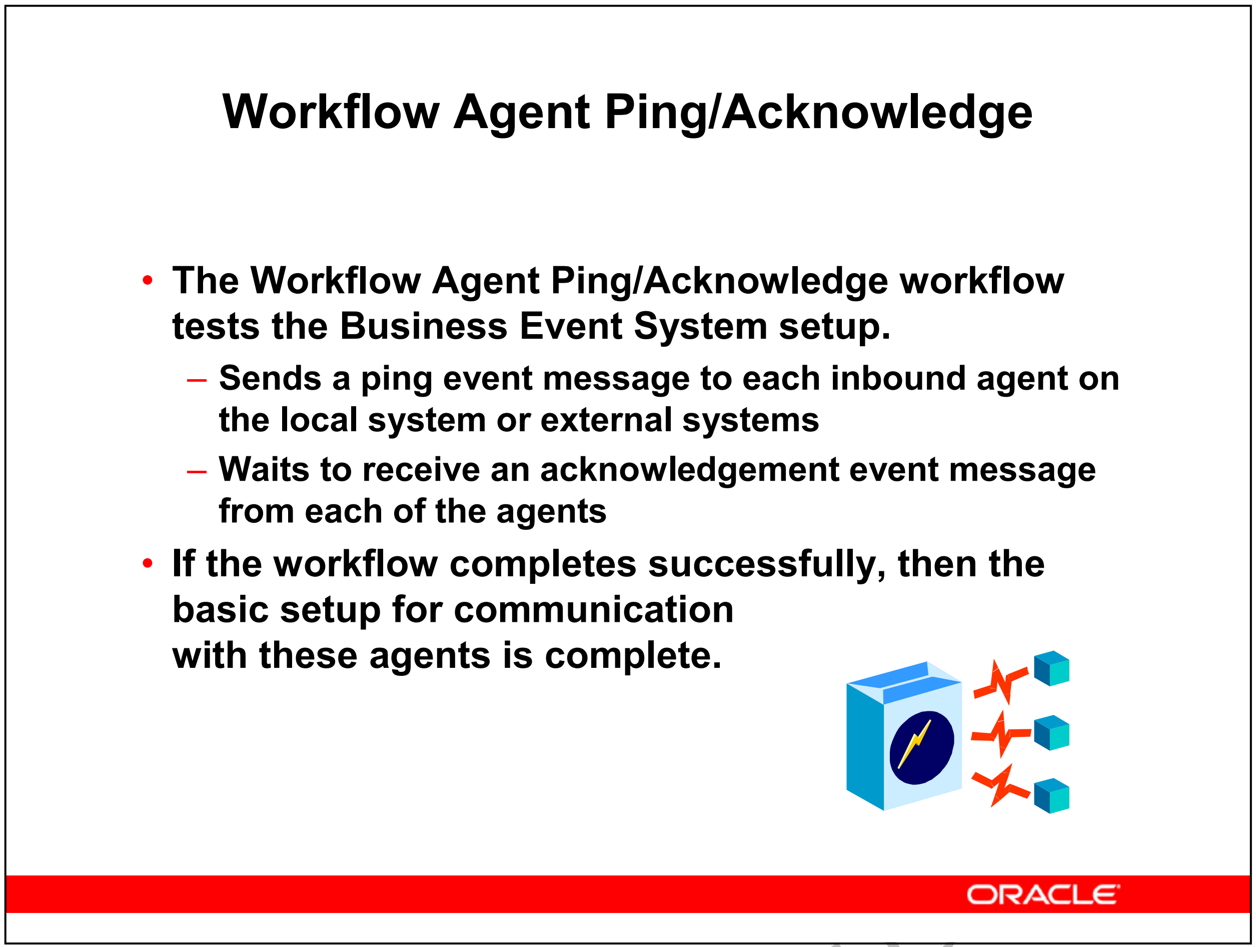


Workflow Agent Ping/Acknowledge



Workflow Agent Ping/Acknowledge

The Workflow Agent Ping/Acknowledge workflow process sends a ping event to each inbound agent on the local system or external systems. When the inbound agent receives the ping event, the Event Manager on that system sends an acknowledgement event back to the workflow process.



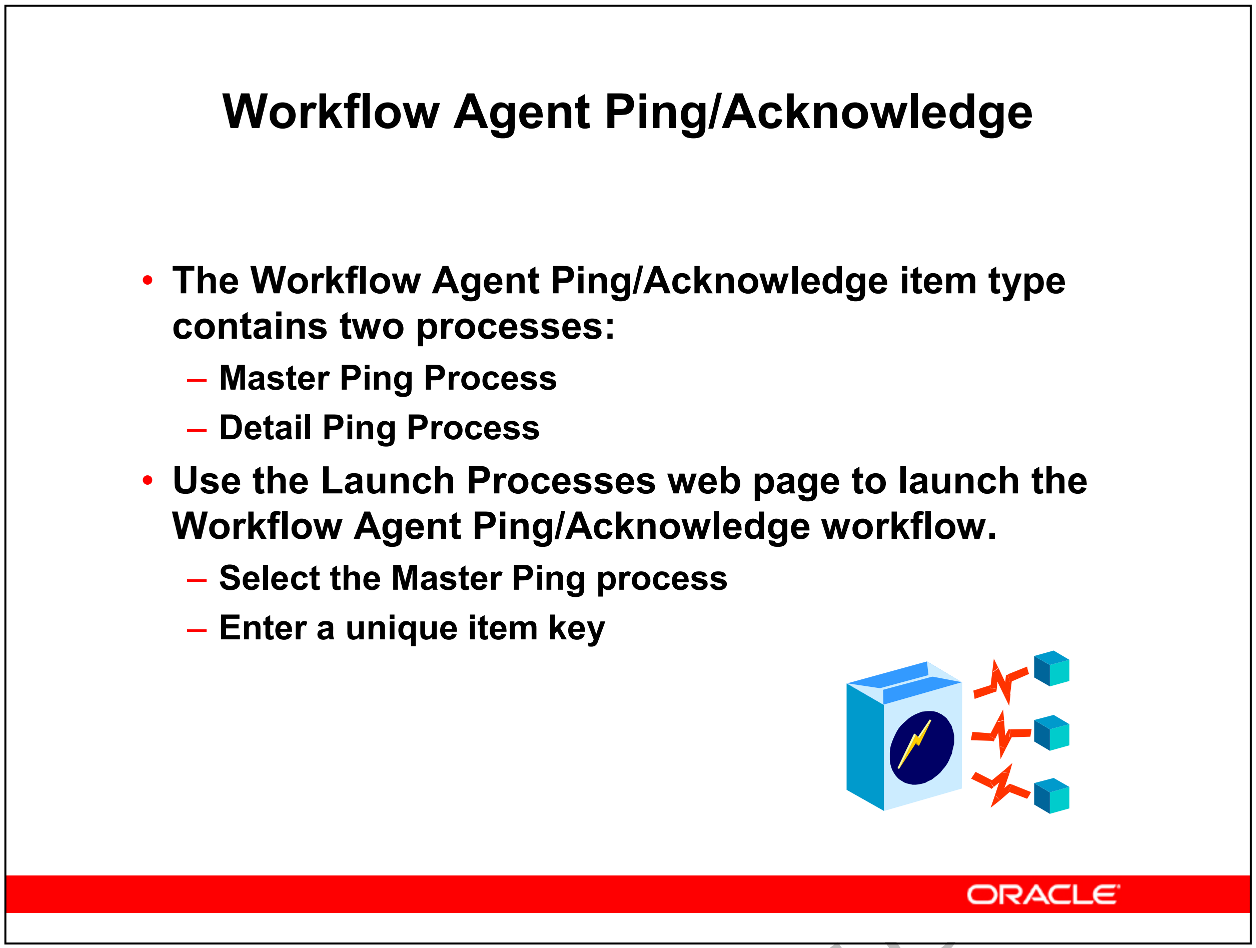
Workflow Agent Ping/Acknowledge

- The Workflow Agent Ping/Acknowledge workflow tests the Business Event System setup.
  - Sends a ping event message to each inbound agent on the local system or external systems
  - Waits to receive an acknowledgement event message from each of the agents
- If the workflow completes successfully, then the basic setup for communication with these agents is complete.

ORACLE

Workflow Agent Ping/Acknowledge

The Workflow Agent Ping/Acknowledge workflow uses the predefined Ping Agent and Acknowledge Ping events, together with the predefined subscriptions to these events, to perform the ping processing. For more information, refer to the Managing Business Events chapter and the Predefined Workflow Events chapter in the *Oracle Workflow Guide*.

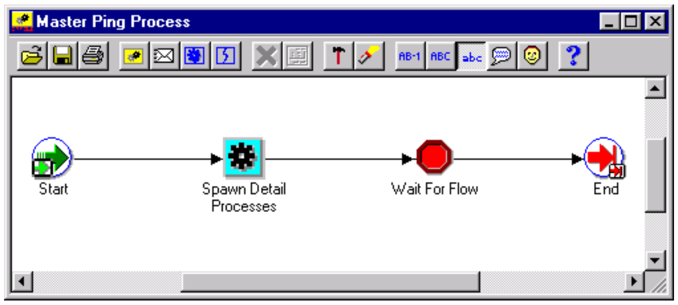


Workflow Agent Ping/Acknowledge

- The Workflow Agent Ping/Acknowledge item type contains two processes:
  – Master Ping Process
  – Detail Ping Process
- Use the Launch Processes web page to launch the Workflow Agent Ping/Acknowledge workflow.
  – Select the Master Ping process
  – Enter a unique item key

ORACLE

Master Ping Process



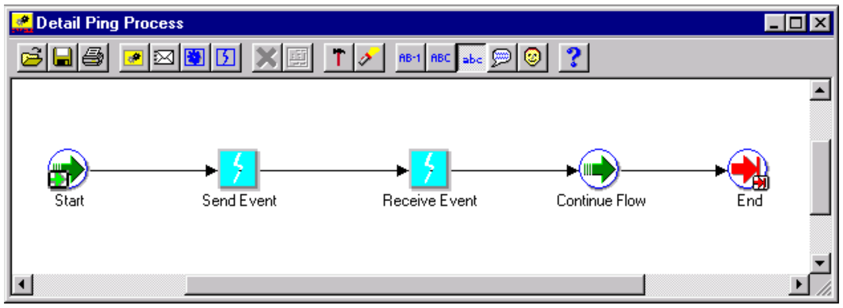
ORACLE

Master Ping Process

- The Workflow Agent Ping/Acknowledge workflow begins when you launch the Master Ping Process using the Launch Processes web page.
- The workflow begins with the Start activity.
- The master process spawns a detail process for each inbound agent that you have defined on the local system or on external systems.
- Each detail process pings an agent by sending it a Ping Agent event and waits to receive an acknowledgement in the form of an Acknowledge Ping event.
- In the master process, a Wait for Flow activity waits for all the detail processes to complete.
- When all the detail processes have completed, the master process ends.

For more information, refer to the Managing Business Events chapter in the *Oracle Workflow Guide*.

Detail Ping Process

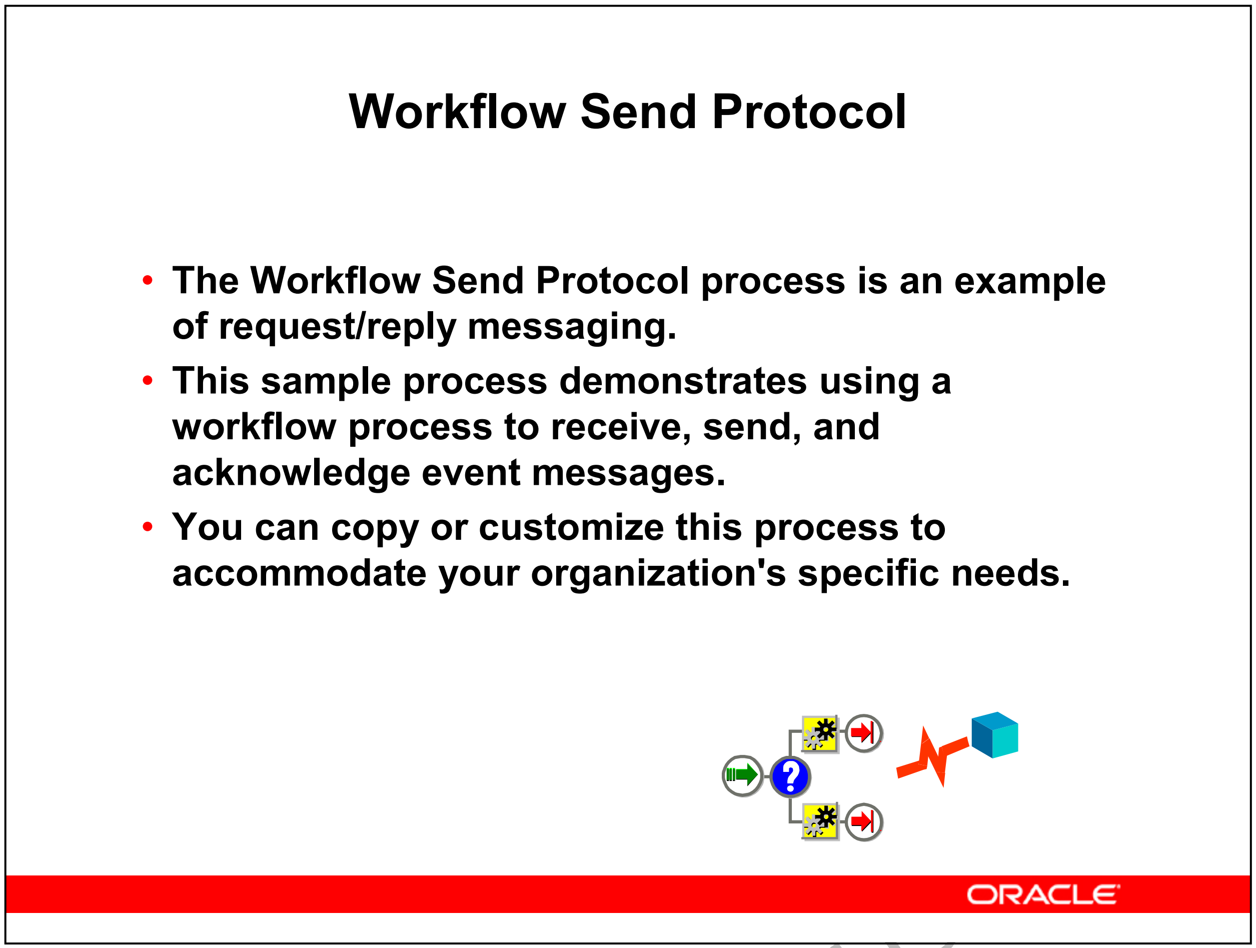


ORACLE

Detail Ping Process

- The Detail Ping process begins when it is launched by the Master Ping process.
- The process begins with the Start activity.
- Next, the process sends a Ping Agent event to the selected inbound agent.
- The process waits to receive an Acknowledge Ping event back from the agent. When the acknowledgement is received, the master process can continue.
- The detail process ends at this point.

For more information, refer to the Managing Business Events chapter in the *Oracle Workflow Guide*.



Workflow Send Protocol

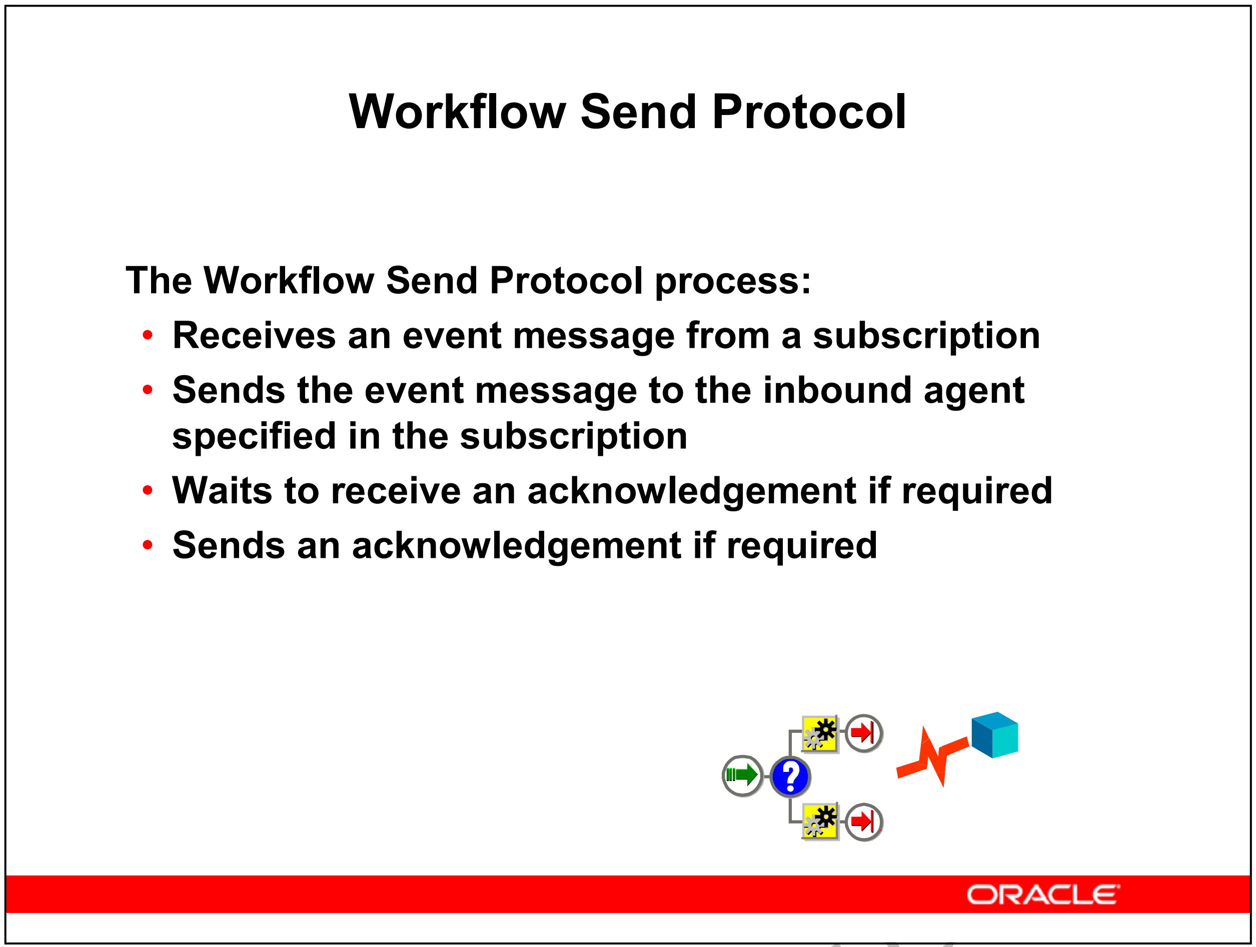
- The Workflow Send Protocol process is an example of request/reply messaging.
- This sample process demonstrates using a workflow process to receive, send, and acknowledge event messages.
- You can copy or customize this process to accommodate your organization's specific needs.

ORACLE

Workflow Send Protocol

The Workflow Send Protocol workflow uses the predefined Workflow Send Protocol and Workflow Send Protocol Acknowledgement events, together with the predefined subscriptions to these events, to perform the send processing. To use the Workflow Send Protocol to send an event to an agent, you must update the appropriate Workflow Send Protocol event subscription with the To Agent that you want to receive the message. You can also optionally specify the Out Agent that you want to send the message.

For more information, refer to the Predefined Workflow Events chapter in the *Oracle Workflow Guide*.

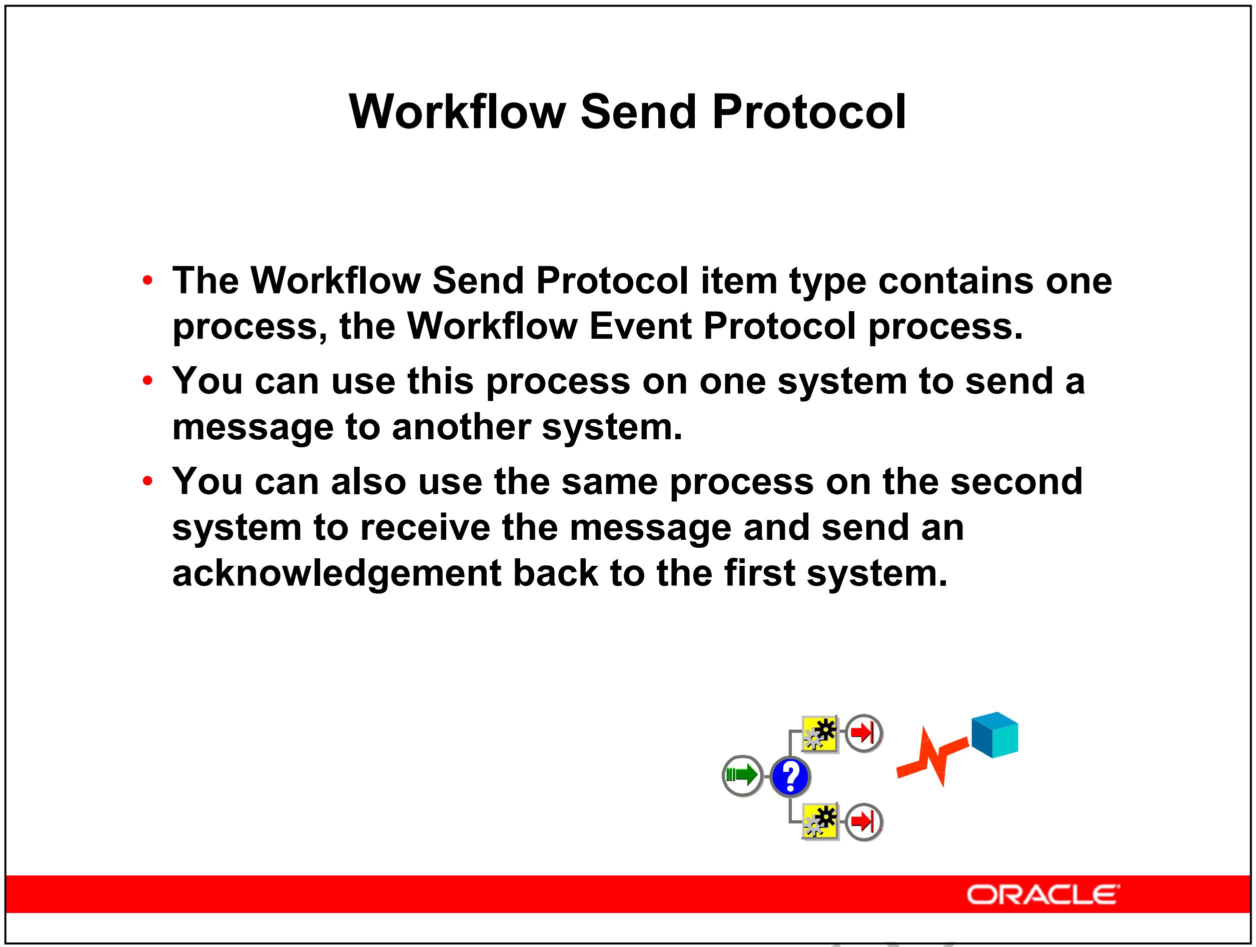


Workflow Send Protocol

The Workflow Send Protocol process:

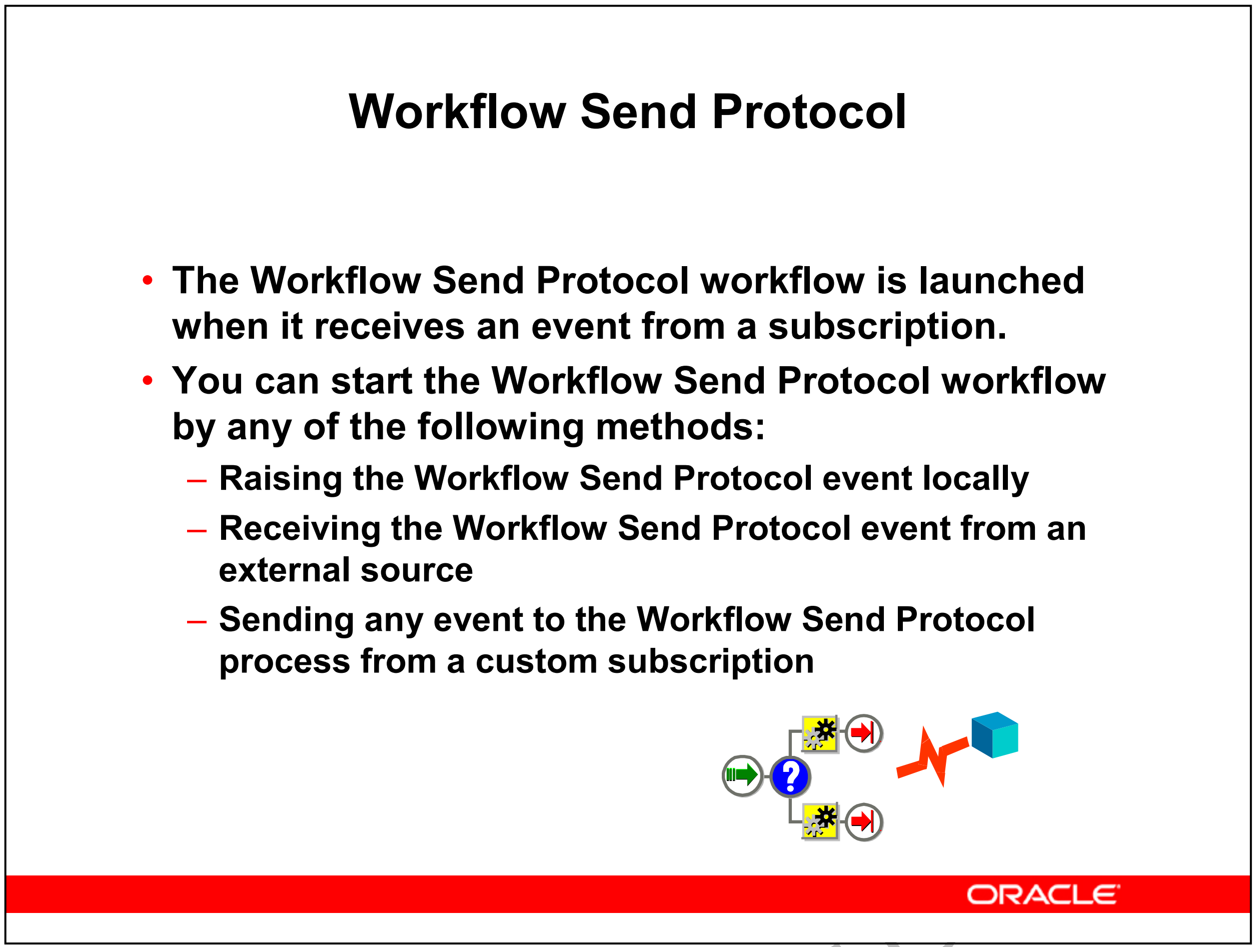
- Receives an event message from a subscription
- Sends the event message to the inbound agent specified in the subscription
- Waits to receive an acknowledgement if required
- Sends an acknowledgement if required

ORACLE



Workflow Send Protocol

- The Workflow Send Protocol item type contains one process, the Workflow Event Protocol process.
- You can use this process on one system to send a message to another system.
- You can also use the same process on the second system to receive the message and send an acknowledgement back to the first system.



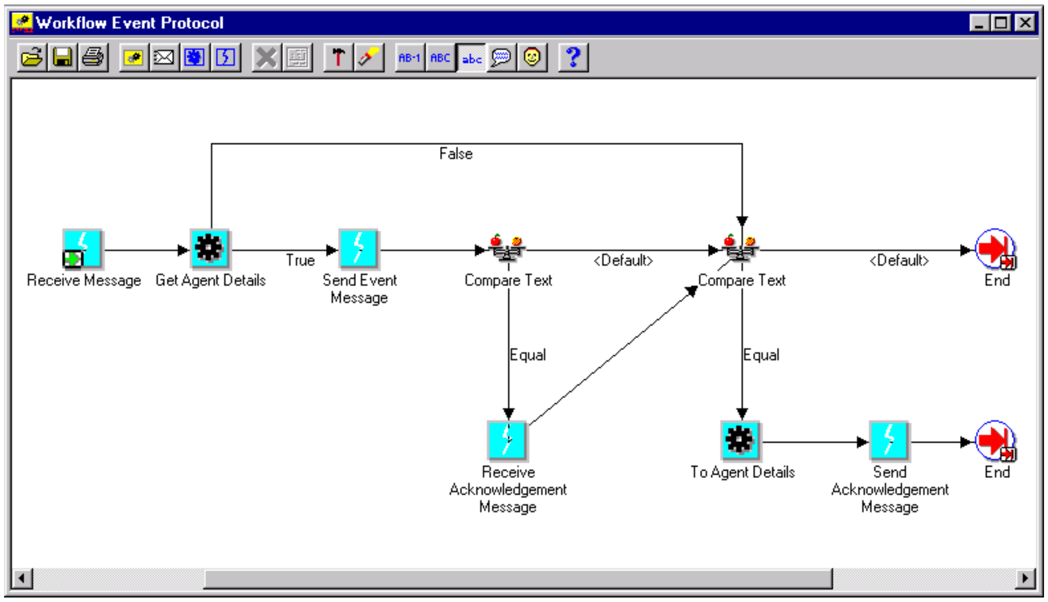
Workflow Send Protocol

- The Workflow Send Protocol workflow is launched when it receives an event from a subscription.
- You can start the Workflow Send Protocol workflow by any of the following methods:
  - Raising the Workflow Send Protocol event locally
  - Receiving the Workflow Send Protocol event from an external source
  - Sending any event to the Workflow Send Protocol process from a custom subscription

Workflow Send Protocol

- You can raise the Workflow Send Protocol event from the Raise event page. Enter a unique event key, and enter any valid XML document as the event data. A predefined subscription sends the event message to the Workflow Event Protocol process.
- The process can also be started when an agent receives the Workflow Send Protocol event from an external source. A predefined subscription sends the event message to the Workflow Event Protocol process.
- Additionally, you can define your own subscription to send any event you choose to the Workflow Event Protocol process. In the subscription, specify the workflow item type as WFSNDPRT and the workflow process name as WFEVPRTC. Ensure that you either use the default rule function or include send processing in your custom rule function to send the event to the workflow. The Workflow Event Protocol process starts when the subscription is executed and the process receives the event message.

Workflow Event Protocol Process

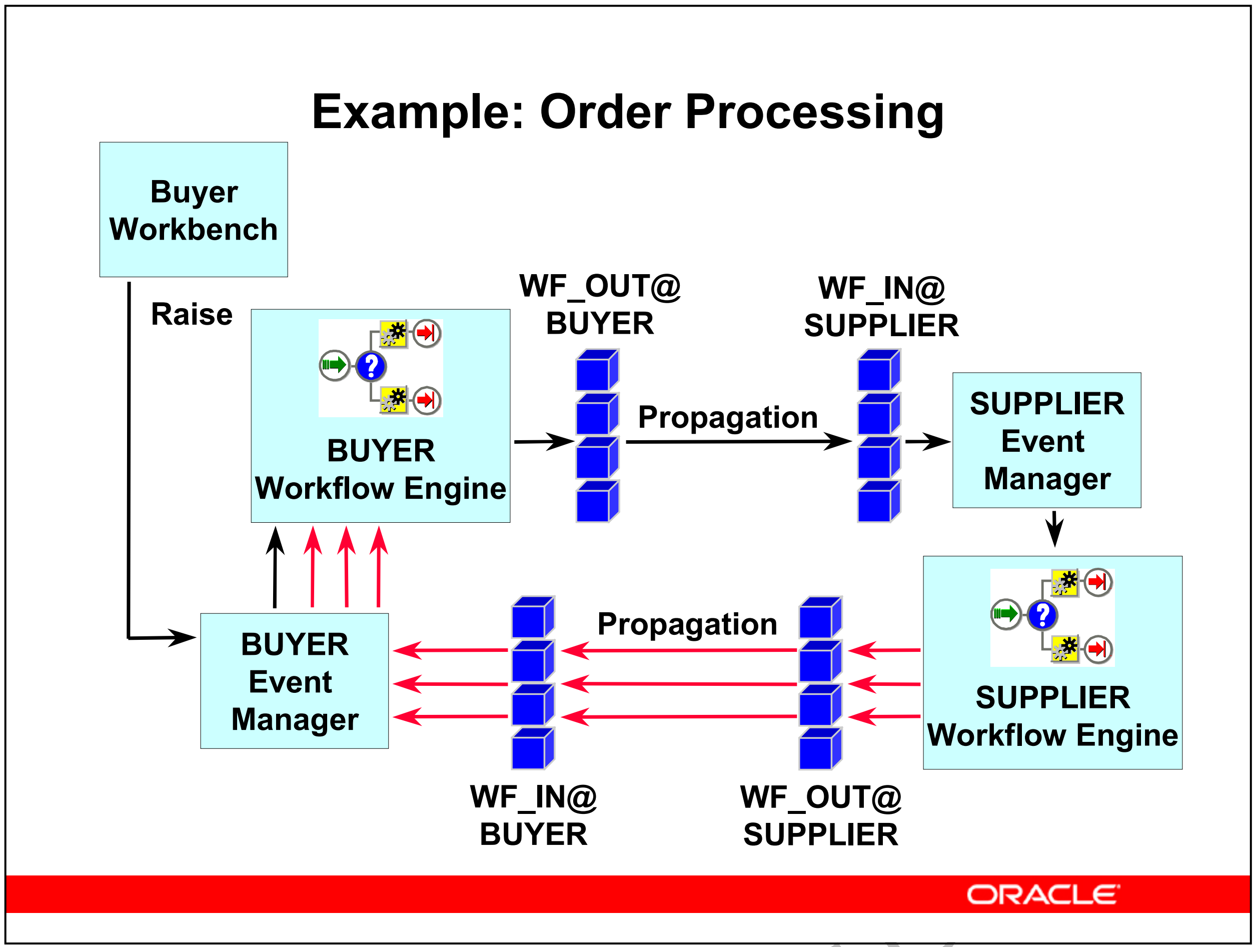


Workflow Event Protocol Process

- The Workflow Send Protocol workflow begins when the Event Manager sends an event message to the Workflow Event Protocol process.
- The workflow begins with the Receive Message activity.
- Next, the process attempts to retrieve the agent details for the intended outbound and inbound agents from the subscription.
- If no inbound agent is specified, the process continues immediately to the second Compare Text node to determine whether to send an acknowledgement message.
- If the subscription does specify a To Agent, the process sends the event message to that agent.
- Then the process continues to the first Compare Text node to determine whether the event message requires an acknowledgement from the recipient, based on a subscription parameter.
- If an acknowledgement is required, the Workflow Engine waits to receive the acknowledgement message.

- Otherwise, the process continues immediately to the second Compare Text node to determine whether to send an acknowledgement message.
- At the second Compare Text node, the process determines whether it should send an acknowledgement of the original message that it received. If no acknowledgement needs to be sent, the process ends at this point.
- Otherwise, the process retrieves the agent details for the inbound agent where the acknowledgement must be sent and sends the acknowledgement message to that agent. Then the process ends.

For more information, refer to the Predefined Workflow Events chapter in the *Oracle Workflow Guide*.

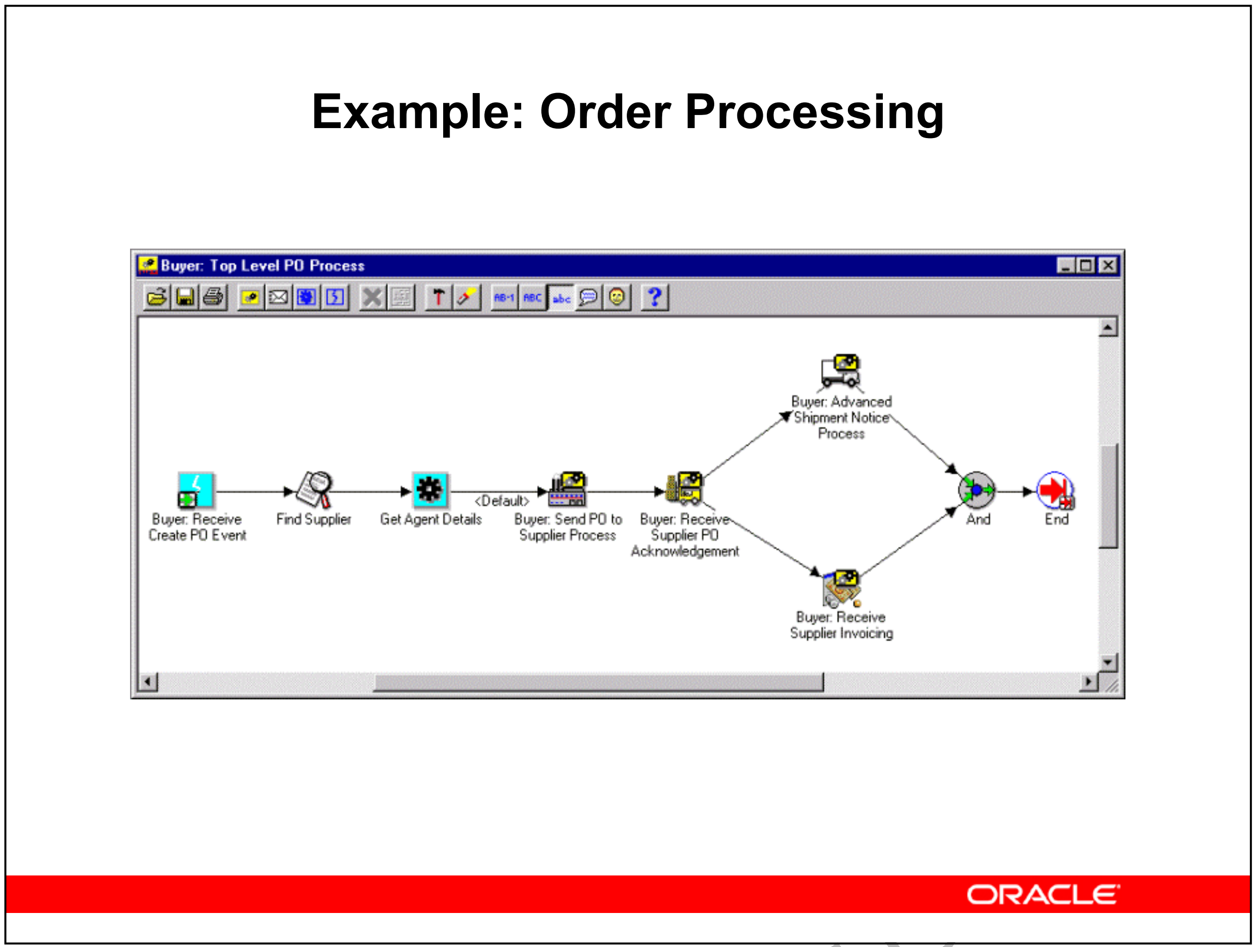


Example: Order Processing

Example: Order Processing

This example shows how you can use events to build workflow processes that transmit business documents between two systems. The example comes from the Event System Demonstration workflow process which is available with the standalone version of Oracle Workflow. The business flow for a purchase order is controlled by workflow processes on two different systems, a Buyer system and a Supplier system, which use events to communicate with each other.

The order processing is initiated when a purchase order is entered from a Buyer Workbench on a Buyer system. Oracle Workflow generates a purchase order XML document and sends this document to a Supplier system in an event message. The Supplier system processes the purchase order and sends back to the Buyer system three event messages with XML documents representing a purchase order acknowledgement, an advanced shipment notice, and an invoice.



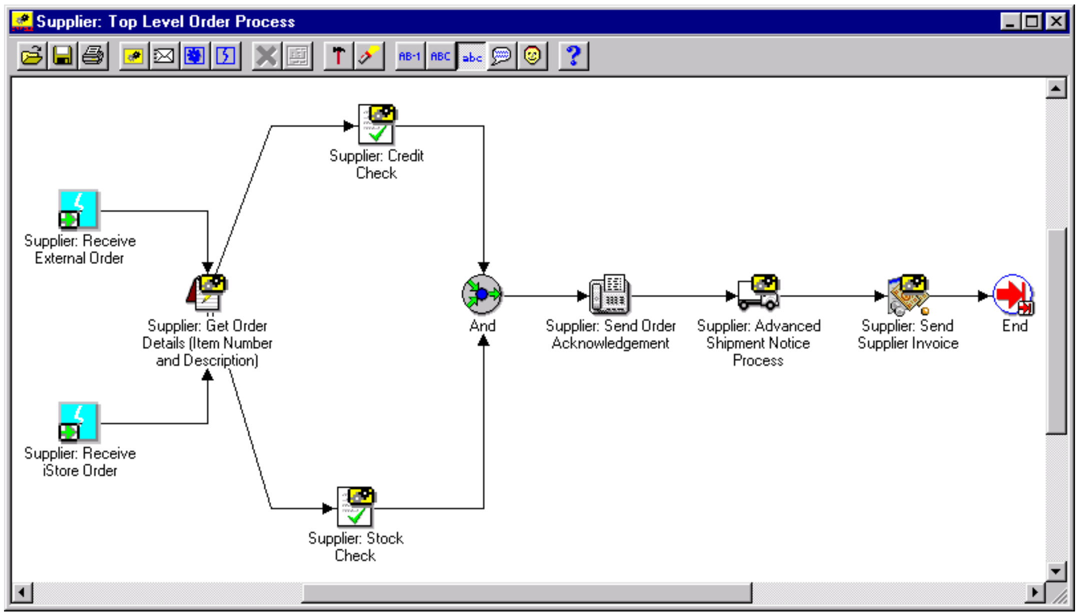
Example: Order Processing

Buyer: Top Level PO Process
Buyer: Receive Create PO Event
Find Supplier
Get Agent Details
<Default>
Buyer: Send PO to Supplier Process
Buyer: Receive Supplier PO Acknowledgement
Buyer: Advanced Shipment Notice Process
Buyer: Receive Supplier Invoicing
And
End
ORACLE

Example: Order Processing

This example shows the Buyer: Top Level PO Process from the Event System Demonstration workflow.

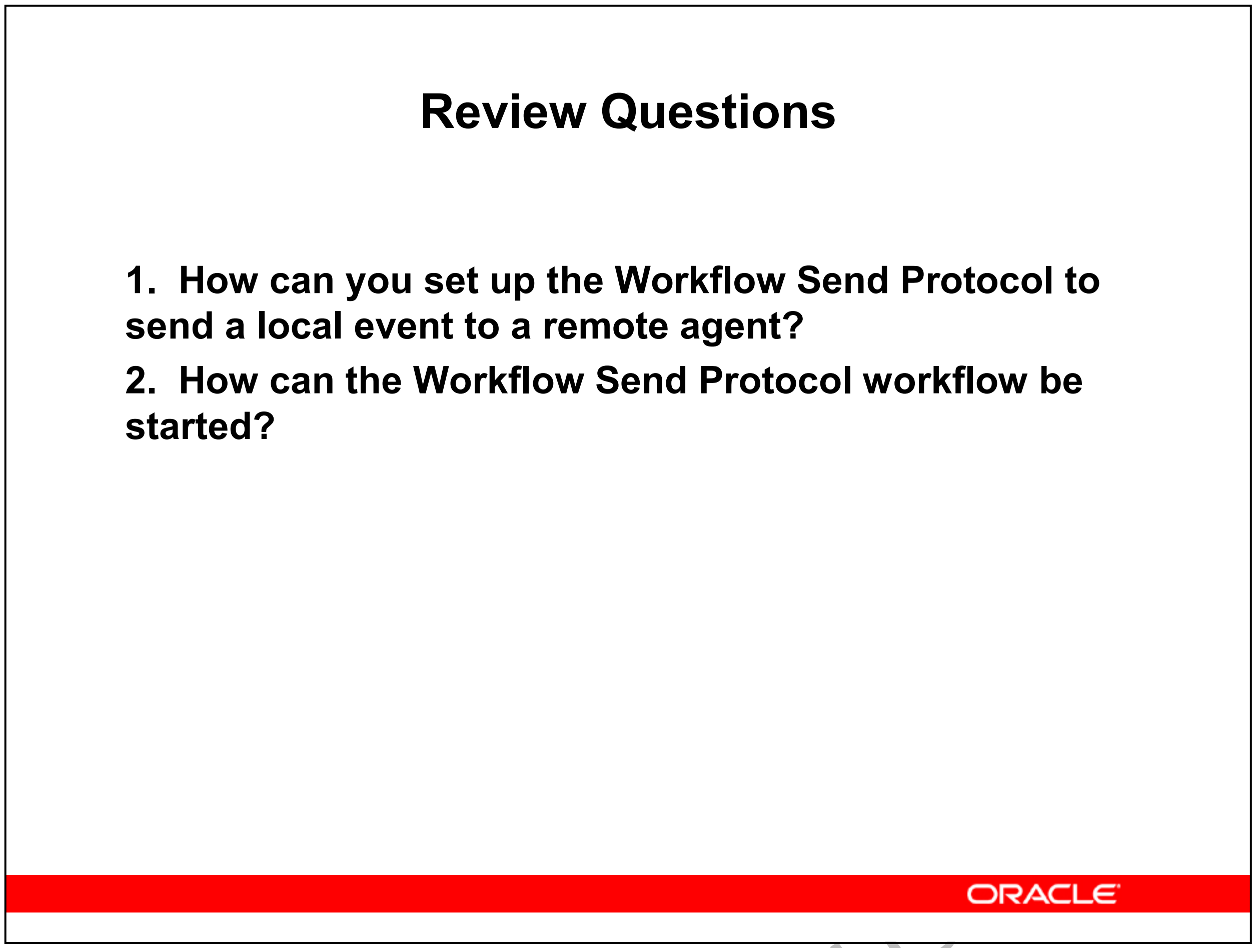
Example: Order Processing


Supplier: Top Level Order Process
Supplier: Credit Check
Supplier: Receive External Order
Supplier: Get Order Details (Item Number and Description)
And
Supplier: Send Order Acknowledgement
Supplier: Advanced Shipment Notice Process
Supplier: Send Supplier Invoice
End
Supplier: Receive iStore Order
Supplier: Stock Check

Example: Order Processing

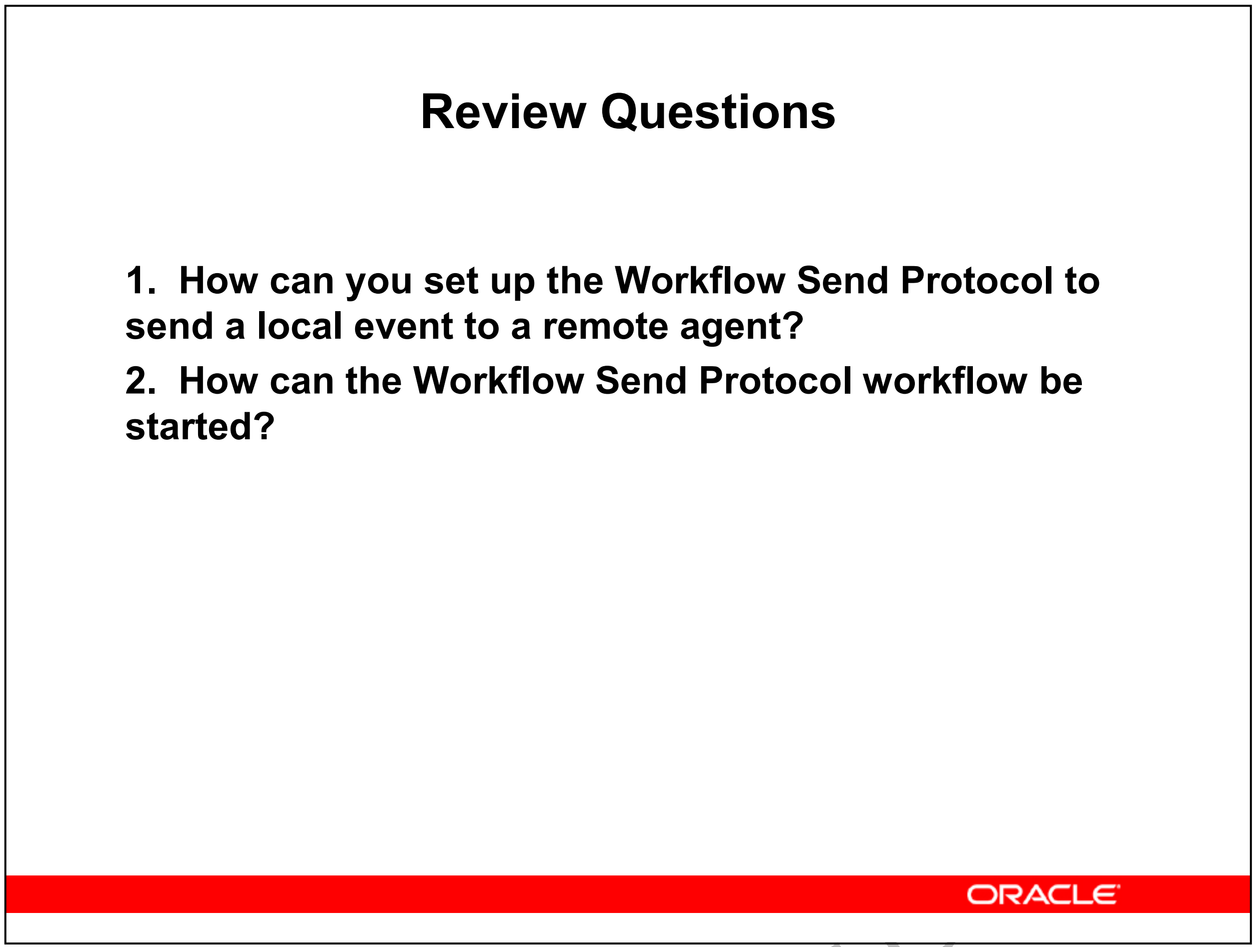
This example shows the Supplier: Top Level Order Process from the Event System Demonstration workflow.

Review Questions



Review Questions

1. How can you set up the Workflow Send Protocol to send a local event to a remote agent?
2. How can the Workflow Send Protocol workflow be started?



Review Questions

1. How can you set up the Workflow Send Protocol to send a local event to a remote agent?
2. How can the Workflow Send Protocol workflow be started?

Review Questions and Solutions

1. How can you set up the Workflow Send Protocol to send a local event to a remote agent?

   Enter the agent as the To Agent in the predefined Local subscription to the Workflow Send Protocol event. You can also optionally specify the Out Agent from which you want to send the event.

2. How can the Workflow Send Protocol process be started?

   You can start the Workflow Send Protocol process by any of the following methods:

   - **Raising the Workflow Send Protocol event locally**
   - **Receiving the Workflow Send Protocol event from an external source**
   - **Sending any event to the Workflow Send Protocol process from a custom subscription**

Practice - Pinging Agents

Overview

In this practice, you will launch the Workflow Agent Ping/Acknowledge process to ping all inbound agents defined in the Event Manager. After completing the practice, you can review the process in the Workflow Monitor to confirm that the appropriate acknowledgements were received and the process completed successfully.

Note: Because many students access the system and create objects during this course, you need a way to distinguish between the objects created by you and by your classmates. Therefore, you will be assigned a terminal number by your instructor. Use this number as a prefix wherever you see *XX* included in the name of something you are defining. In this way, you can ensure that the definitions you create are unique.

Assumptions

- The instructor will provide you with the username and password of a user with workflow administrator privileges. The workflow administrator is defined in the Global Workflow Preferences page.

- For standalone Oracle Workflow, the instructor will provide you with the URL for the Oracle Workflow home page. The URL is <*webagent*>/wfa_html.home, where <*webagent*> is the base URL of the web agent configured for Oracle Workflow in your Web server.

- For Oracle Workflow embedded in Oracle E-Business Suite, the instructor will provide you with the name of a Workflow administrator responsibility that includes Event Manager functionality. The username you use to log in should have this responsibility assigned to it.

Tasks

1. Launch the Workflow Agent Ping/Acknowledge process to ping all inbound agents defined in the Event Manager.

   Solution:

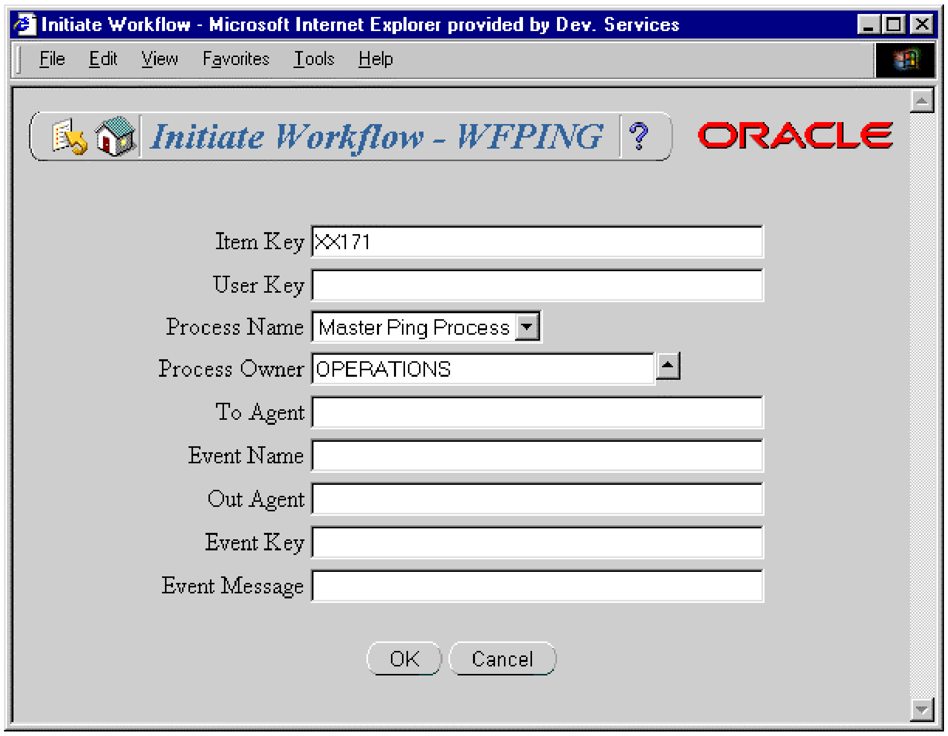
   1. Use a web browser to connect to the Oracle Workflow home page with the URL provided by the instructor for standalone Oracle Workflow, or to a Workflow administrator responsibility provided by the instructor for Oracle Workflow embedded in Oracle E-Business Suite. Log in as a user with workflow administrator privileges.

   2. Choose the Launch Processes link.

   3. In the Launch Processes page, choose the Workflow Agent Ping/Acknowledge item type.

4. In the Initiate Workflow page, enter *XX*171 in the Item Key field.

5. Select Master Ping Process in the Process Name field.


Initiate Workflow - Microsoft Internet Explorer provided by Dev. Services
File Edit View Favorites Tools Help
Initiate Workflow - WFPING
ORACLE
OK
Cancel

Item Key	XX171
User Key	
Process Name	Master Ping Process
Process Owner	OPERATIONS
To Agent	
Event Name	
Out Agent	
Event Key	
Event Message	

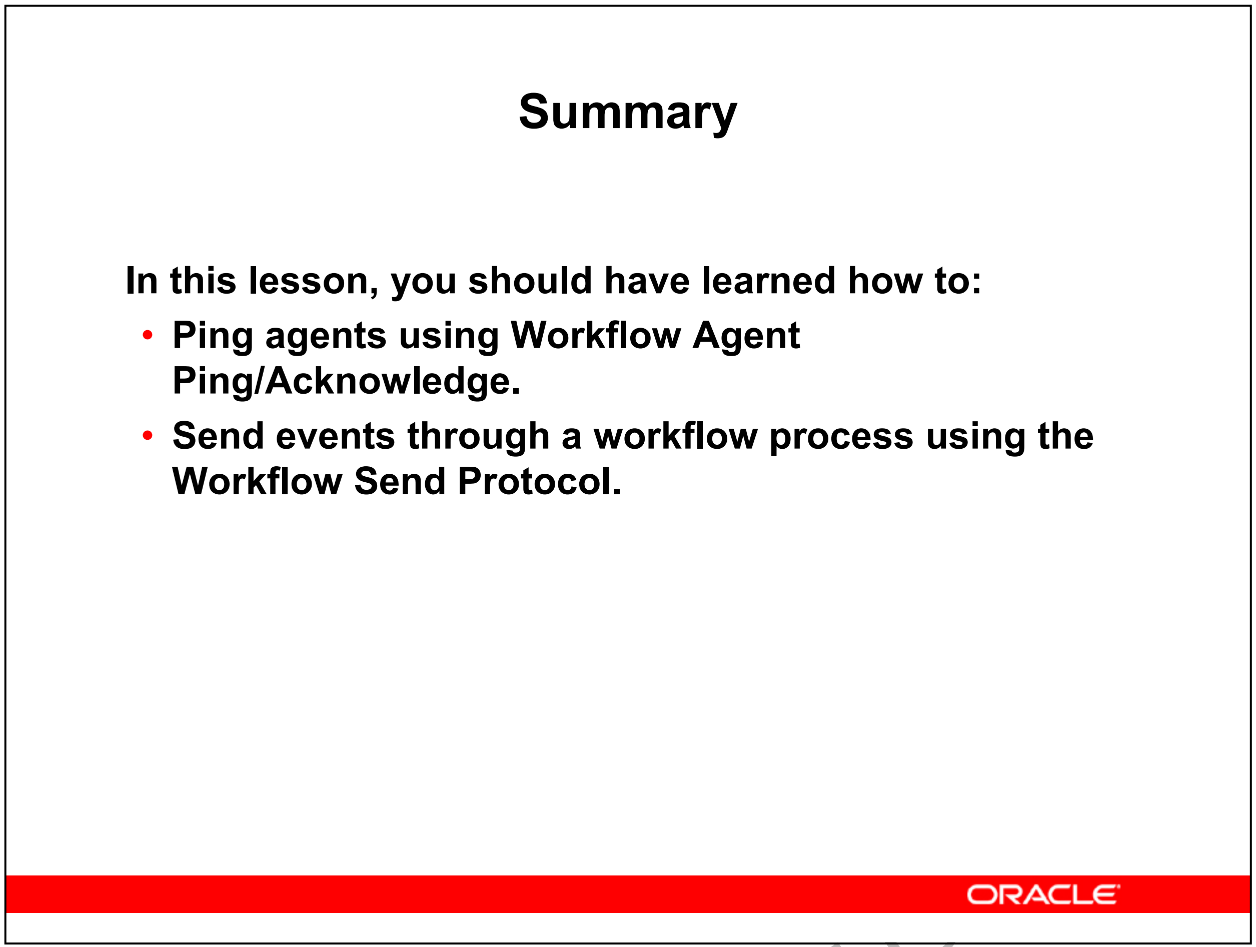
6. Choose OK.

7. Review the process activities in the Activities List page that appears.

8. Choose View Diagram to review the status of the process in the Workflow Monitor. If the process completes successfully, then the basic setup for communication with the inbound agents defined in the Event Manager is complete.

   Note: The amount of time it takes for the process to complete depends on the propagation scheduled for the outbound queue and the listeners being run for the inbound queues.

9. You can also review the Ping Agent and Acknowledge Ping event messages on the local queues by choosing Event Queue Summary from the Workflow home page.



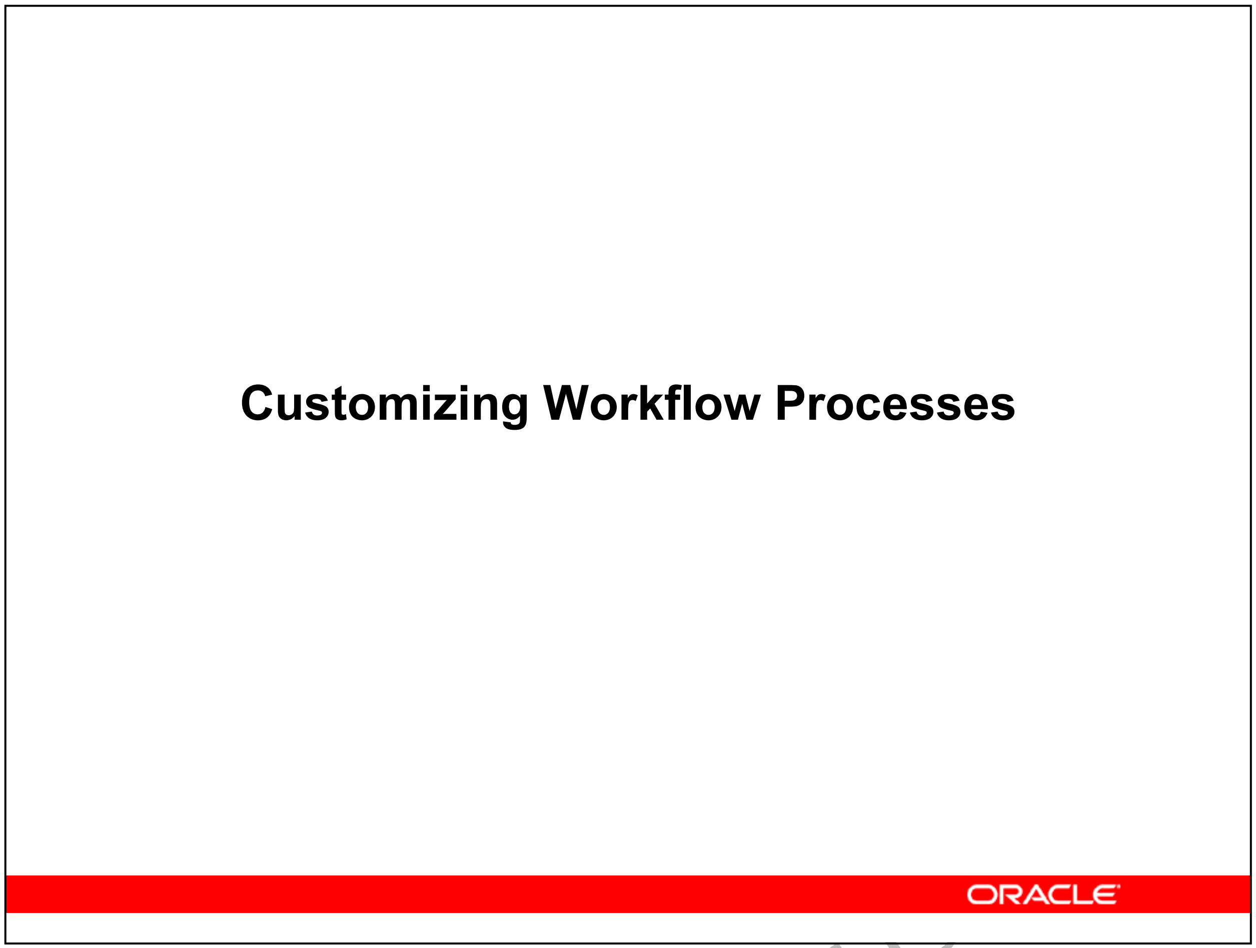
Summary

In this lesson, you should have learned how to:

- Ping agents using Workflow Agent Ping/Acknowledge.
- Send events through a workflow process using the Workflow Send Protocol.

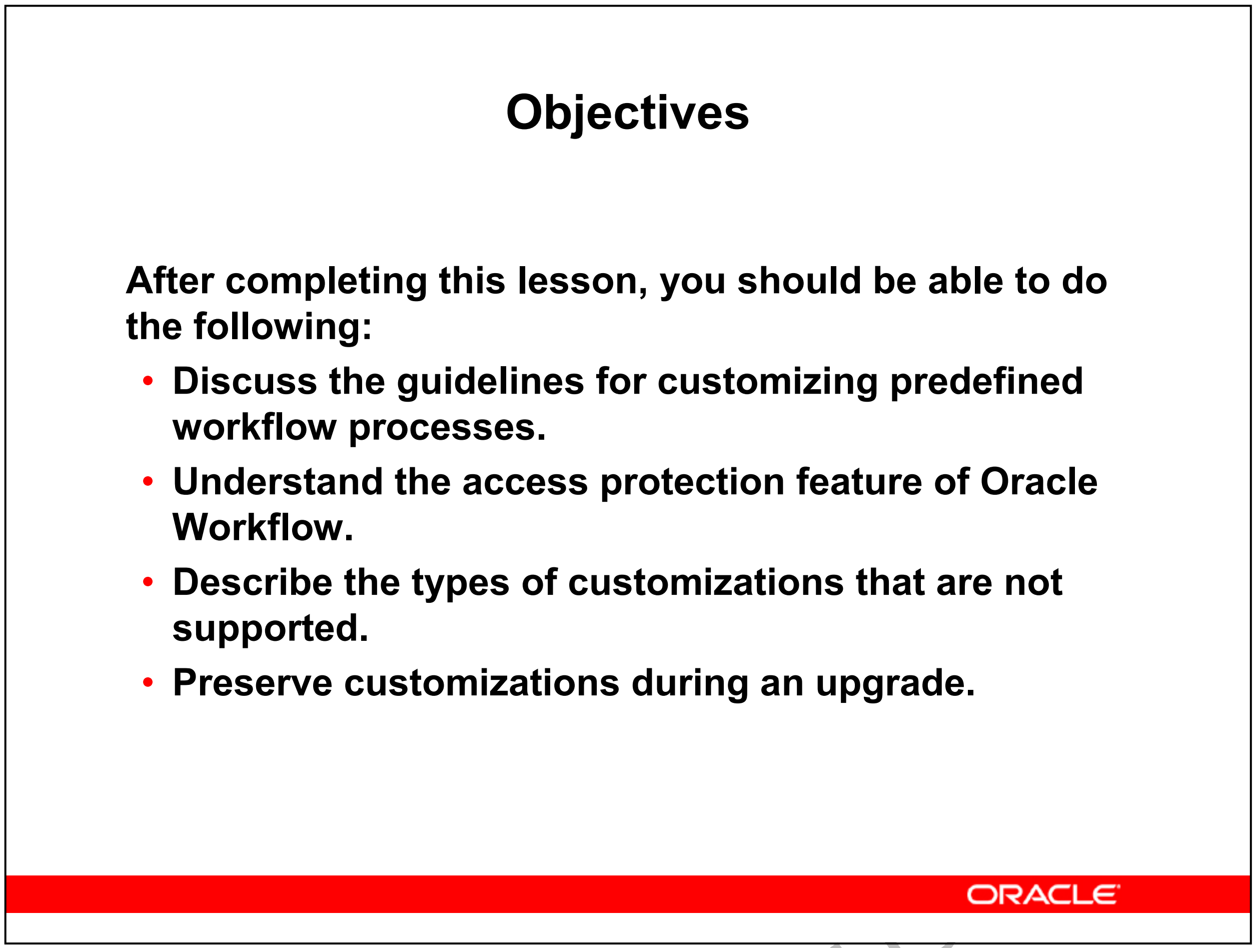
Customizing Workflow Processes

Chapter 18



Customizing Workflow Processes

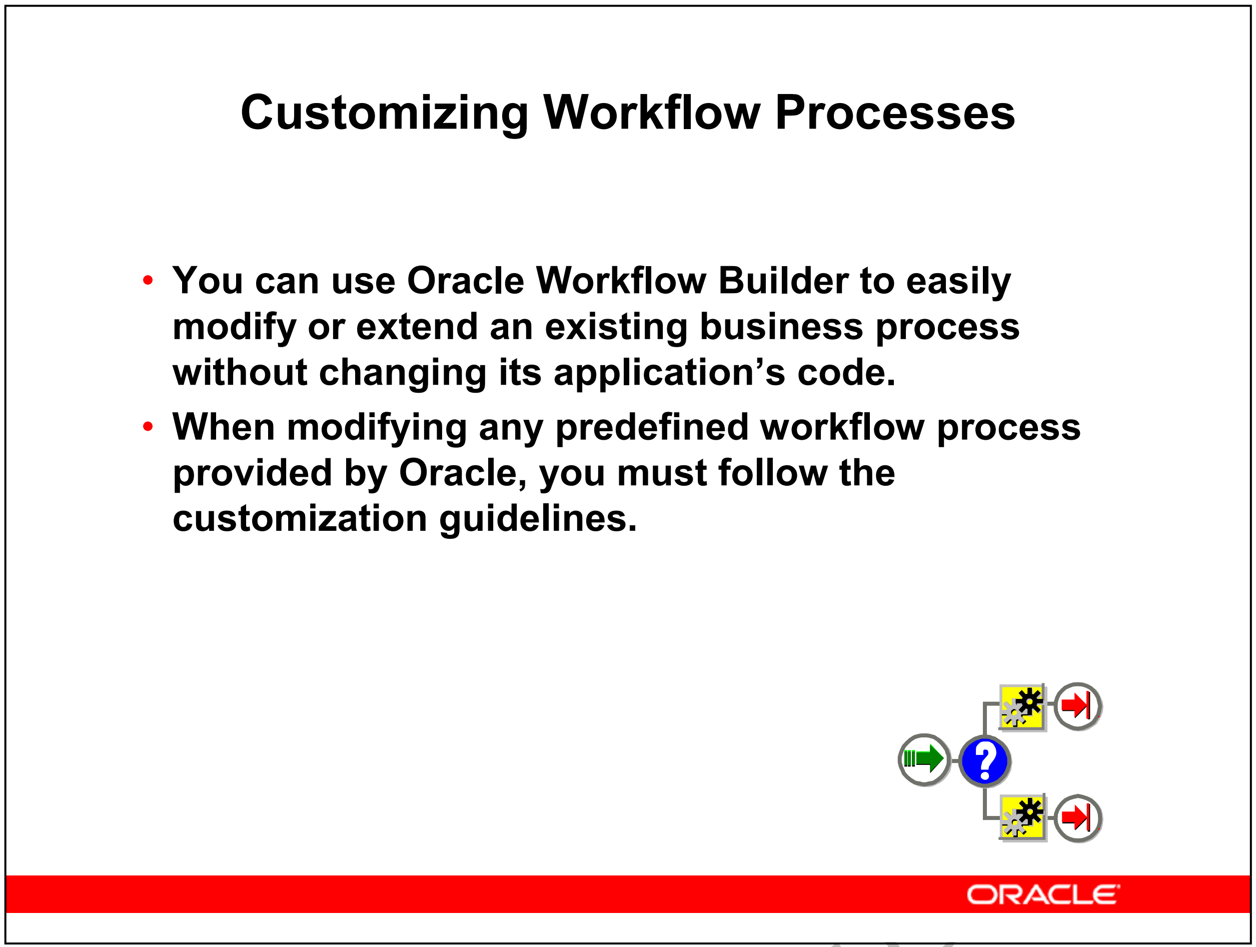
ORACLE



Objectives

After completing this lesson, you should be able to do the following:

- Discuss the guidelines for customizing predefined workflow processes.
- Understand the access protection feature of Oracle Workflow.
- Describe the types of customizations that are not supported.
- Preserve customizations during an upgrade.



Customizing Workflow Processes

- **You can use Oracle Workflow Builder to easily modify or extend an existing business process without changing its application's code.**
- **When modifying any predefined workflow process provided by Oracle, you must follow the customization guidelines.**

Customizing Workflow Processes

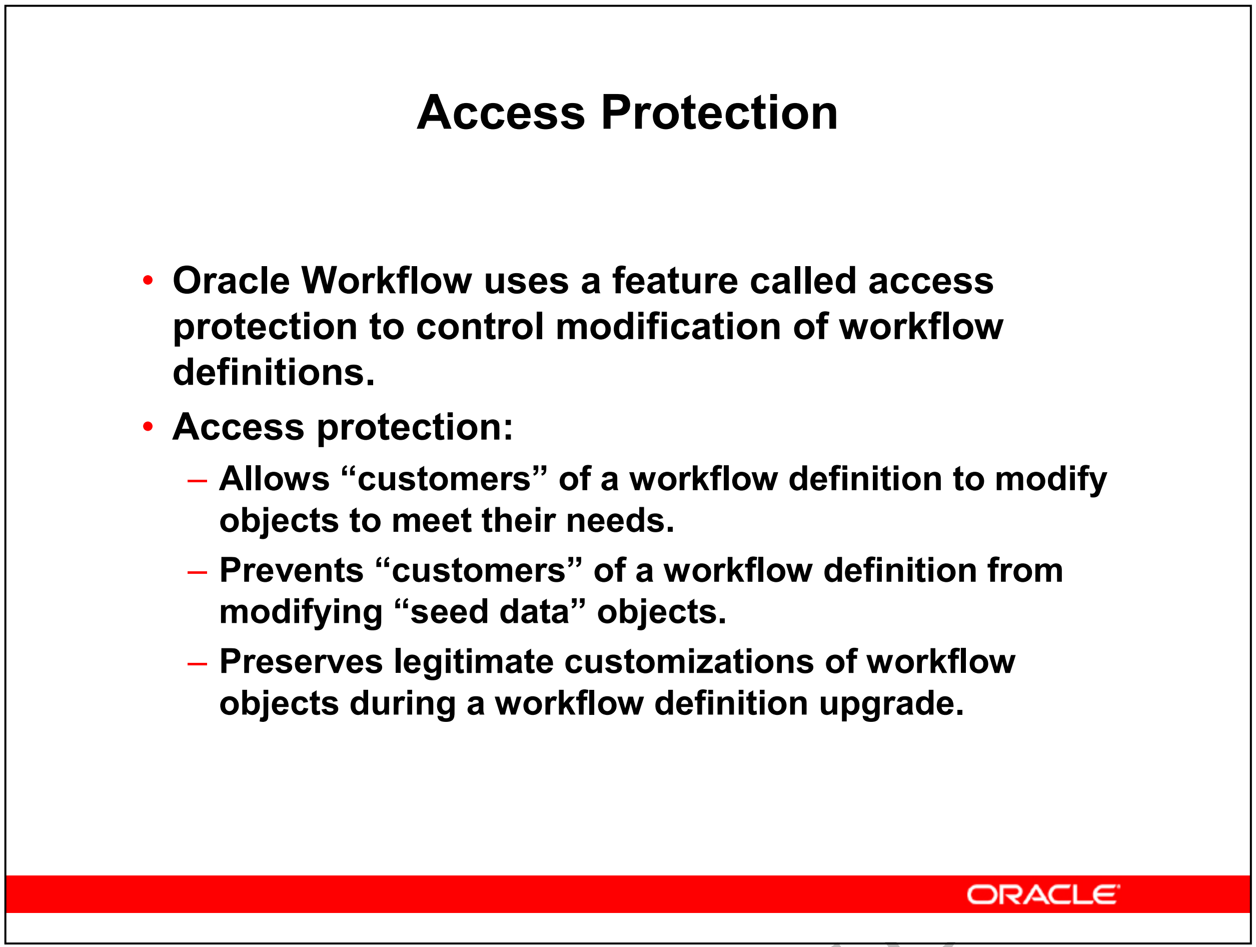
When customizing workflow processes provided by an Oracle product, you must follow these guidelines:

1. Verify that all setup steps have been completed as documented in the Oracle Workflow Guide, and the product-specific user's guides.
2. Test the unmodified seeded workflow, event, or subscription on a test database and ensure that it runs successfully with the setup and data specific to your environment.
3. Refer to the product-specific user's guide and any documentation update for the specific workflow, event, or subscription of interest. These documentation sources specifically mention what should NOT be modified. Oracle Support Services will not support modifications to any object that is specifically documented as not modifiable.
4. Gradually build in customizations step–by–step, and test the customized workflow or subscription after each step.
5. When creating PL/SQL procedures, conform to the standard PL/SQL API templates documented in the Oracle Workflow Guide. Be sure to handle exceptions in the event of an error so you can track down the procedure where the error has occurred.

6. Do not implement the customized workflow, event, or subscription in production without fully ensuring that it works successfully on a test database, which is a replica of your production setup.

The following types of customizations are supported:

1. Any customization that is stated as required in the seeded workflow's product-specific user's guide or documentation update.
2. Customization examples documented in the product-specific user's guide or documentation update. Any issues that arise are fully supported to resolution, to the extent that the customization example was followed as documented. Any deviation from what is documented amounts to a custom development issue that needs further evaluation. See number 3 below.
3. Customizations that are not explicitly stated as unsupported customizations, as required customizations, or as supported customization examples are supported to the extent that the customer must first isolate the problem following the customization guidelines listed above. If upon evaluation, Oracle Support Services deems that the isolated problem stems from an Oracle product, Oracle will supply a solution. Otherwise, it is the responsibility of the customer to correct the custom development issue.



Access Protection

- Oracle Workflow uses a feature called access protection to control modification of workflow definitions.
- Access protection:
  - Allows “customers” of a workflow definition to modify objects to meet their needs.
  - Prevents “customers” of a workflow definition from modifying “seed data” objects.
  - Preserves legitimate customizations of workflow objects during a workflow definition upgrade.

Access Protection

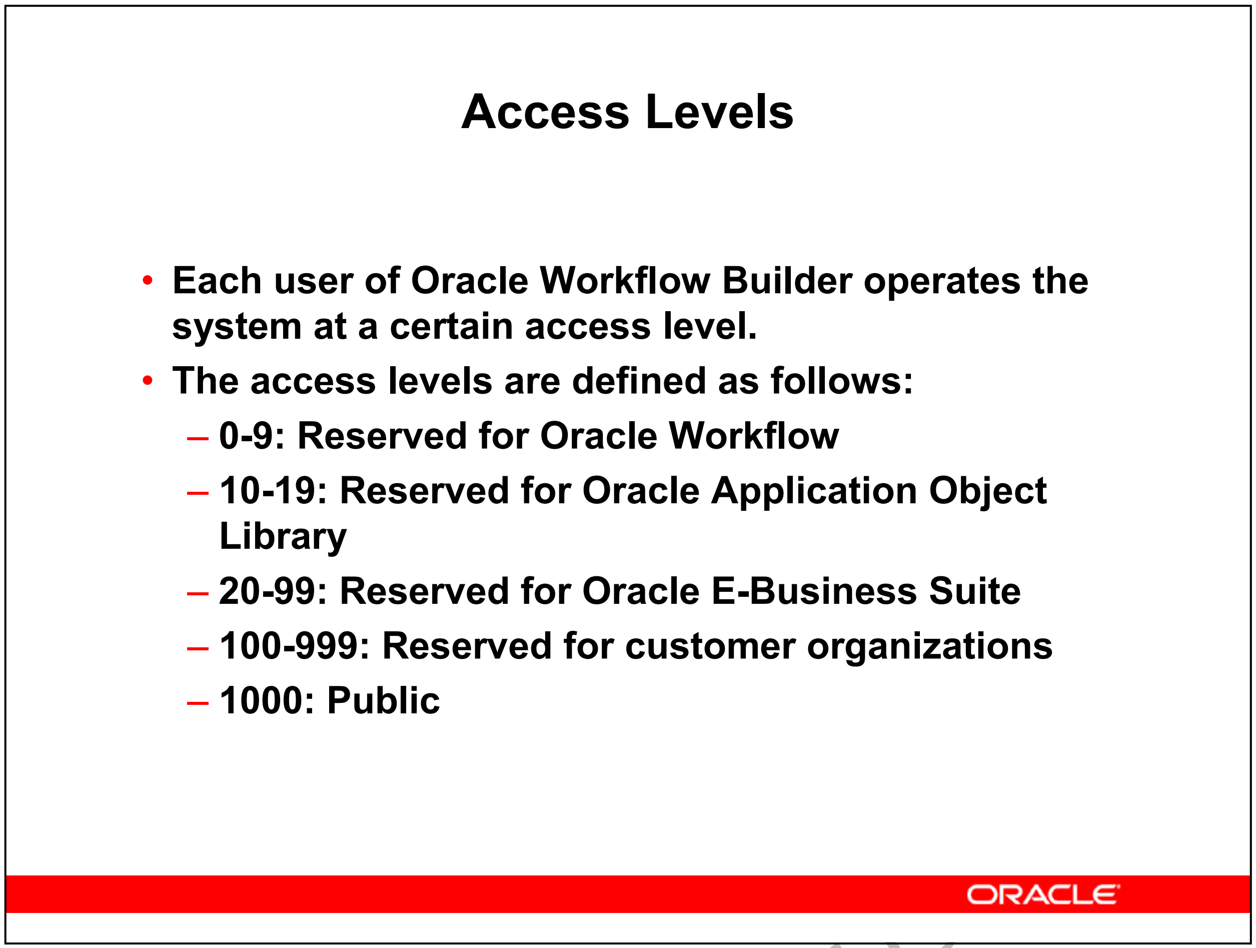
As a workflow developer, you can use access protection to allow or discourage “customers” of your workflows from modifying your “seed data” workflow definitions.

As a customer of predefined workflows provided by Oracle, you can use access protection to preserve valid customizations you have made to a predefined workflow during a workflow definition upgrade.

All workflow objects except lookup codes, function attributes, and message attributes contain an Access tab in their property pages. Lookup codes, function attributes, and message attributes inherit their access settings from their parent lookup type, function, or message, respectively.

The Access tab lets you define whether:

- Future customizations to the object are preserved during a workflow definition upgrade.
- The object can be edited by users operating at a higher access level.



Access Levels

- Each user of Oracle Workflow Builder operates the system at a certain access level.
- The access levels are defined as follows:
  - 0-9: Reserved for Oracle Workflow
  - 10-19: Reserved for Oracle Application Object Library
  - 20-99: Reserved for Oracle E-Business Suite
  - 100-999: Reserved for customer organizations
  - 1000: Public

Access Levels

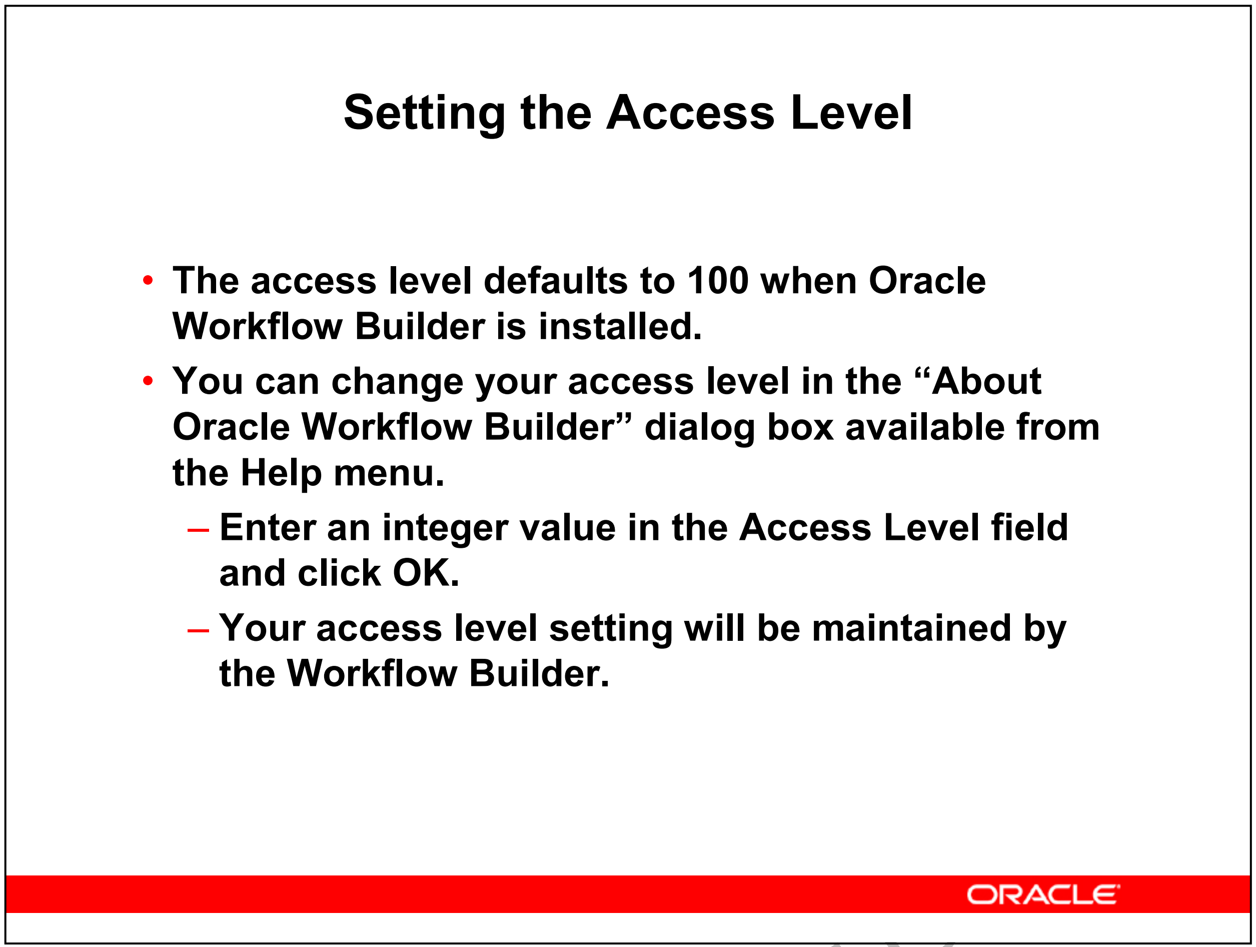
Protection Levels

- If you protect an object against customization, you effectively assign the object a protection level equal to your current access level.
- Objects protected against customizations are considered "seed data."
- Only users operating at an access level equal to or lower than the protection level of the object can modify the object.
- Users operating at an access level greater than the protection level of the object will see a small lock on the icon for the object in the navigator tree, indicating that the object is read-only.

Customization Levels

- If you set an object to be customizable, its protection level is set to 1000.
- The customization level of an object is set to the access level of the initial user who customizes the object.
- A customized object is locked from further modification except from users with access levels equal to the customization level of the object.

- The customization level is relevant only with respect to unprotected workflow objects.
- If an object is protected at a certain level, it shouldn’t be modified at all except by an access level equal to or less than the protected level of the object.



Setting the Access Level

- The access level defaults to 100 when Oracle Workflow Builder is installed.
- You can change your access level in the “About Oracle Workflow Builder” dialog box available from the Help menu.
  - Enter an integer value in the Access Level field and click OK.
  - Your access level setting will be maintained by the Workflow Builder.

Setting Access Levels

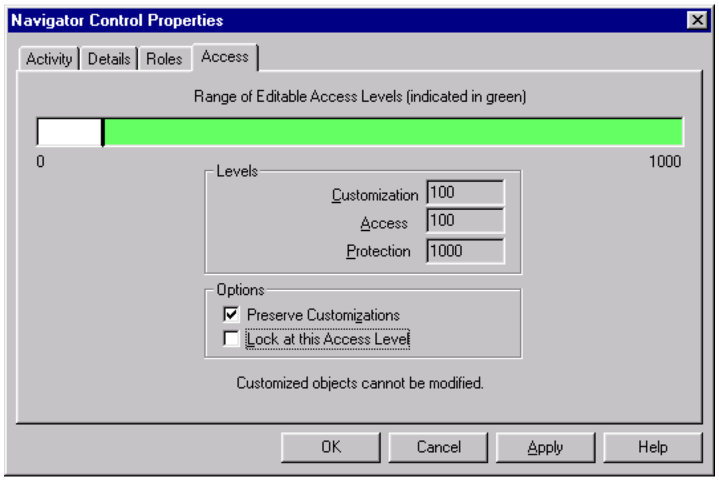
Oracle Workflow Builder Installation

If you previously installed Oracle Workflow Builder on a PC and you change your access level to a value other than 100, reinstalling Oracle Workflow Builder will not modify the access level at which you previously operated.

“Allow Modifications of Customized Objects” Check Box

- Checked: The Workflow Builder saves your edits, overwriting protected objects that you have access to modify as well as any previously customized objects (equivalent to Workflow Definitions Loader Upload mode).
- Unchecked: The Workflow Builder saves edits only to protected objects that you have access to change and does not overwrite previously customized objects (equivalent to Workflow Definitions Loader Upgrade mode).

Setting Protection and Customization Levels



Range of Editable Access Levels (indicated in green)

0 1000

Levels

Customization	100
Access	100
Protection	1000

Options

- [✓] Preserve Customizations
- [ ] Lock at this Access Level

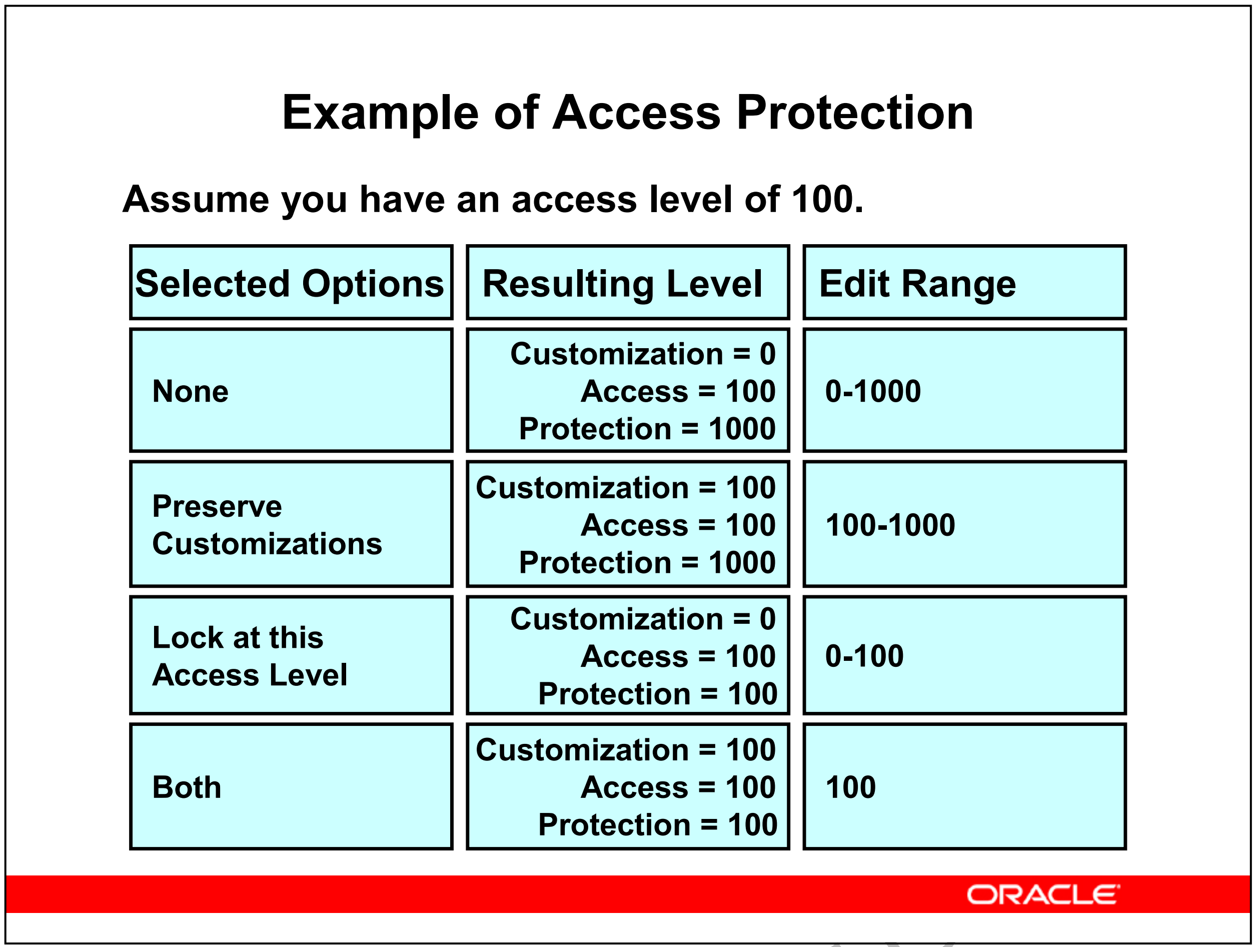
Customized objects cannot be modified.

Access Properties Page

In the Oracle Workflow Builder, select the Access tab for an object to display the Access property page.

- The indicator bar provides a visual range of access levels that can edit the object.
  - Black vertical line: Current access level
  - White range: Cannot edit the object
  - Solid green: Can edit the object
  - Cross-hatch green: Usually cannot modify the object because it has been customized, but can now do so because Oracle Workflow Builder is set to Upload mode so that customized objects can be modified
- The Levels region shows the Customization, Access, and Protection levels of the object based on how you set the check boxes in the Options region.
- Use the Options region to set the protection/customization level of an object.
  - Preserve Customizations: Prevents customized objects from being overwritten during a workflow definition upgrade

- Lock at this Access Level: Protects the object at the current access level and does not allow the object to be customized by higher access levels



Example of Access Protection

Assume you have an access level of 100.

Selected Options	Resulting Level	Edit Range

None	Customization = 0
	Access = 100
	Protection = 1000
		0-1000

Preserve Customizations	Customization = 100
	Access = 100
	Protection = 1000
		100-1000

Lock at this Access Level	Customization = 0
	Access = 100
	Protection = 100
		0-100

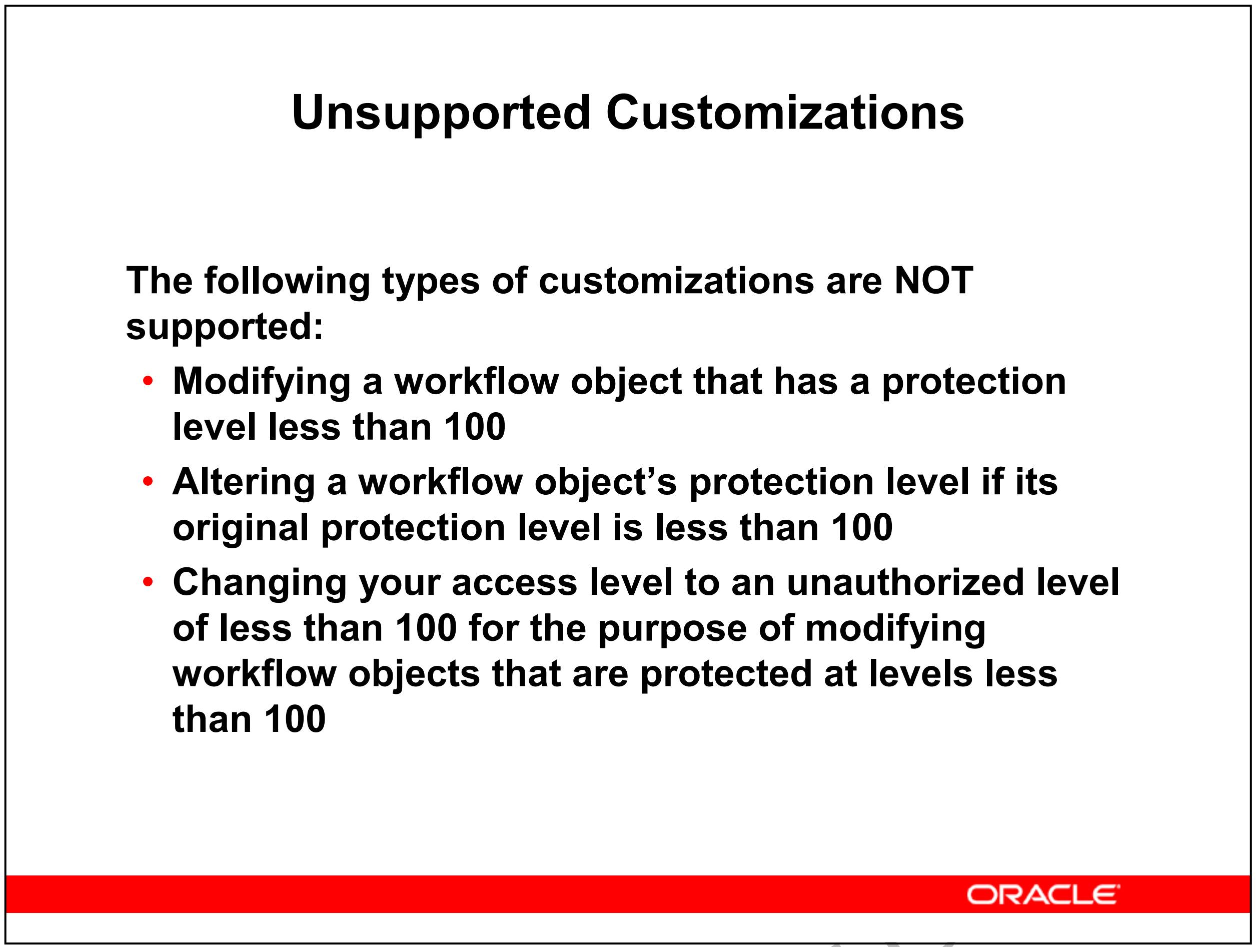
Both	Customization = 100
	Access = 100
	Protection = 100
		100

Protection and Customization Levels

Assuming an access level of 100, these protection and customization levels result when the following check boxes are selected in the Access properties region:

- None: Object can be updated at any time by any access level.
- Preserve Customizations: Disallows customized objects from being overwritten during a workflow upgrade.
  - Object may be updated by access levels 100-1000.
  - If the Allow modifications of customized objects check box is selected, customized objects can also be updated by access levels 0-99 as represented by green crosshatches in the indicator bar.
- Lock at this Access Level: Protects the object at the current access level and does not allow the object to be customized, except by access levels 0-100.
- Both: Object can only be updated by the access level at which the object is protected.
  - Object may only be updated by access level 100.

- If the Allow Modifications of Customized Objects check box is selected, customized objects can also be updated by access levels 0-99 as represented by green crosshatches in the indicator bar.



Unsupported Customizations

The following types of customizations are NOT supported:

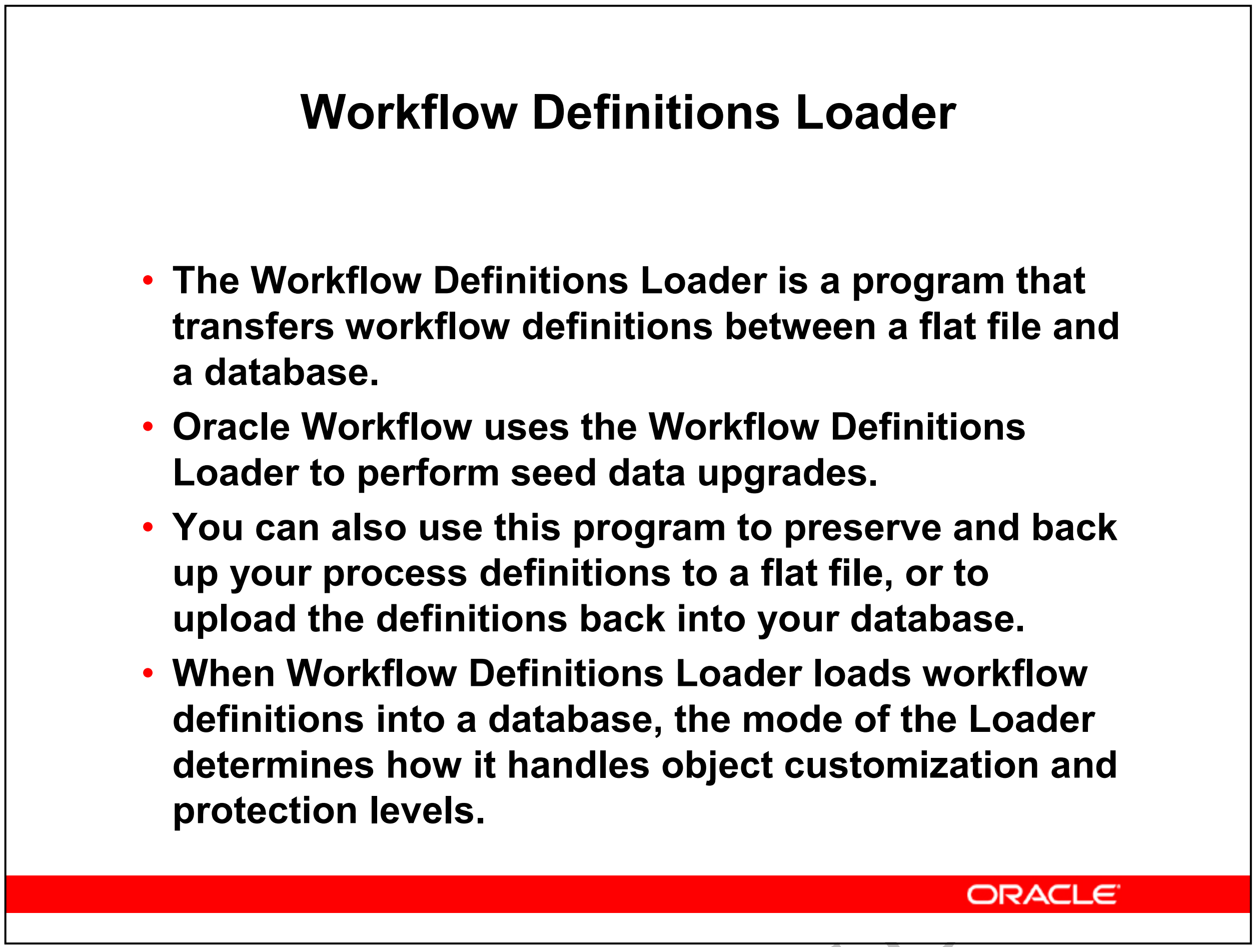
- **Modifying a workflow object that has a protection level less than 100**
- **Altering a workflow object's protection level if its original protection level is less than 100**
- **Changing your access level to an unauthorized level of less than 100 for the purpose of modifying workflow objects that are protected at levels less than 100**

ORACLE

Unsupported Customizations

The following types of customizations are also NOT supported:

- Customizations that are explicitly documented as being UNSUPPORTED in the seeded workflow's product-specific user's guide or documentation update notes. This includes modifying processes, attributes, function activities, event activities, notifications, lookup types, messages, events, or subscriptions that are specifically documented as not to be modified.
- Manual modifications of Workflow tables with a prefix of WF_ or FND_ , unless documented in the Oracle Workflow Guide or required by Oracle Support Services.



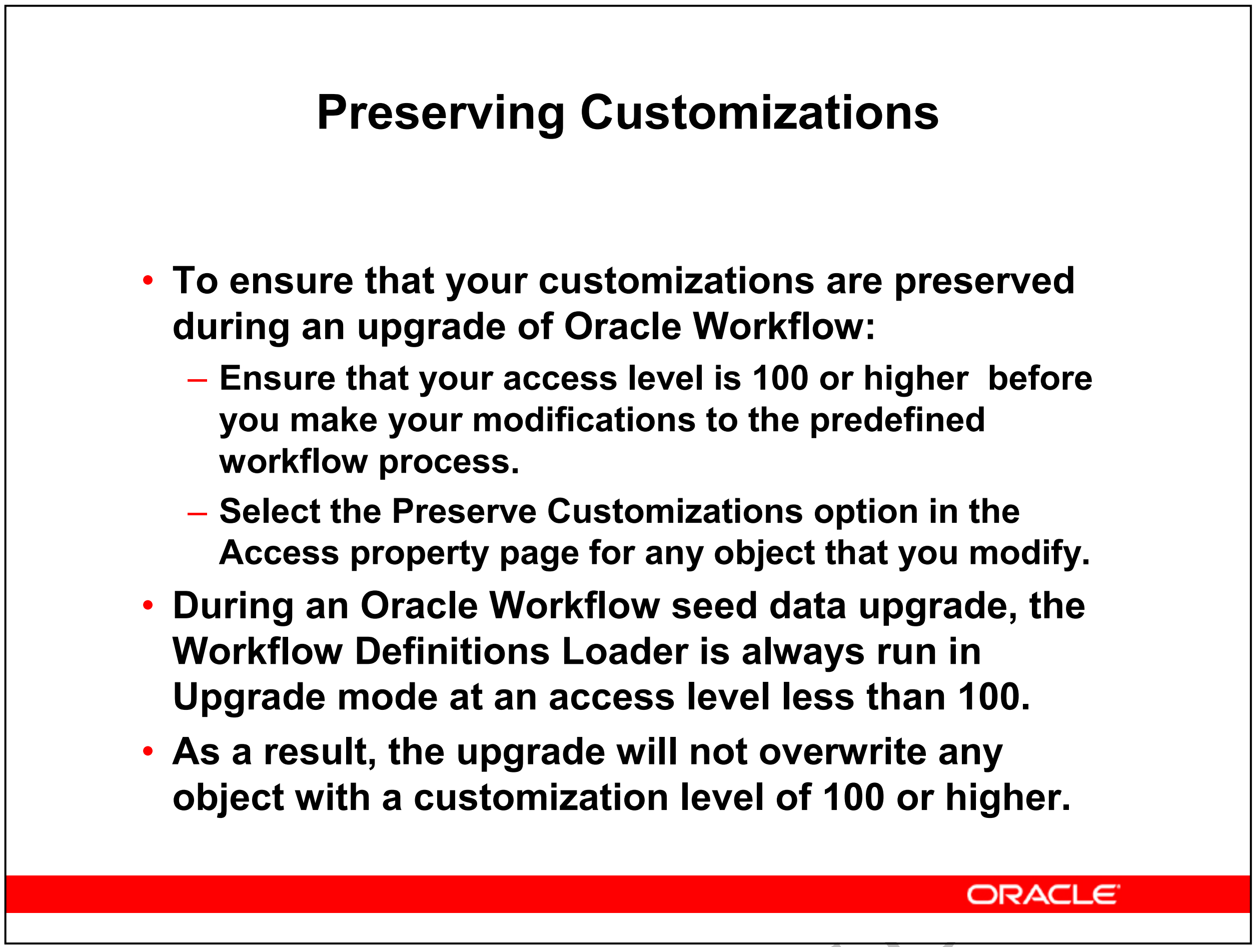
Workflow Definitions Loader

- The Workflow Definitions Loader is a program that transfers workflow definitions between a flat file and a database.
- Oracle Workflow uses the Workflow Definitions Loader to perform seed data upgrades.
- You can also use this program to preserve and back up your process definitions to a flat file, or to upload the definitions back into your database.
- When Workflow Definitions Loader loads workflow definitions into a database, the mode of the Loader determines how it handles object customization and protection levels.

Workflow Definitions Loader

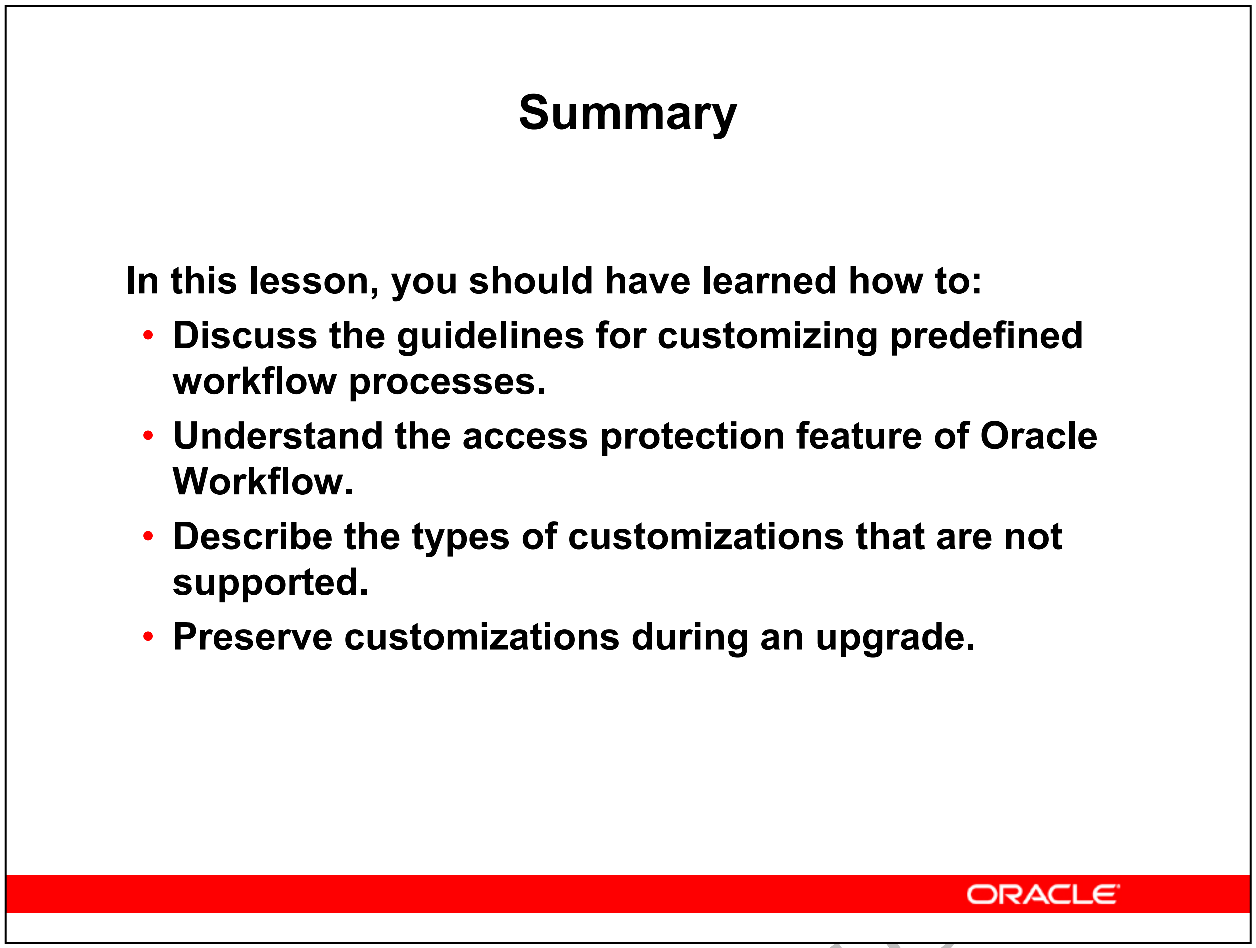
The Loader can be run in the following modes:

- Upgrade: Upgrade to a definition in an input file, preserving customizations, using the access level in the input file.
- Upload: Upload the definition from an input file, ignoring preserve customizations settings, using the access level in the input file.
- Force: Force the upload of a definition from an input file to a database regardless of an object's protection level.
- Download: Download specified item type definitions from the database to an output file.



Preserving Customizations

- To ensure that your customizations are preserved during an upgrade of Oracle Workflow:
  - Ensure that your access level is 100 or higher before you make your modifications to the predefined workflow process.
  - Select the Preserve Customizations option in the Access property page for any object that you modify.
- During an Oracle Workflow seed data upgrade, the Workflow Definitions Loader is always run in Upgrade mode at an access level less than 100.
- As a result, the upgrade will not overwrite any object with a customization level of 100 or higher.



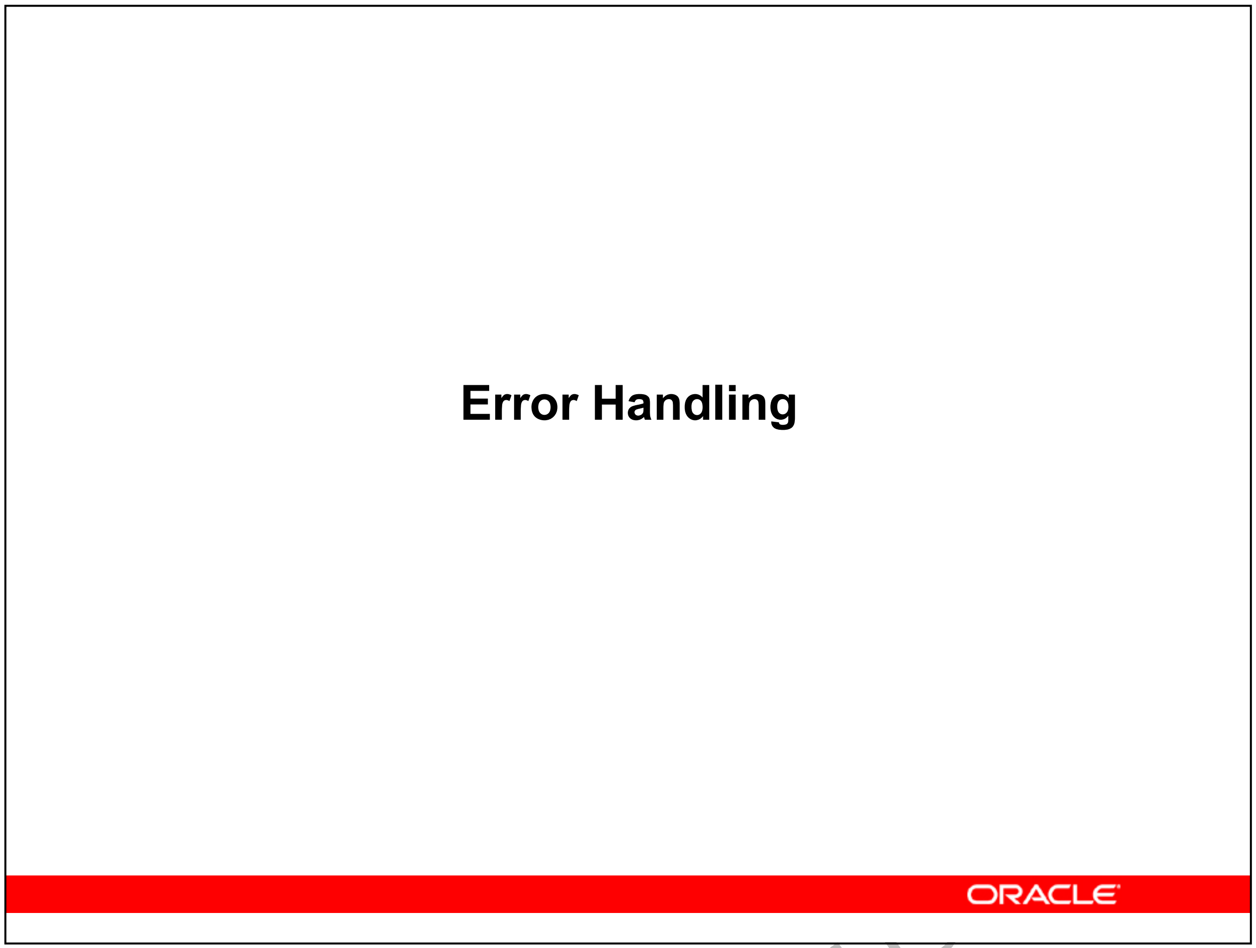
Summary

In this lesson, you should have learned how to:

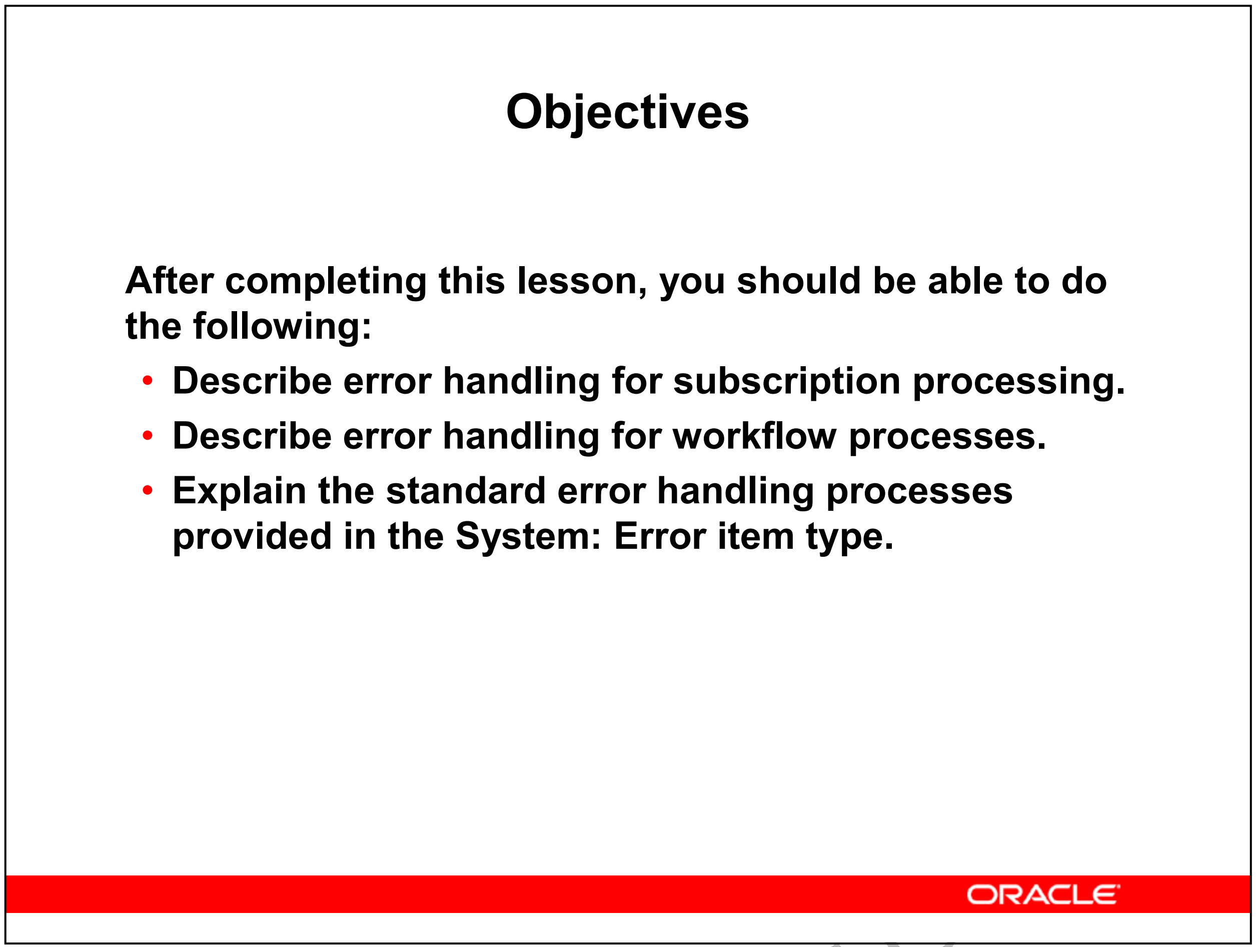
- Discuss the guidelines for customizing predefined workflow processes.
- Understand the access protection feature of Oracle Workflow.
- Describe the types of customizations that are not supported.
- Preserve customizations during an upgrade.

Error Handling

Chapter 19



Error Handling

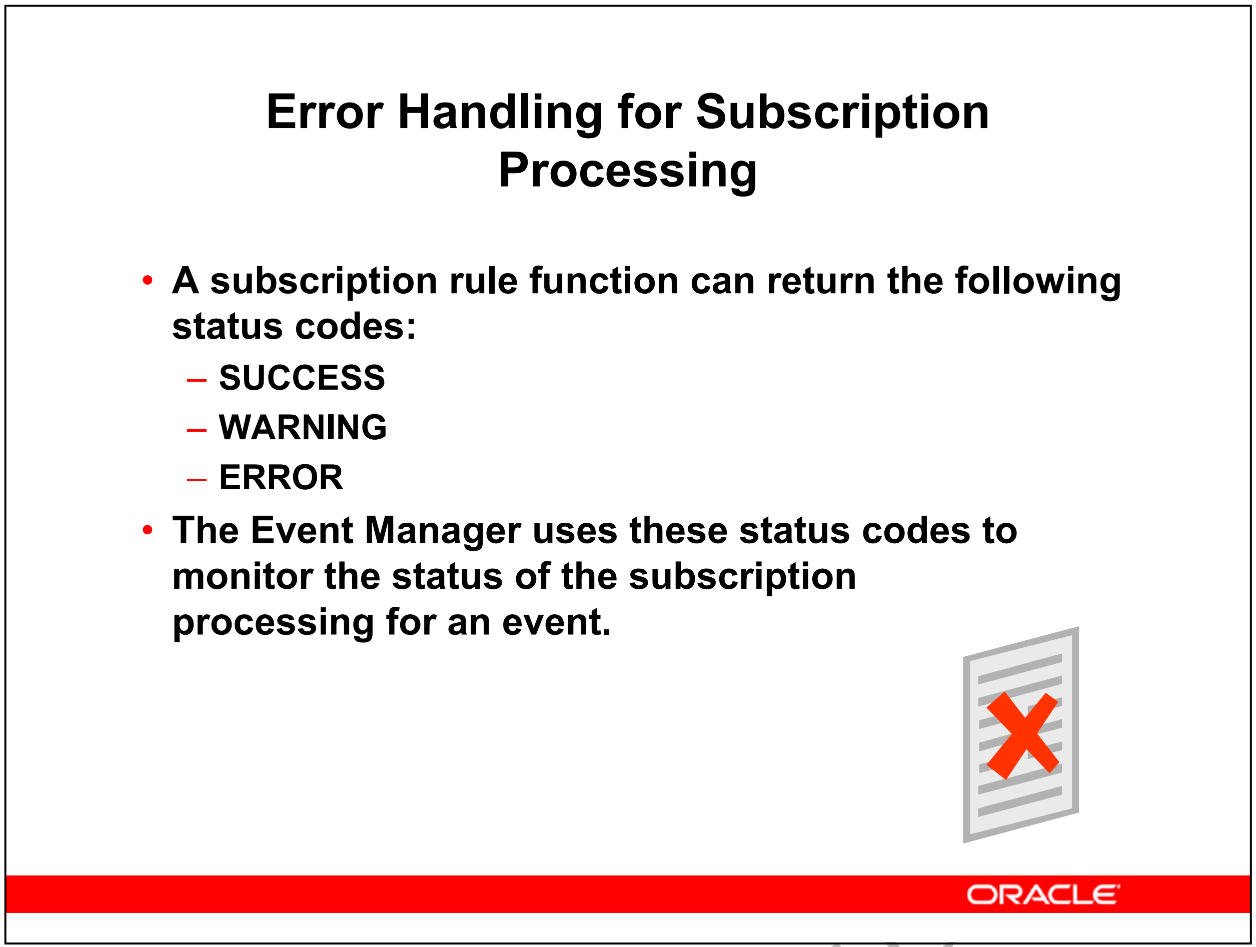


Objectives

After completing this lesson, you should be able to do the following:

- Describe error handling for subscription processing.
- Describe error handling for workflow processes.
- Explain the standard error handling processes provided in the System: Error item type.

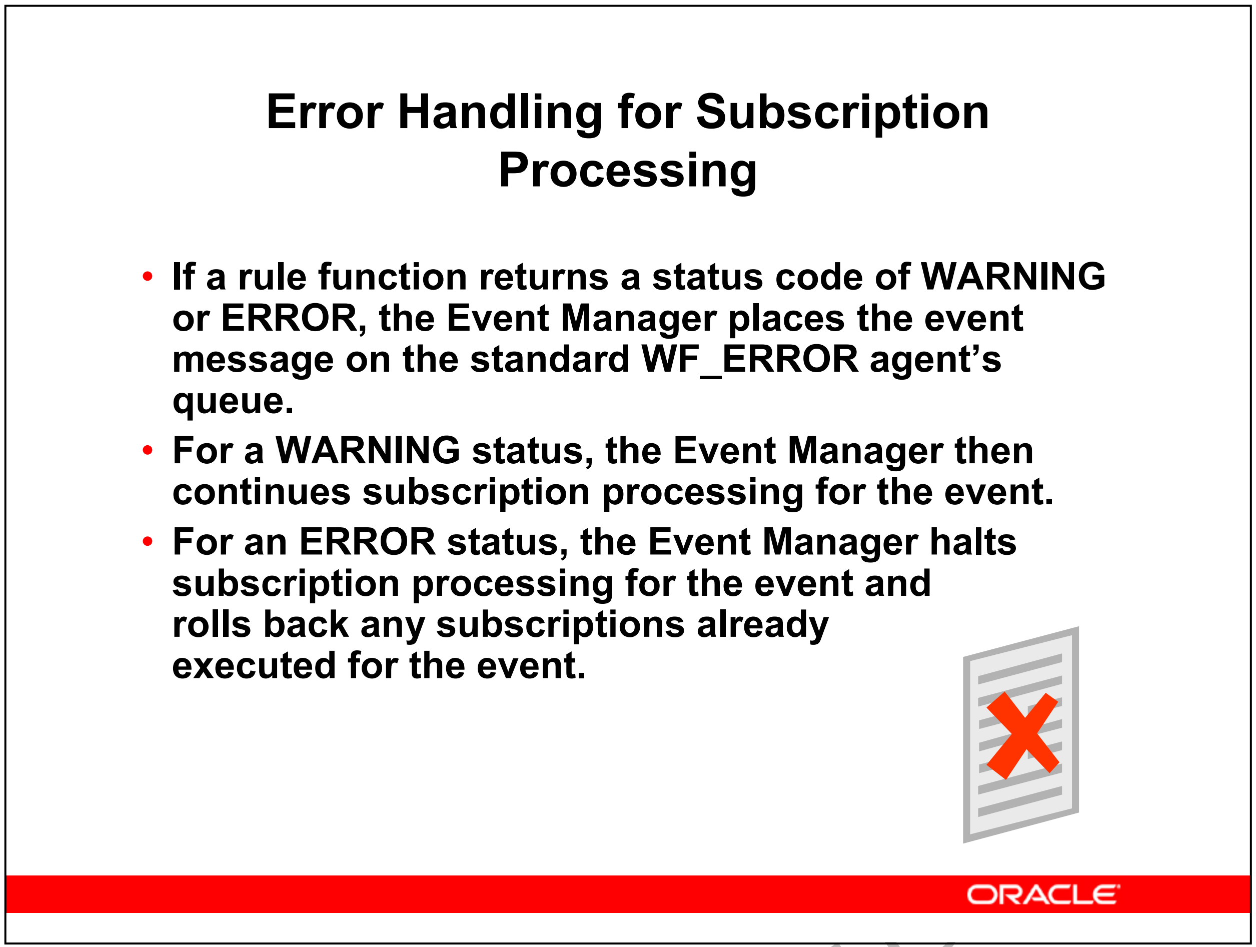
ORACLE



Error Handling for Subscription Processing

- **A subscription rule function can return the following status codes:**
  - **SUCCESS**
  - **WARNING**
  - **ERROR**
- **The Event Manager uses these status codes to monitor the status of the subscription processing for an event.**

ORACLE

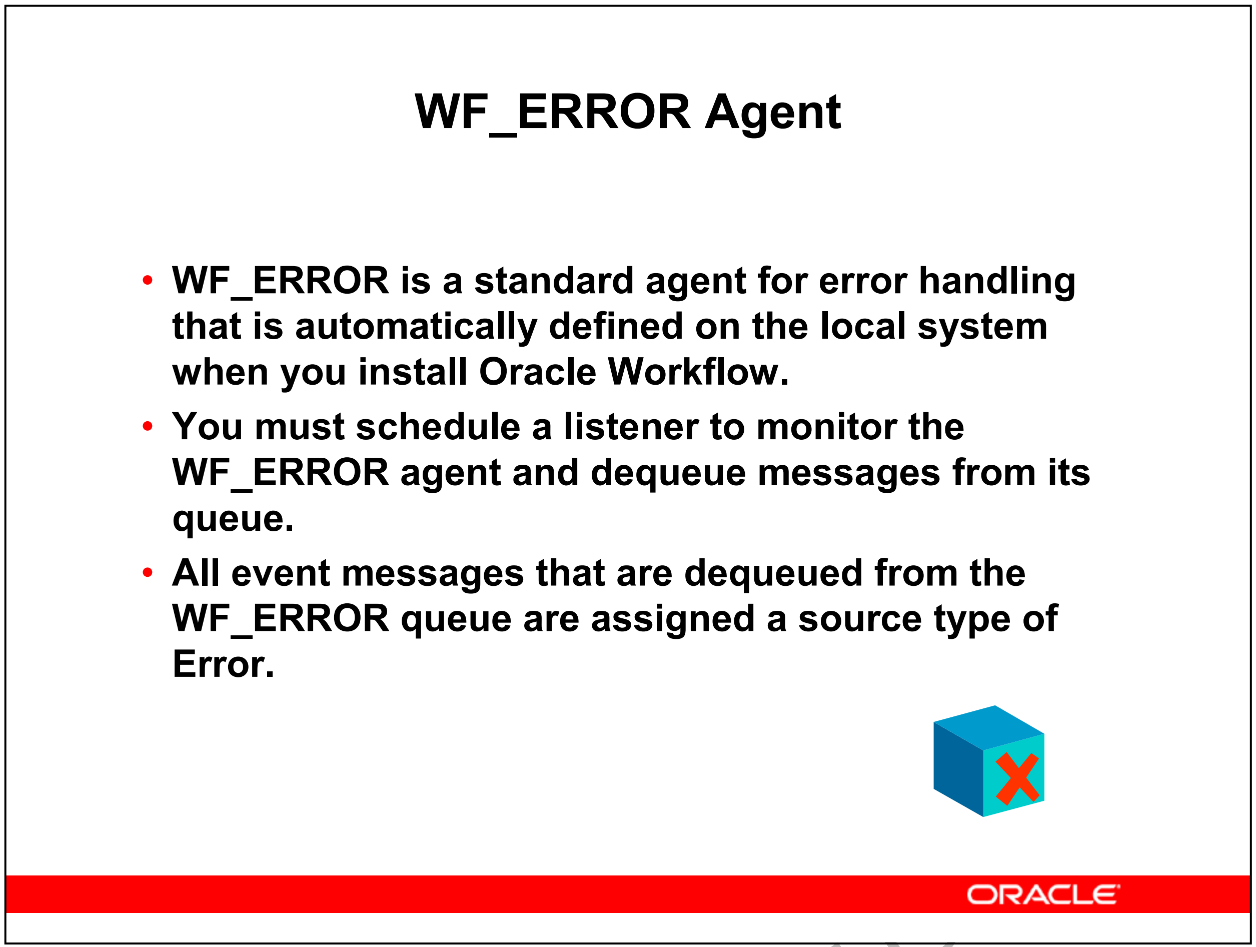


Error Handling for Subscription Processing

- If a rule function returns a status code of WARNING or ERROR, the Event Manager places the event message on the standard WF_ERROR agent's queue.
- For a WARNING status, the Event Manager then continues subscription processing for the event.
- For an ERROR status, the Event Manager halts subscription processing for the event and rolls back any subscriptions already executed for the event.

Errors in Subscription Processing

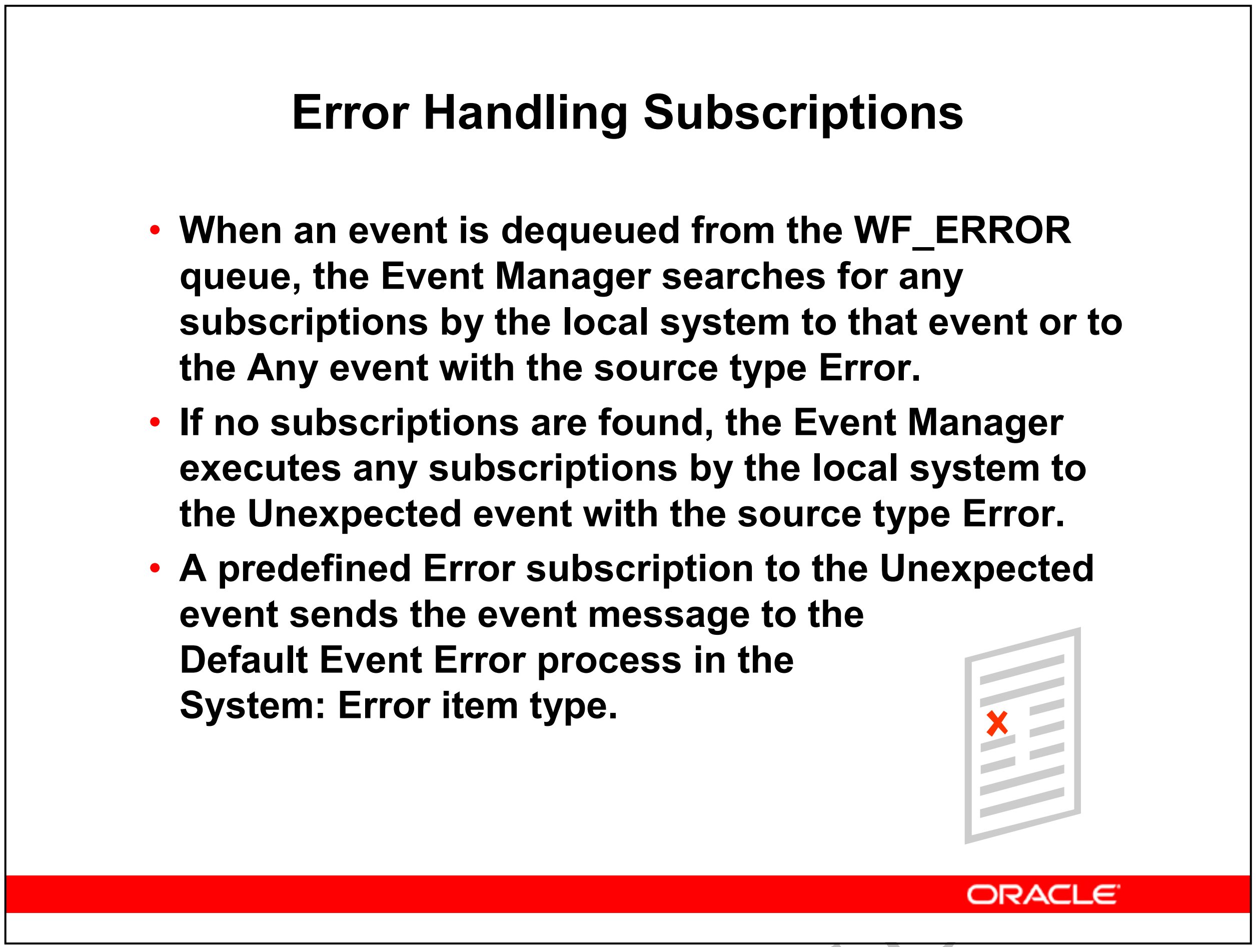
Note: If a rule function raises an exception, the Event Manager rolls back all subscription processing for the event and raises the error to the calling application. In this case the event message is not placed on the WF_ERROR queue.



WF_ERROR Agent

- WF_ERROR is a standard agent for error handling that is automatically defined on the local system when you install Oracle Workflow.
- You must schedule a listener to monitor the WF_ERROR agent and dequeue messages from its queue.
- All event messages that are dequeued from the WF_ERROR queue are assigned a source type of Error.

ORACLE



Error Handling Subscriptions

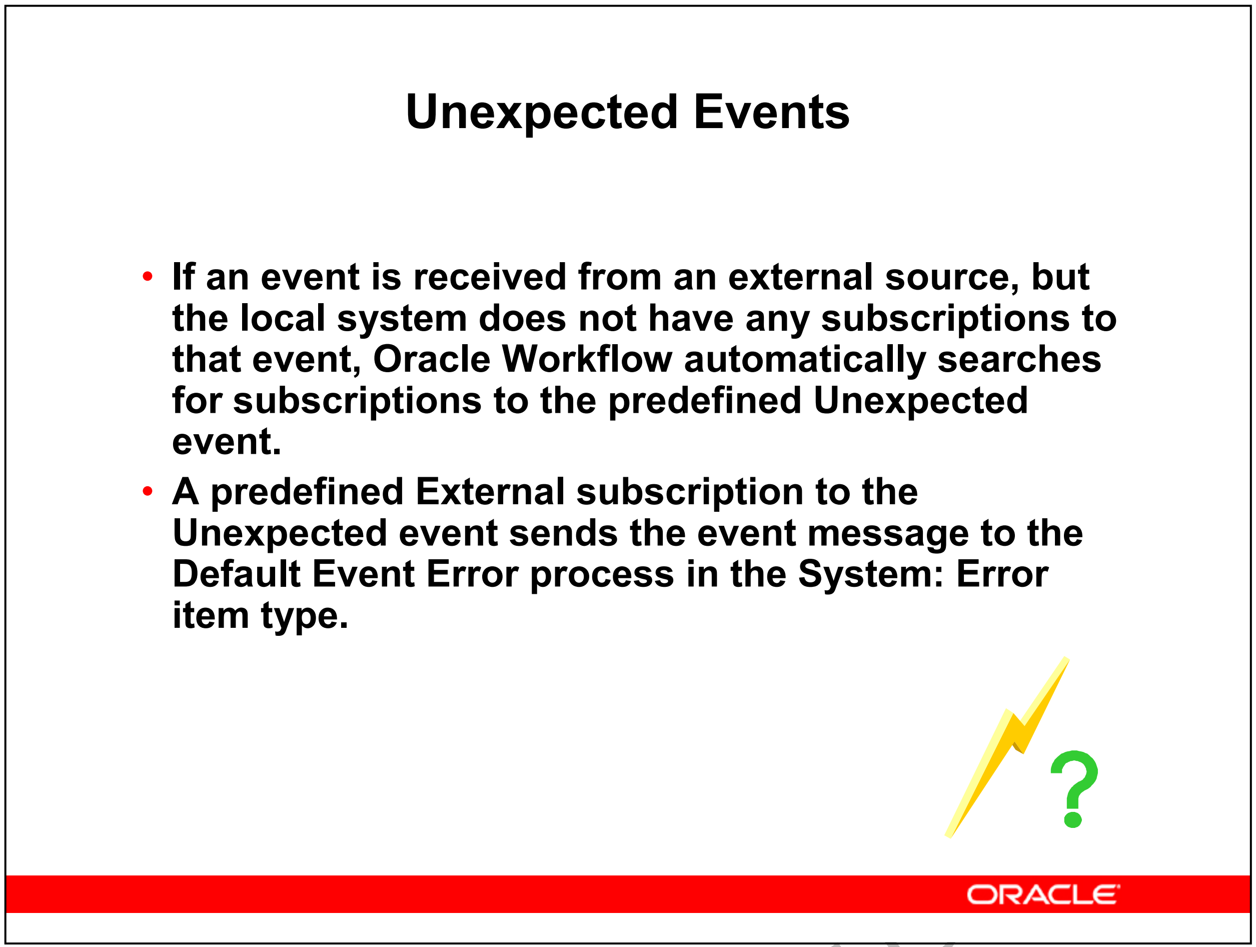
- When an event is dequeued from the WF_ERROR queue, the Event Manager searches for any subscriptions by the local system to that event or to the Any event with the source type Error.
- If no subscriptions are found, the Event Manager executes any subscriptions by the local system to the Unexpected event with the source type Error.
- A predefined Error subscription to the Unexpected event sends the event message to the Default Event Error process in the System: Error item type.

ORACLE

Error Handling Subscriptions

You must not change or disable the definition of the Unexpected event or of the predefined Error subscription to that event. If you disable this subscription, then the Event Manager will not be able to perform error handling for any events for which you have not defined custom Error subscriptions.

You can set up custom error handling for a particular event by defining a subscription to that event with a source type of Error and specifying the custom processing you want to execute as the subscription action.



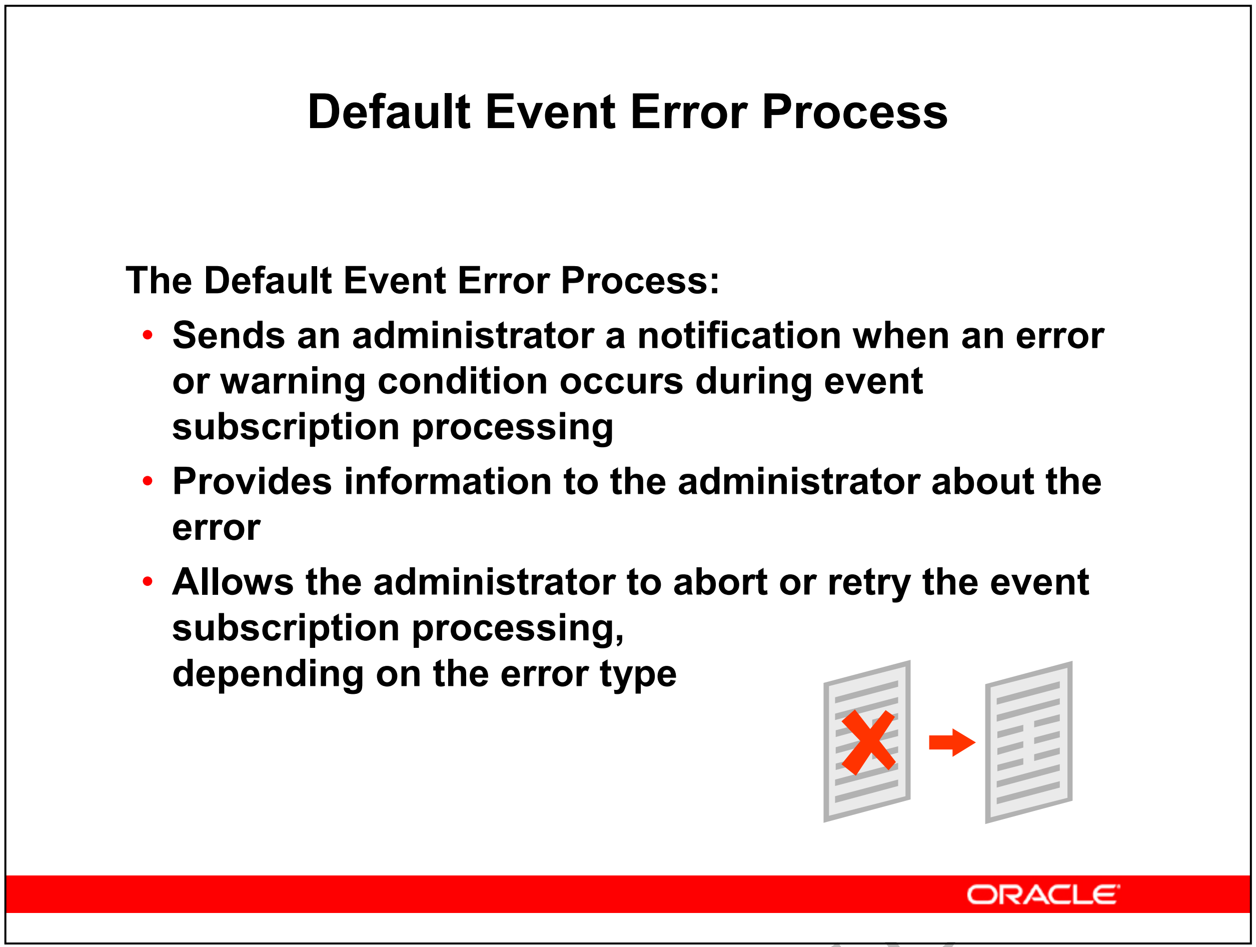
Unexpected Events

- If an event is received from an external source, but the local system does not have any subscriptions to that event, Oracle Workflow automatically searches for subscriptions to the predefined Unexpected event.
- A predefined External subscription to the Unexpected event sends the event message to the Default Event Error process in the System: Error item type.

Unexpected Events

Oracle Workflow also provides a predefined Local subscription to the Unexpected event that sends the event message to the Default Event Error process when there are no subscriptions to a locally raised event. This subscription is disabled by default. To use this subscription, you must enable it.

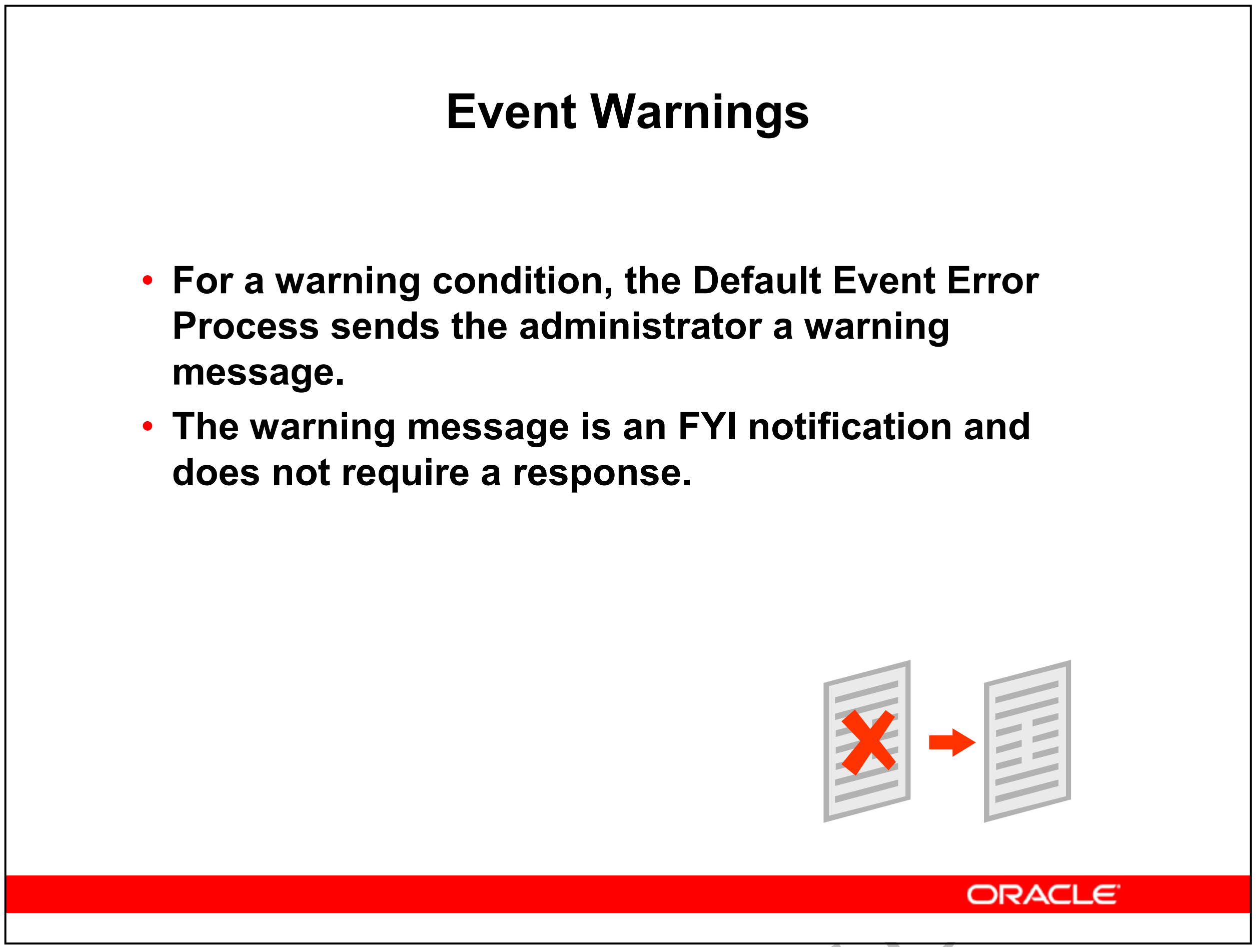
Attention: If you want to enable this subscription, be careful to consider all the events that can be raised on your local system and trigger the subscription. Many local events may be raised to which you do not want to subscribe. Additionally, if a large number of events are raised on the local system, enabling this subscription may flood the Business Event System.



Default Event Error Process

The Default Event Error Process:

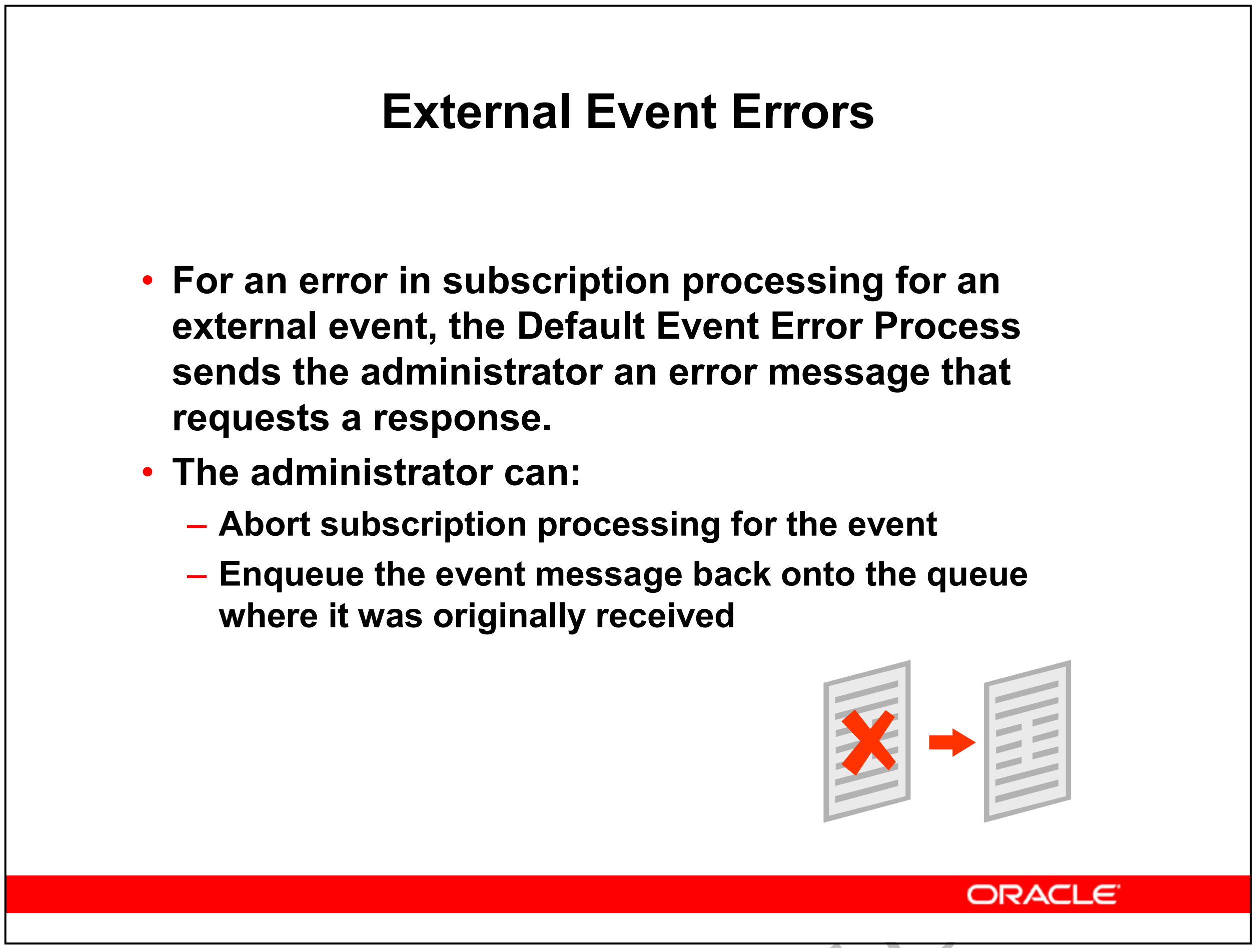
- Sends an administrator a notification when an error or warning condition occurs during event subscription processing
- Provides information to the administrator about the error
- Allows the administrator to abort or retry the event subscription processing, depending on the error type



Event Warnings

- For a warning condition, the Default Event Error Process sends the administrator a warning message.
- The warning message is an FYI notification and does not require a response.

ORACLE



External Event Errors

- For an error in subscription processing for an external event, the Default Event Error Process sends the administrator an error message that requests a response.
- The administrator can:
  - Abort subscription processing for the event
  - Enqueue the event message back onto the queue where it was originally received

External Event Errors

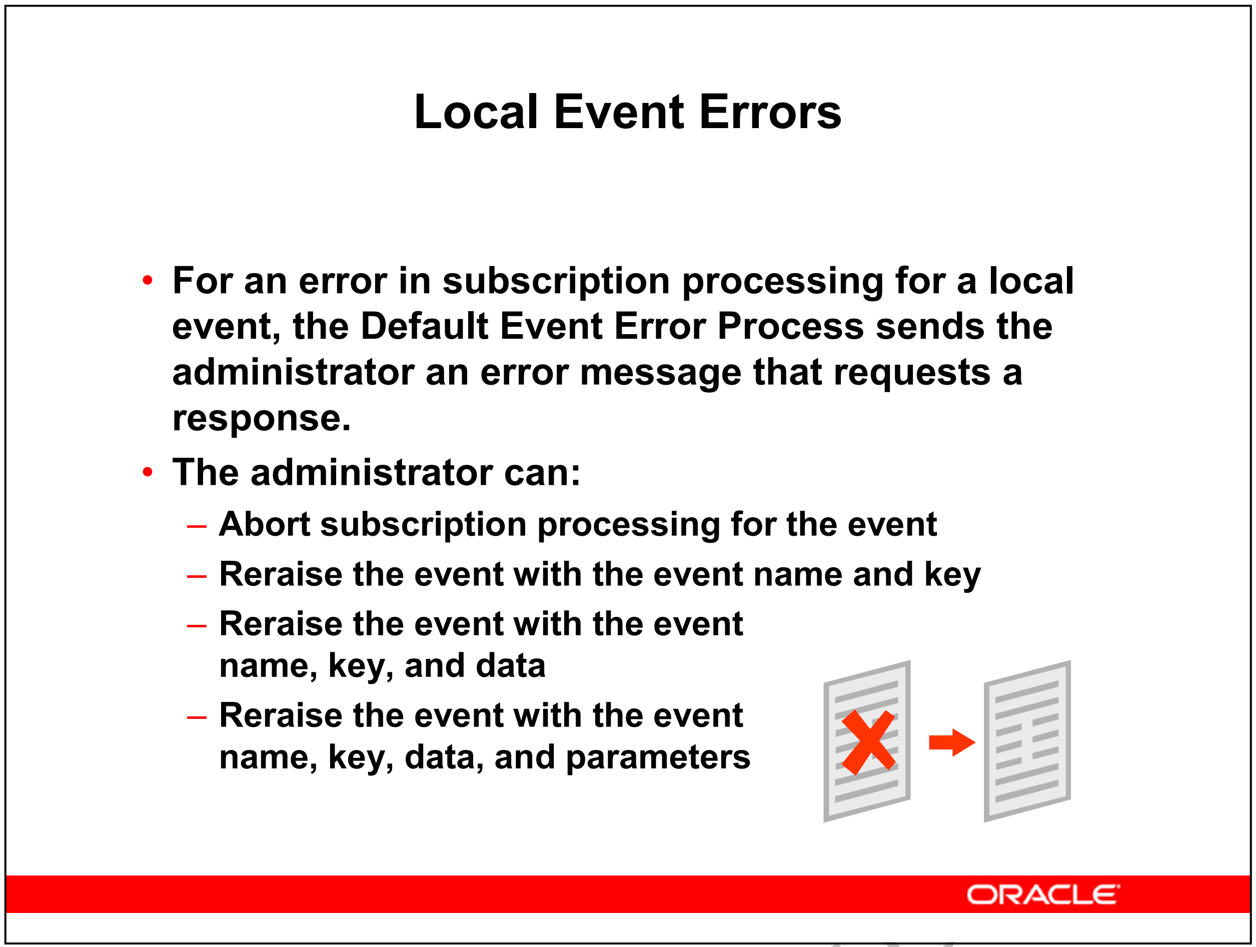
Abort

For example, if the event data contained in an event message is corrupted, the system administrator can abort subscription processing on that event message.

Enqueue Event

The event message is enqueued on its original queue with a priority of -1 so that it will be the first message to be dequeued the next time the listener runs.

The system administrator can attempt to correct the error before re-enqueuing the event. For example, the system administrator can create a subscription to handle an unexpected event and then re-enqueue the event message to trigger the new subscription.



Local Event Errors

- For an error in subscription processing for a local event, the Default Event Error Process sends the administrator an error message that requests a response.
- The administrator can:
  - Abort subscription processing for the event
  - Reraise the event with the event name and key
  - Reraise the event with the event name, key, and data
  - Reraise the event with the event name, key, data, and parameters

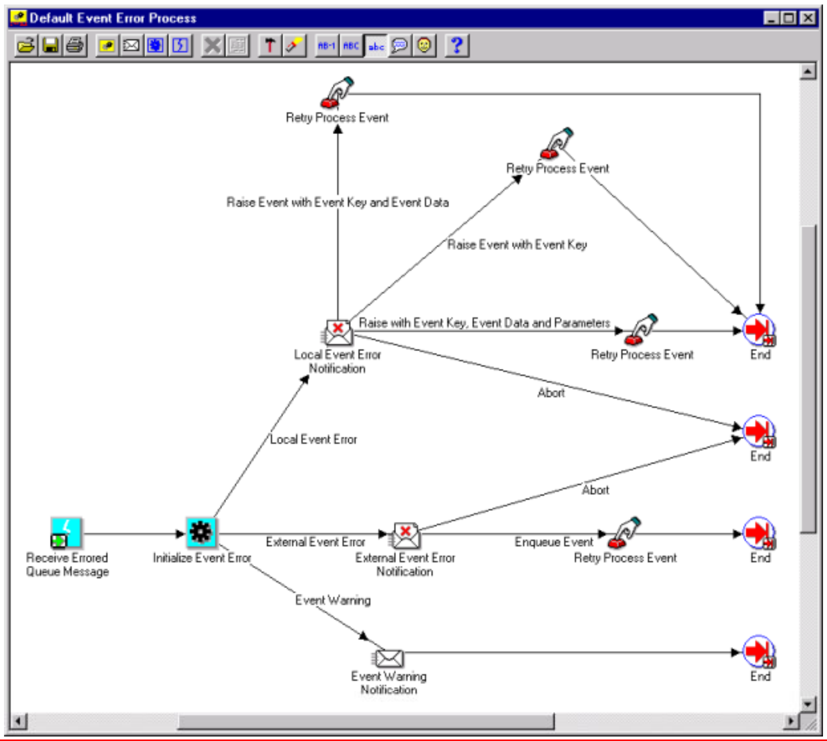
Local Event Errors

The system administrator can choose the level of information to provide to the Event Manager when reraising the event. For example, if an error exists in the event data that was originally provided, the event can be reraised with only the event name and the event key, forcing the Event Manager to regenerate the event data using the event's Generate function.

The system administrator can also attempt to correct the error before reraising the event.

Default Event Error Process

Default Event Error Process

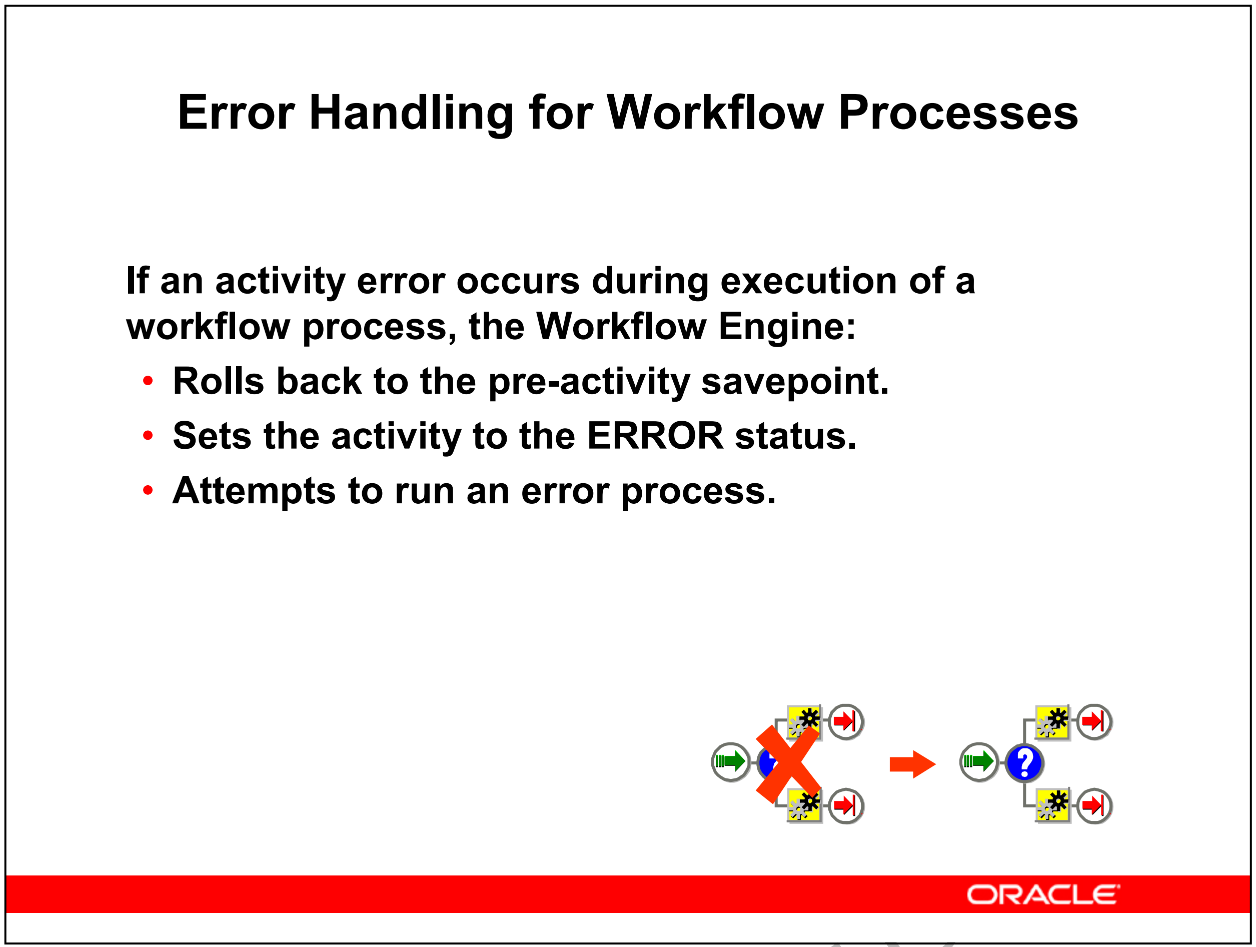


Default Event Error Process
Retry Process Event
Raise Event with Event Key and Event Data
Retry Process Event
Raise Event with Event Key
Raise with Event Key, Event Data and Parameters
Local Event Error Notification
Retry Process Event
End
Abort
End
Local Event Error
Abort
Receive Errored Queue Message
Initialize Event Error
External Event Error
External Event Error Notification
Enqueue Event
Retry Process Event
End
Event Warning
Event Warning Notification
End

Default Event Error Process

- The Default Event Error Process begins when the Event Manager sends an errored event message to the process.
- The workflow begins with the Receive Errored Queue Message activity.
- Next, the process determines the error type:
  - Event Warning
  - External Event Error
  - Local Event Error
- The process notifies the administrator of the error, sending different messages depending on the error type.
- For an Event Warning condition, the process ends after the notification is sent.
- For an External Event Error, the next activity depends on the administrator’s response.
  - Abort—The process ends.
  - Enqueue Event—The process retries subscription processing for the event by re-enqueuing the event on its original queue. Then the process ends.
- For a Local Event Error, the next activity also depends on the administrator’s response.

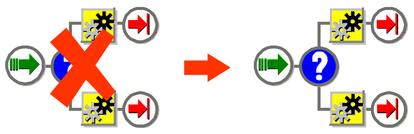
- Abort—The process ends.
- Raise Event with Event Key, Raise Event with Event Key and Event Data, or Raise with Event Key, Event Data, and Parameters—The process retries subscription processing for the event by reraising the event with the specified level of information. Then the process ends.



Error Handling for Workflow Processes

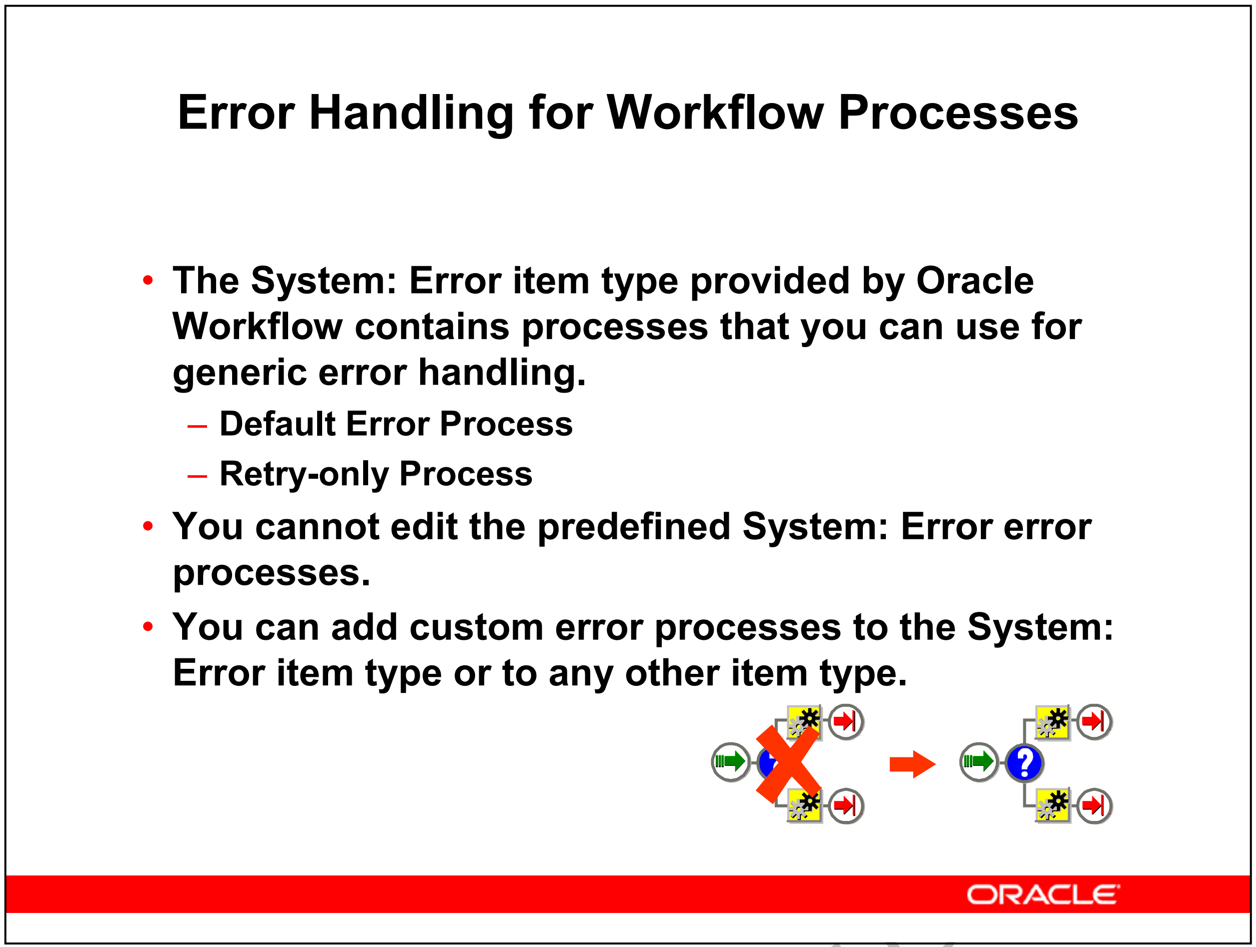
If an activity error occurs during execution of a workflow process, the Workflow Engine:

- **Rolls back to the pre-activity savepoint.**
- **Sets the activity to the ERROR status.**
- **Attempts to run an error process.**



Error Processing

The Workflow Engine attempts to locate an error process to run by starting with the activity which caused the error, and then checking each parent process activity until an associated error process is located. An error process can be associated with an activity in the activity details property page in the Workflow Builder.



Error Handling for Workflow Processes

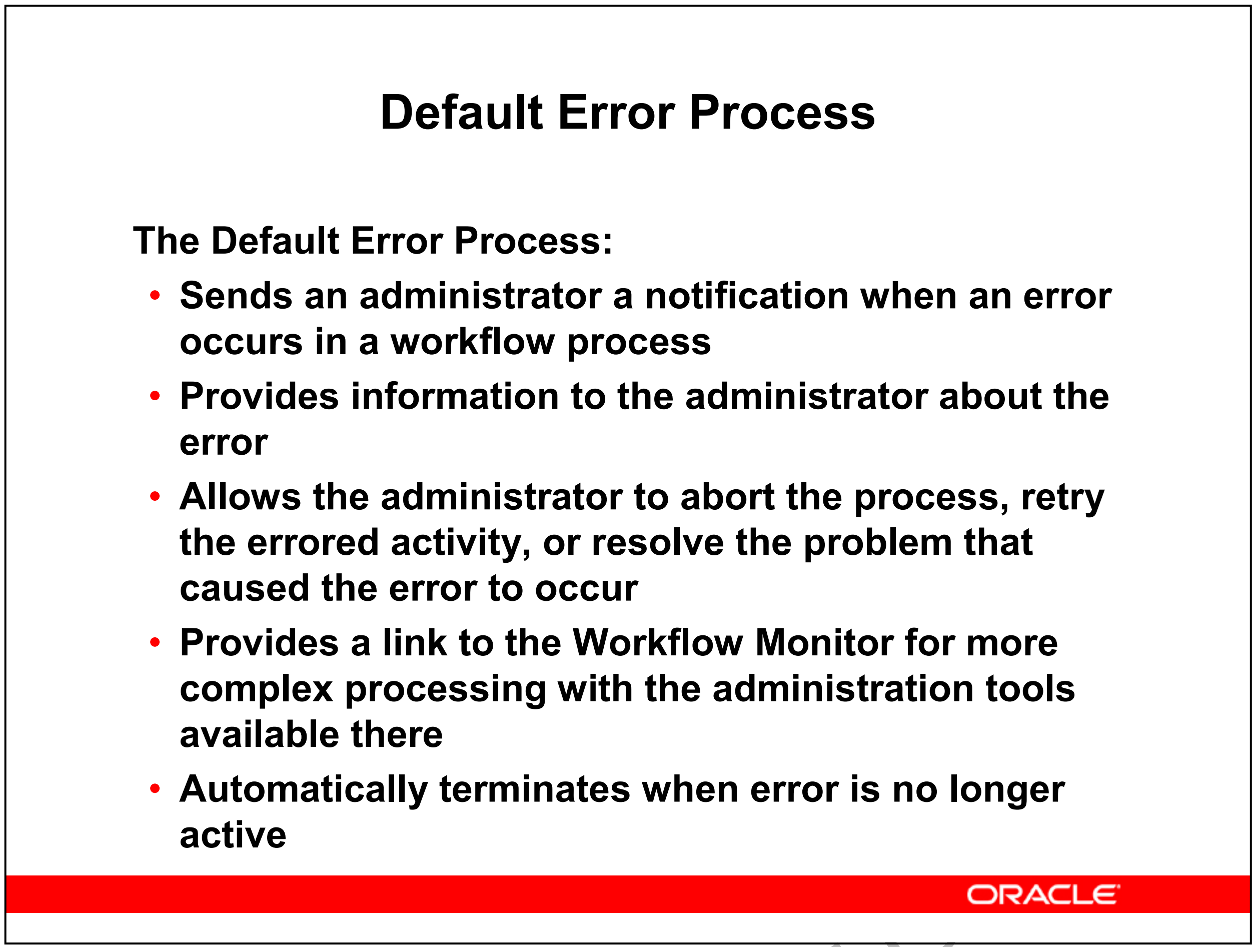
- The System: Error item type provided by Oracle Workflow contains processes that you can use for generic error handling.
  - Default Error Process
  - Retry-only Process
- You cannot edit the predefined System: Error error processes.
- You can add custom error processes to the System: Error item type or to any other item type.

ORACLE

Error Handling

Although you cannot edit the predefined error processes, you can define two item attributes in your item type to control the error processing they perform.

- WF_ADMINISTRATOR: Specify the role to which Oracle Workflow sends the error notification. The default is the System Administrator role.
- ERROR_TIMEOUT: Specify whether the error notification times out.



Default Error Process

The Default Error Process:

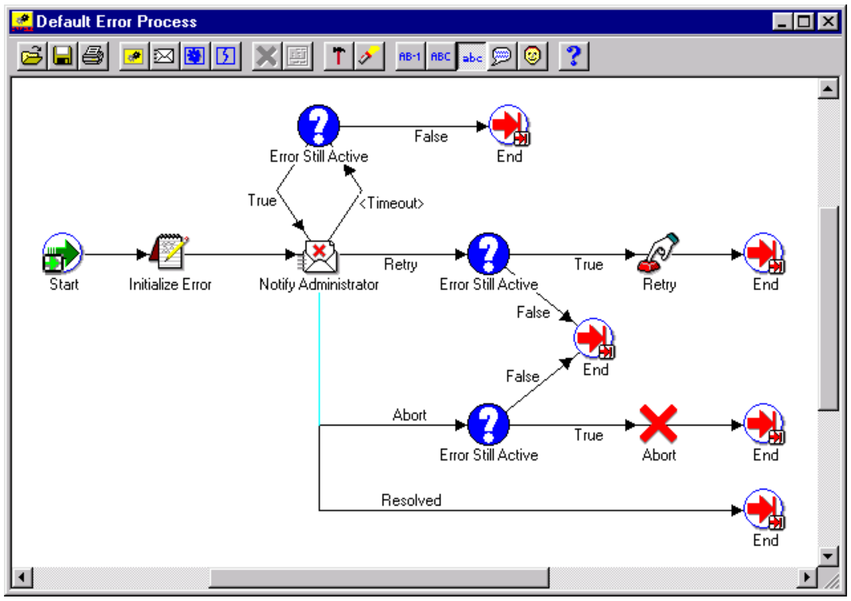
- **Sends an administrator a notification when an error occurs in a workflow process**
- **Provides information to the administrator about the error**
- **Allows the administrator to abort the process, retry the errored activity, or resolve the problem that caused the error to occur**
- **Provides a link to the Workflow Monitor for more complex processing with the administration tools available there**
- **Automatically terminates when error is no longer active**

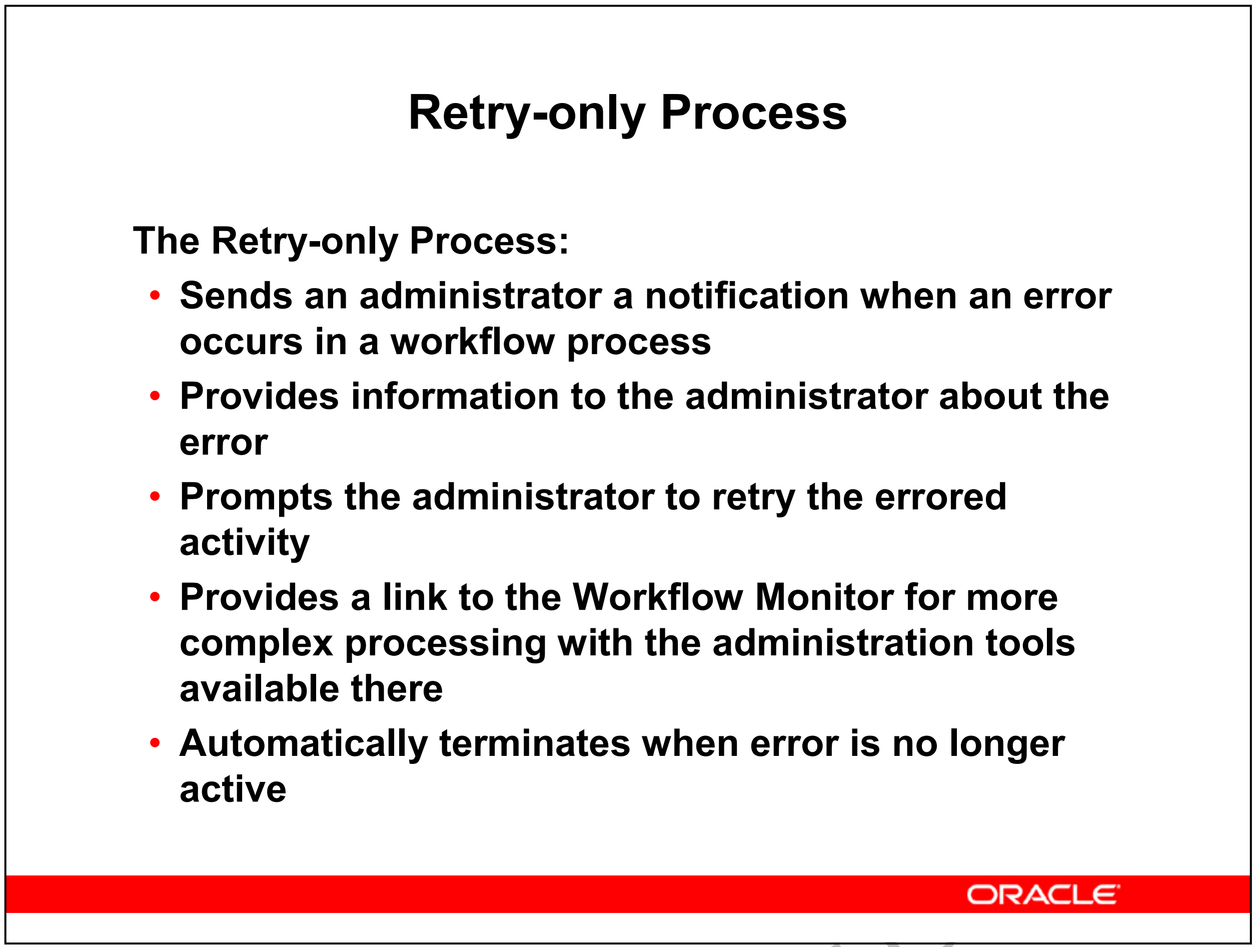
Default Error Process

The notification to the administrator includes the following information:

- A request to either retry or abort the process
- Item Type
- Item Key
- User Key
- Error Name
- Error Message
- Error Stack
- Activity ID
- Activity Label
- Result Code
- Notification ID
- Assigned User
- Monitor URL: Enables the administrator to navigate to the error process

Default Error Process





Retry-only Process

The Retry-only Process:

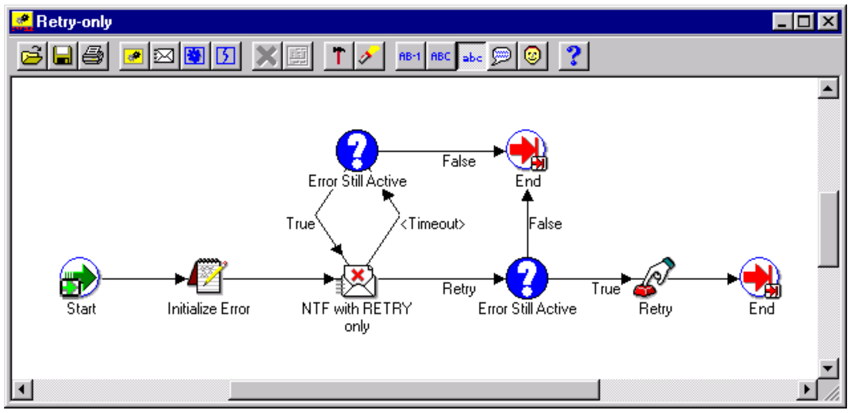
- **Sends an administrator a notification when an error occurs in a workflow process**
- **Provides information to the administrator about the error**
- **Prompts the administrator to retry the errored activity**
- **Provides a link to the Workflow Monitor for more complex processing with the administration tools available there**
- **Automatically terminates when error is no longer active**

Retry-only Process

The notification to the administrator includes the following information:

- A request to either retry or abort the process
- Item Type
- Item Key
- User Key
- Error Name
- Error Message
- Error Stack
- Activity ID
- Activity Label
- Result Code
- Notification ID
- Assigned User
- Monitor URL: Enables the administrator to navigate to the error process

Retry-only Process



Practice - Handling an Error in Subscription Processing

Overview

In this practice, you will define a subscription to deliberately return an error in order to test subscription processing error handling.

- Define a subscription that uses the standard WF_RULE.Error rule function to deliberately return an ERROR status code.
- Raise an event to trigger the subscription. Oracle Workflow will send the errored event to the Default Event Error Process, which will send a notification of the error.
- Review the notification to confirm that the subscription was executed and to respond to the error by aborting subscription processing for the event.
- Optionally raise another event to generate a new error and respond to the error by modifying the subscription definition.

Note: Because many students access the system and create objects during this course, you need a way to distinguish between the objects created by you and by your classmates. Therefore, you will be assigned a terminal number by your instructor. Use this number as a prefix wherever you see *XX* included in the name of something you are defining. In this way, you can ensure that the definitions you create are unique.

Assumptions

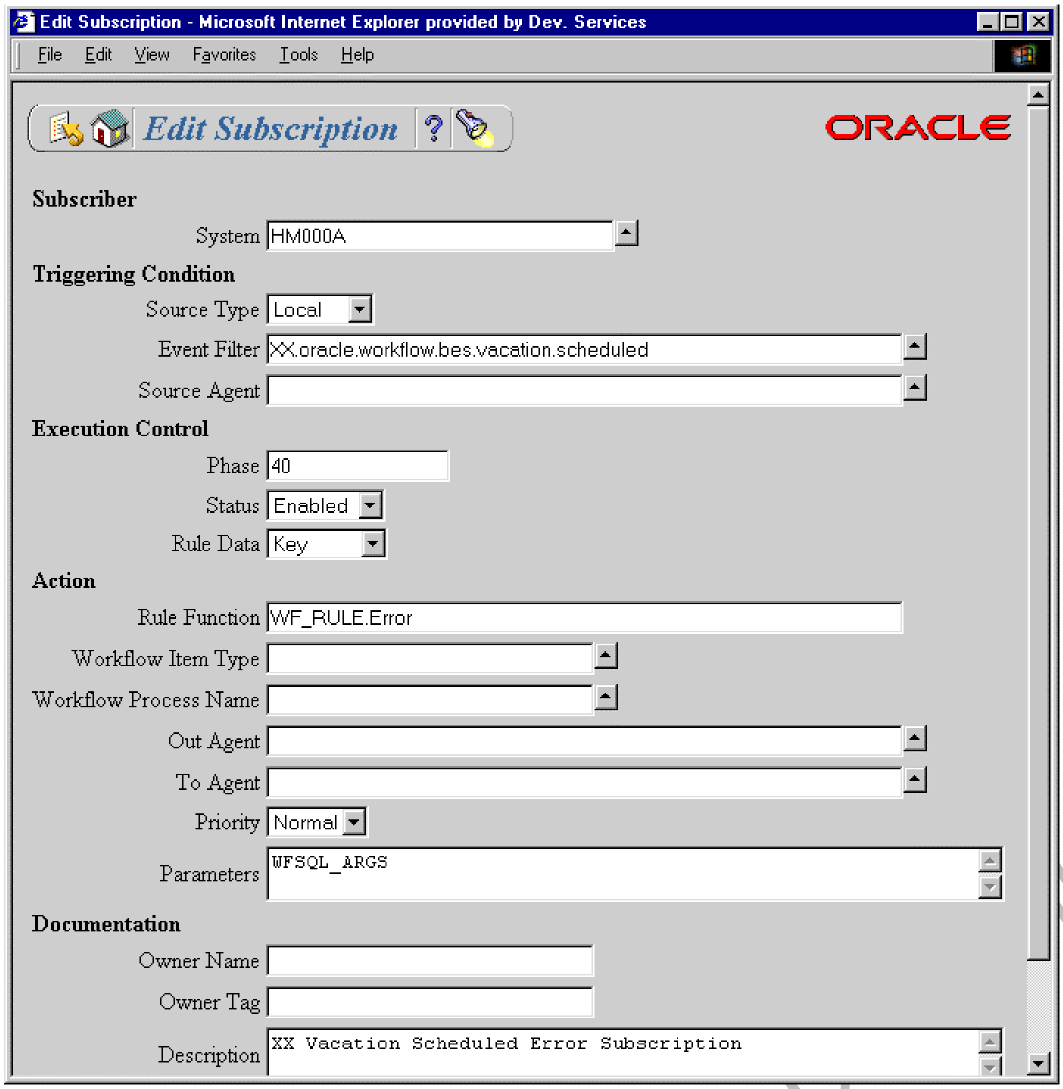
- The instructor will provide you with the username and password of a user with workflow administrator privileges. The workflow administrator is defined in the Global Workflow Preferences page.
- For standalone Oracle Workflow, the instructor will provide you with the URL for the Oracle Workflow home page. The URL is <*webagent*>/wfa_html.home, where <*webagent*> is the base URL of the web agent configured for Oracle Workflow in your Web server.
- For Oracle Workflow embedded in Oracle E-Business Suite, the instructor will provide you with the name of a Workflow administrator responsibility that includes Event Manager functionality. The username you use to log in should have this responsibility assigned to it.

Tasks

1. Define a subscription that uses the standard WF_RULE.Error rule function to deliberately return an ERROR status code.

   Solution:

1. Use a web browser to connect to the Oracle Workflow home page with the URL provided by the instructor for standalone Oracle Workflow, or to a Workflow administrator responsibility provided by the instructor for Oracle Workflow embedded in Oracle E-Business Suite. Log in as a user with workflow administrator privileges.
2. Choose the Event Subscriptions link for standalone Oracle Workflow or the Add Event Subscriptions link for Oracle Workflow embedded in Oracle E-Business Suite.
3. In the Event Subscriptions page, choose the Add Subscription button to open the Edit Subscription page.
4. In the System field, select the local system as the subscriber.
5. In the Source Type field, select Local.
6. In the Event Filter field, select the *XX*.oracle.workflow.bes.vacation.scheduled event that you defined in the Defining an Event practice.
7. Leave the Source Agent field blank.
8. Enter 40 in the Phase field.
9. In the Status field, select Enabled.
10. In the Rule Data field, select Key.
11. Enter WF_RULE.Error in the Rule Function field. This rule function returns the status code ERROR.
12. Leave the Workflow Item Type, Workflow Process Name, Out Agent, To Agent, Owner Name, and Owner Tag fields blank. Leave the Priority field set to the default value, which is Normal.
13. In the Parameters field, enter WFSQL_ARGS. This is the internal name of the Workflow error message “Invalid value(s) passed for arguments.” WF_RULE.Error will set this error message into the event message.
14. In the Description field, enter *XX* Vacation Scheduled Error Subscription.


Edit Subscription - Microsoft Internet Explorer provided by Dev. Services
File Edit View Favorites Tools Help
Edit Subscription
ORACLE

Subscriber

System	HM000A

Triggering Condition

Source Type	Local
Event Filter	XX.oracle.workflow.bes.vacation.scheduled
Source Agent	

Execution Control

Phase	40
Status	Enabled
Rule Data	Key

Action

Rule Function	WF_RULE.Error
Workflow Item Type	
Workflow Process Name	
Out Agent	
To Agent	
Priority	Normal
Parameters	WFSQL_ARGS

Documentation

Owner Name	
Owner Tag	
Description	XX Vacation Scheduled Error Subscription

15. Choose the Submit button to save the subscription.

2. Raise an event to trigger the subscription.

Solution:

16. Navigate back to the Oracle Workflow home page.

17. Choose the Raise Event link for standalone Oracle Workflow or the Raise Business Event link for Oracle Workflow embedded in Oracle E-Business Suite.

18. In the Event Name field, select *XX*.oracle.workflow.bes.vacation.scheduled.

19. In the Event Key field, enter a unique event key such as *XX*191.

20. For the purposes of this practice, since you will not perform further processing on this event, you do not need to enter any event data. Leave the Event Data field blank.

21. Choose the Submit button to raise the event.

22. Choose OK in the confirmation window.

3. Review the notification to confirm that the subscription was executed and to respond to the error by aborting subscription processing for the event.

Solution:

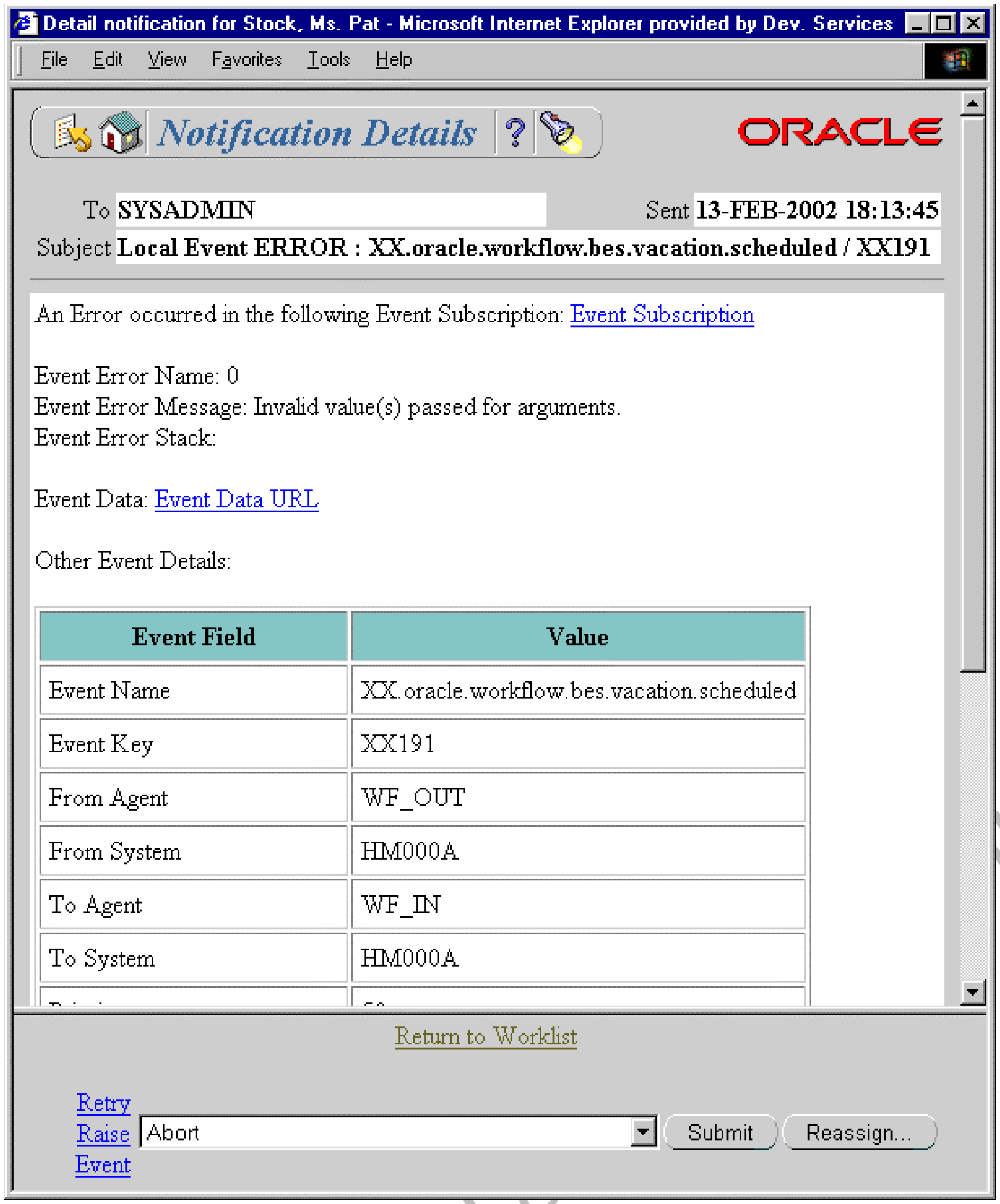
23. Navigate back to the Oracle Workflow home page.

24. Choose the Find Notifications link and search for notifications sent to the SYSADMIN role.

25. Choose the Local Event Error notification with your event name and event key in the subject.

    Note: The amount of time it takes for the notification to appear on the Worklist depends in part on the listener being run for the WF_ERROR agent.

26. Review the information in the notification. The notification should include the error message specified in the subscription parameters.


Detail notification for Stock, Ms. Pat - Microsoft Internet Explorer provided by Dev. Services
File Edit View Favorites Tools Help
Notification Details
ORACLE

To SYSADMIN Sent 13-FEB-2002 18:13:45

Subject Local Event ERROR : XX.oracle.workflow.bes.vacation.scheduled / XX191

An Error occurred in the following Event Subscription: Event Subscription

Event Error Name: 0
Event Error Message: Invalid value(s) passed for arguments.
Event Error Stack:

Event Data: Event Data URL

Other Event Details:

Event Field	Value
Event Name	XX.oracle.workflow.bes.vacation.scheduled
Event Key	XX191
From Agent	WF_OUT
From System	HM000A
To Agent	WF_IN
To System	HM000A

27. In the response region, select Abort and choose Submit.

4. Optionally raise another event to generate a new error and respond to the error by modifying the subscription definition.

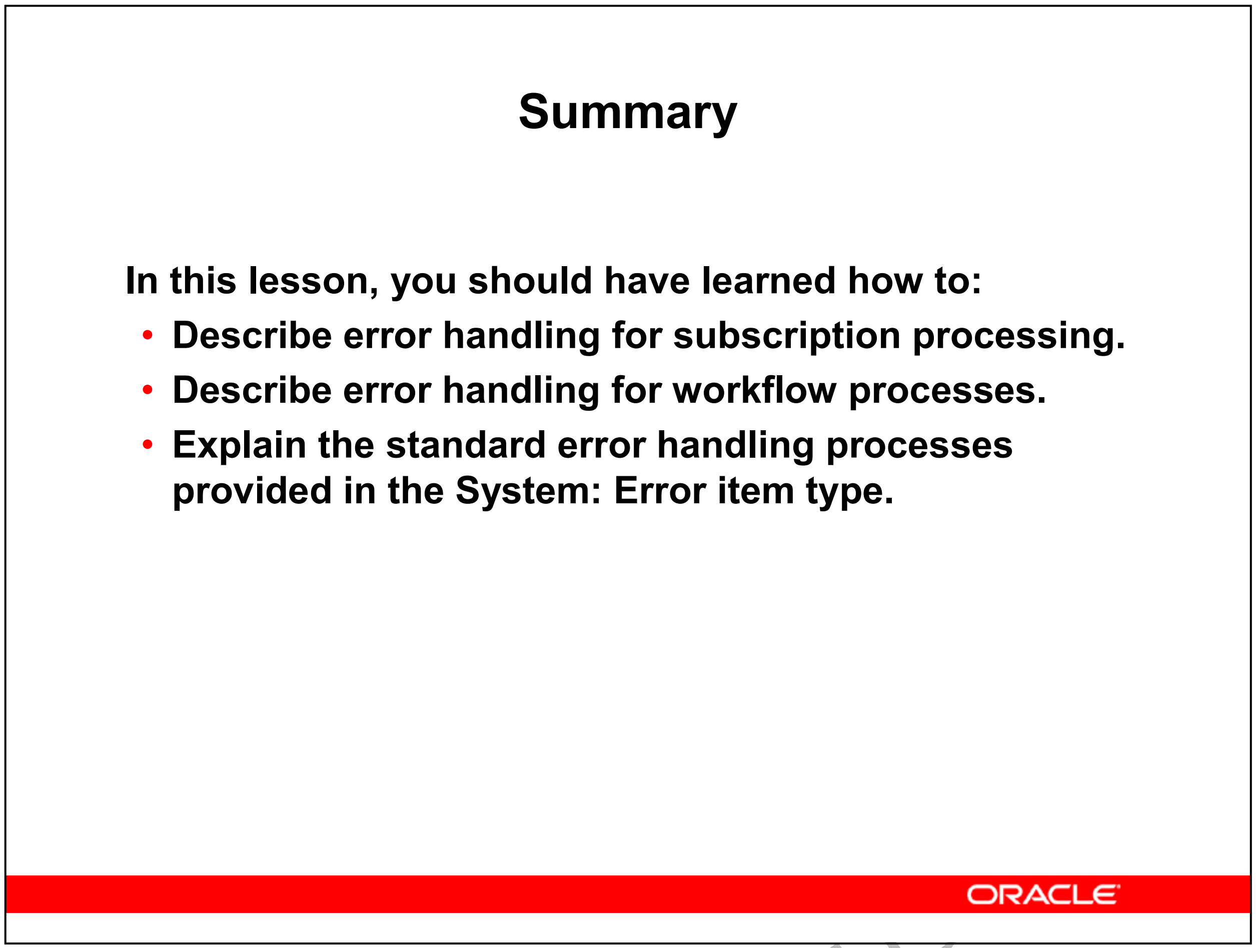
Solution:

28. Repeat Task 2 to generate a new error. Use a unique event key.

29. Review the new error notification.

30. Modify your subscription definition by changing the rule function to WF_RULE.Success. (Hint: Use the Event Subscription link in the error notification to navigate to the subscription definition.)

31. Return to the error notification and select Raise Event with Event Key as your response. This time, the subscription should complete successfully without generating an error.



Summary

In this lesson, you should have learned how to:

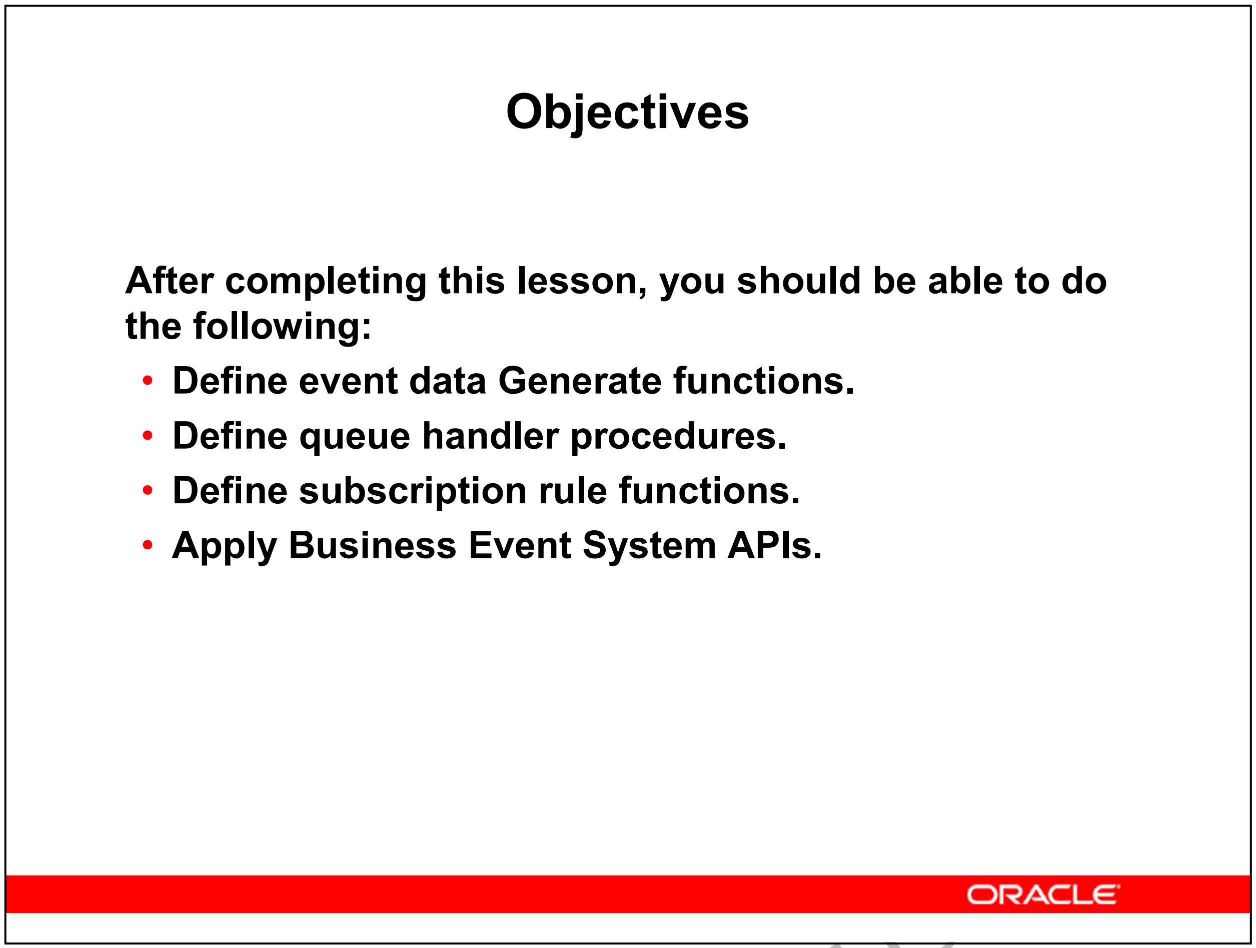
- Describe error handling for subscription processing.
- Describe error handling for workflow processes.
- Explain the standard error handling processes provided in the System: Error item type.

Business Event System APIs

Chapter 20

Business Event System APIs

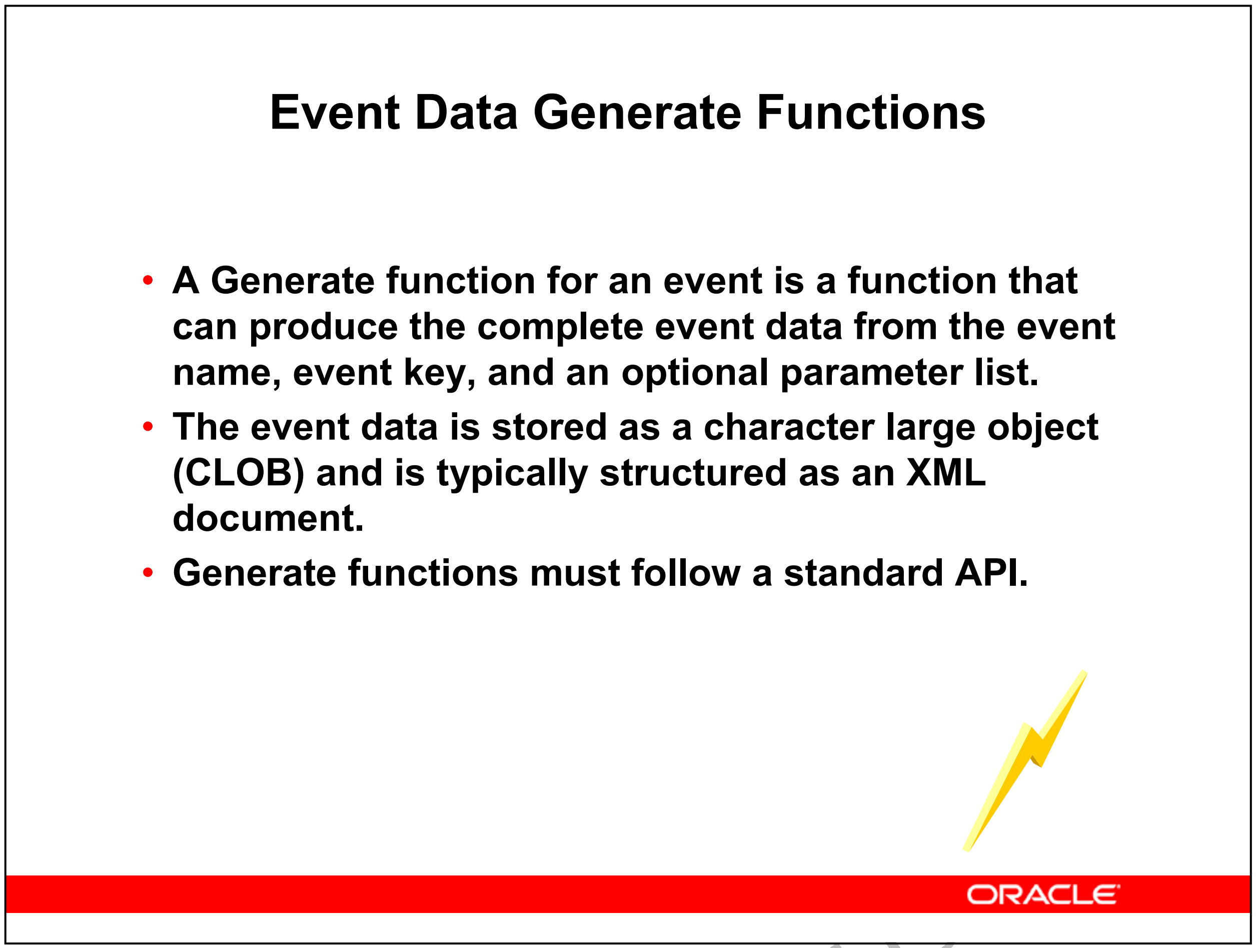
Objectives



Objectives

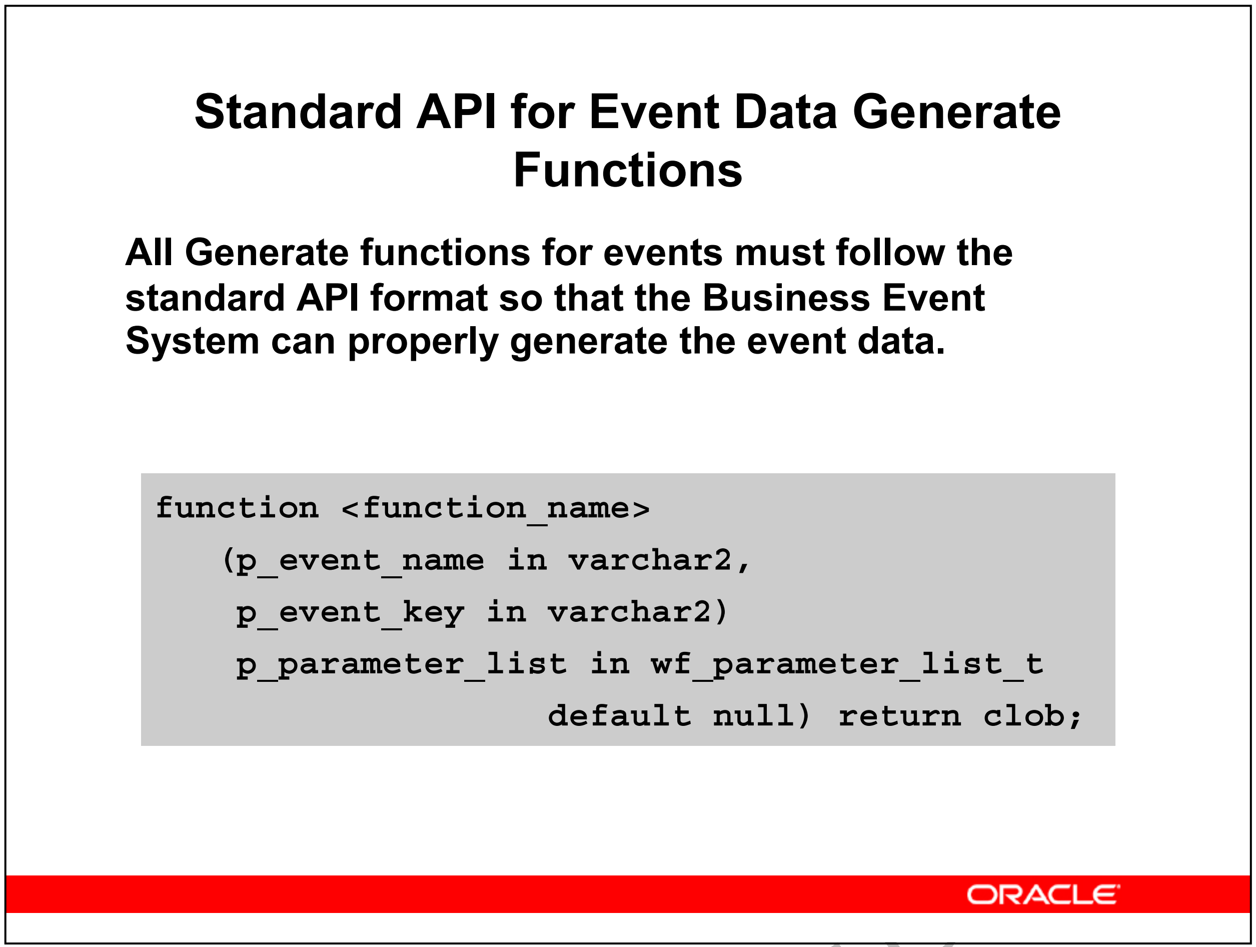
After completing this lesson, you should be able to do the following:

- **Define event data Generate functions.**
- **Define queue handler procedures.**
- **Define subscription rule functions.**
- **Apply Business Event System APIs.**



Event Data Generate Functions

- A Generate function for an event is a function that can produce the complete event data from the event name, event key, and an optional parameter list.
- The event data is stored as a character large object (CLOB) and is typically structured as an XML document.
- Generate functions must follow a standard API.



Standard API for Event Data Generate Functions

All Generate functions for events must follow the standard API format so that the Business Event System can properly generate the event data.

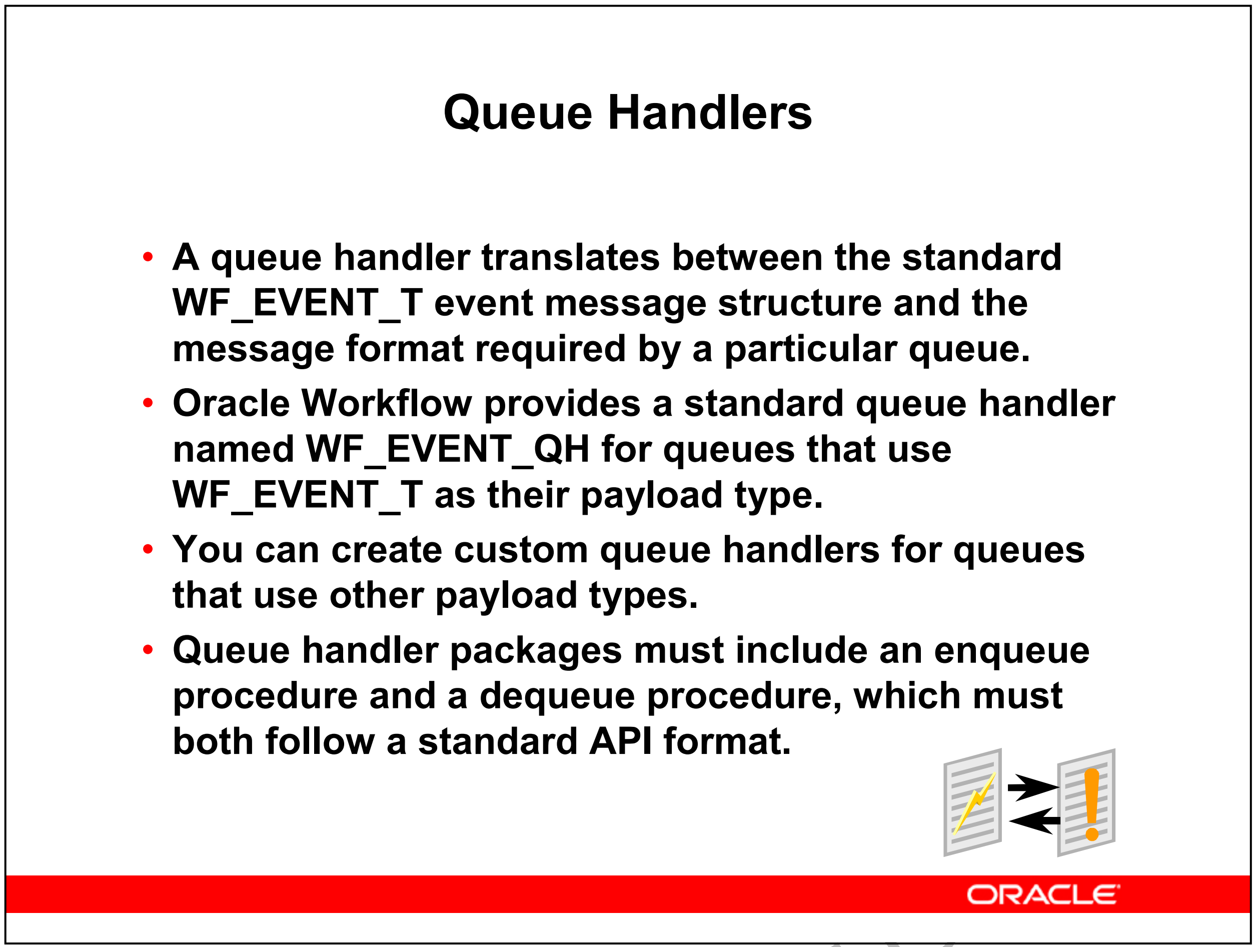
```
function <function_name>
   (p_event_name in varchar2,
    p_event_key in varchar2)
    p_parameter_list in wf_parameter_list_t
                       default null) return clob;
```

Standard API for Event Data Generate Functions

Arguments:

- p_event_name—The internal name of the event.
- p_event_key—A string generated when the event occurs within a program or application. The event key uniquely identifies a specific instance of the event.
- p_parameter_list —An optional list of additional parameter name and value pairs for the event.

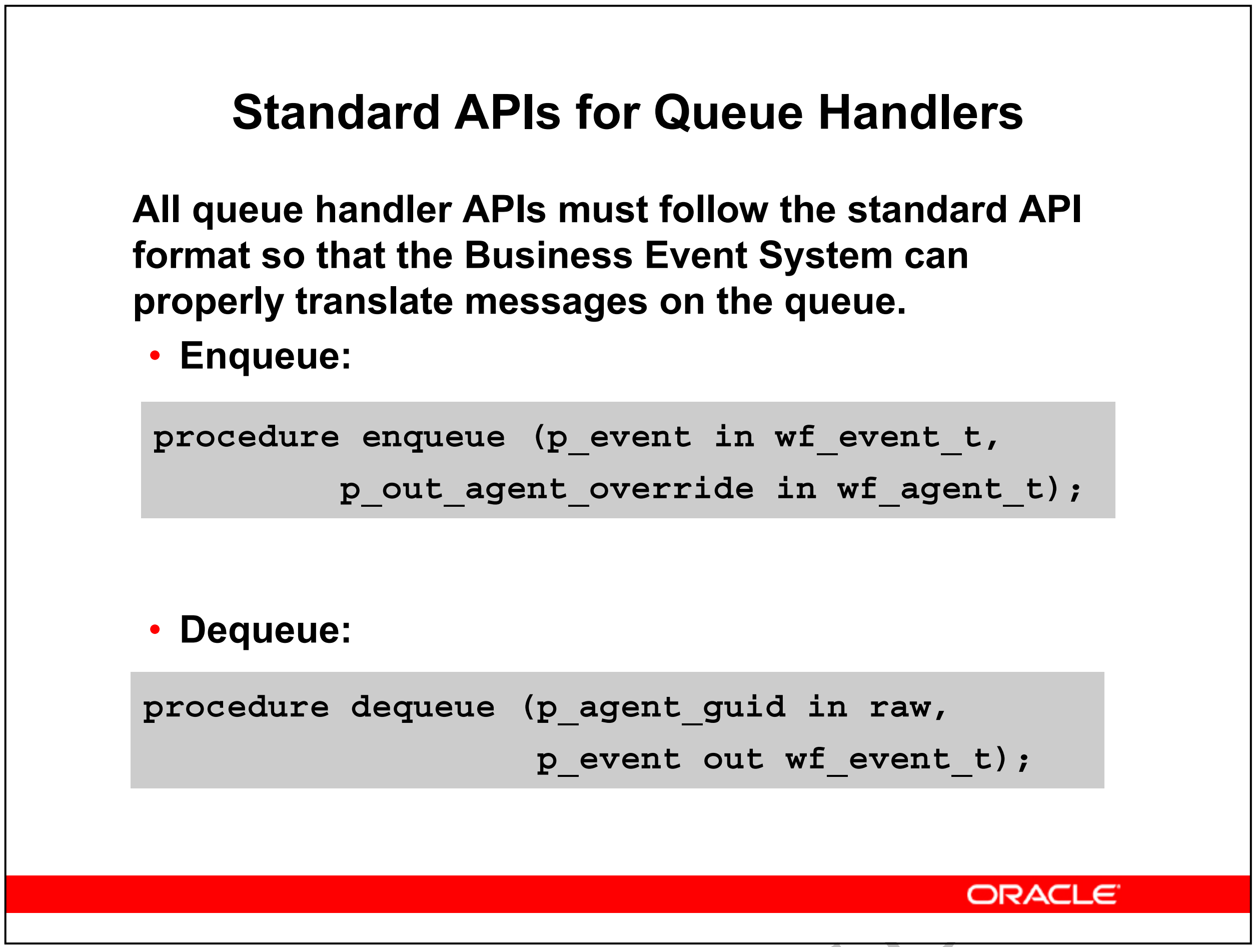
For more information, refer to the Defining Procedures and Functions for Oracle Workflow chapter in the *Oracle Workflow Guide*.



Queue Handlers

- A queue handler translates between the standard WF_EVENT_T event message structure and the message format required by a particular queue.
- Oracle Workflow provides a standard queue handler named WF_EVENT_QH for queues that use WF_EVENT_T as their payload type.
- You can create custom queue handlers for queues that use other payload types.
- Queue handler packages must include an enqueue procedure and a dequeue procedure, which must both follow a standard API format.

ORACLE



Standard APIs for Queue Handlers

All queue handler APIs must follow the standard API format so that the Business Event System can properly translate messages on the queue.

- **Enqueue:**

```
procedure enqueue (p_event in wf_event_t,
        p_out_agent_override in wf_agent_t);
```

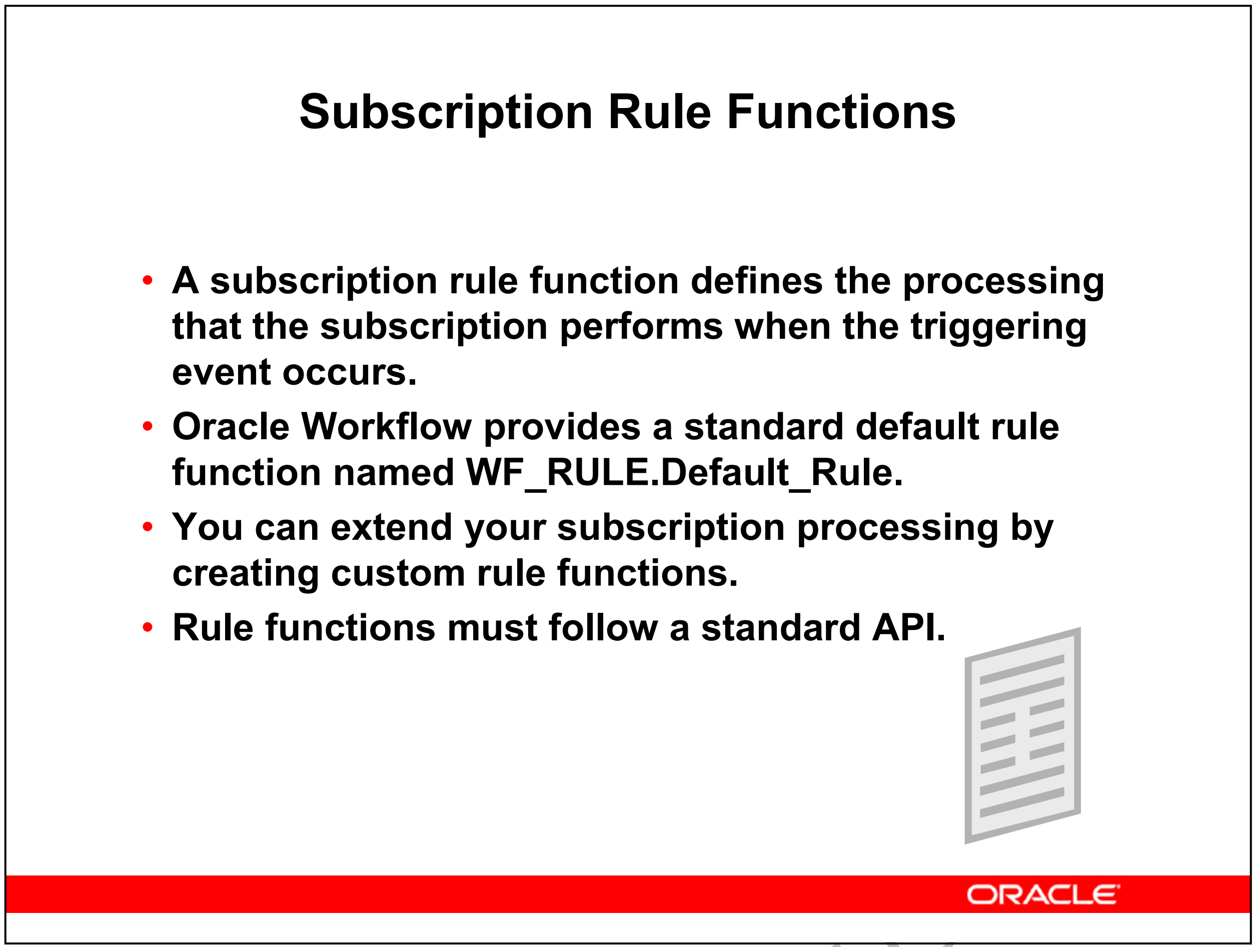
- **Dequeue:**

```
procedure dequeue (p_agent_guid in raw,
                   p_event out wf_event_t);
```

Standard API for Subscription Rule Functions

- Enqueue arguments:
  - p_event—The event message.
  - p_out_agent_ override—The outbound agent on whose queue the event message should be enqueued, overriding the outbound agent specified within the event message.
- Dequeue arguments:
  - p_agent_guid—The globally unique identifier of the inbound agent from whose queue the event message should be dequeued.
  - p_event—The event message.

For more information, refer to the Defining Procedures and Functions for Oracle Workflow chapter in the *Oracle Workflow Guide*.

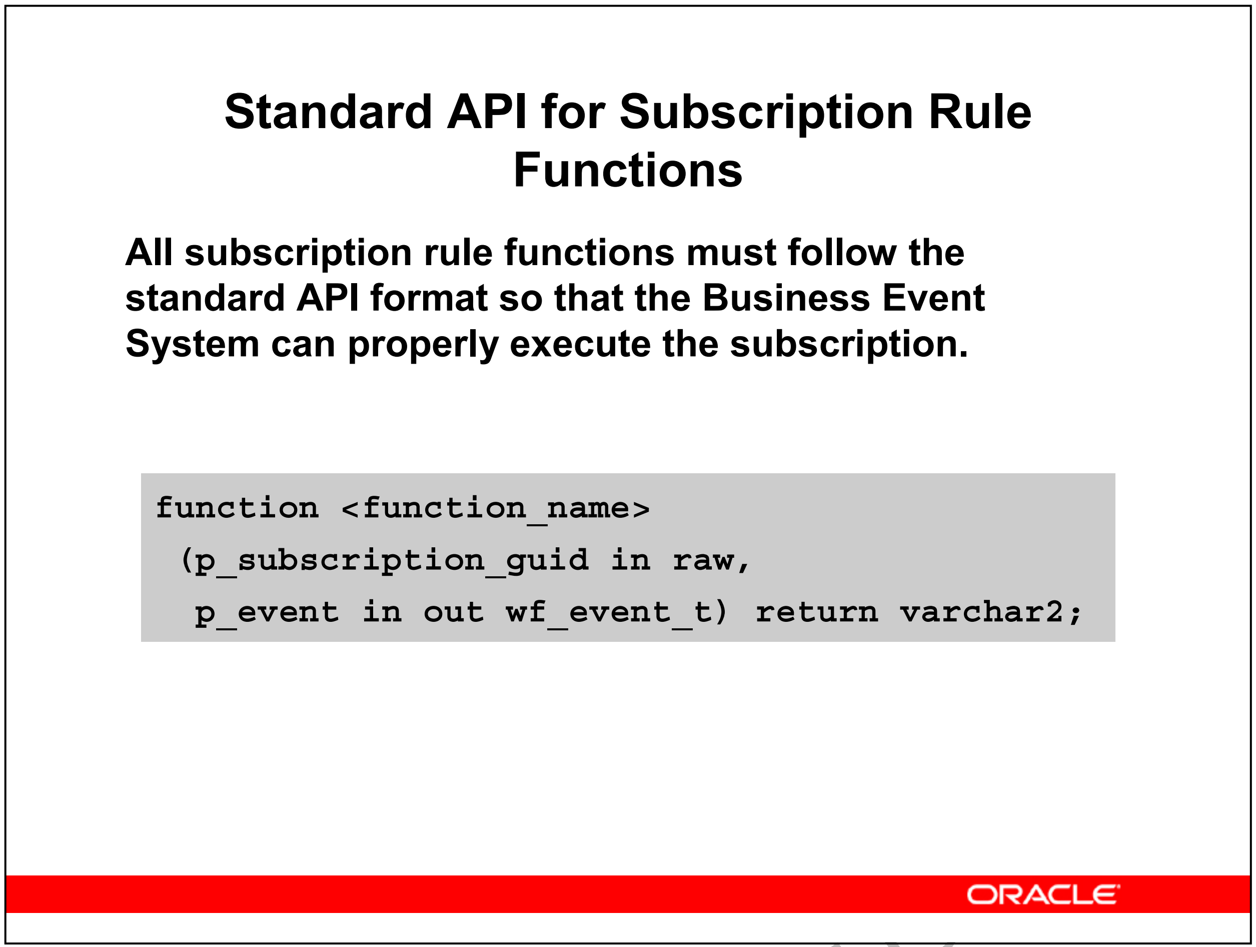


Subscription Rule Functions

- A subscription rule function defines the processing that the subscription performs when the triggering event occurs.
- Oracle Workflow provides a standard default rule function named WF_RULE.Default_Rule.
- You can extend your subscription processing by creating custom rule functions.
- Rule functions must follow a standard API.

Subscription Rule Functions

- A rule function may read or write to the event message or perform any other database action.
- However, you should never commit within a rule function. The Event Manager never issues a commit as it is the responsibility of the calling application to commit.
- A rule function must not change the connection context in any way, including security and NLS settings.



Standard API for Subscription Rule Functions

All subscription rule functions must follow the standard API format so that the Business Event System can properly execute the subscription.

```
function <function_name>
 (p_subscription_guid in raw,
  p_event in out wf_event_t) return varchar2;
```

Standard API for Subscription Rule Functions

A rule function for an event subscription must have the following standard API:

```
function <function_name>
     (p_subscription_guid in raw,
      p_event in out WF_EVENT_T) return varchar2 is

<local declarations>

begin

<your executable statements>

<optional code for WARNING>
```

```
   WF_CORE.CONTEXT('<package name>', '<function name>',
                   p_event.getEventName( ),
                   p_subscription_guid);
   WF_EVENT.setErrorInfo(p_event, 'WARNING');
   return 'WARNING';

return 'SUCCESS';

exception
  when others then
  WF_CORE.CONTEXT('<package name>', '<function name>',
                  p_event.getEventName( ),
                  p_subscription_guid);
 WF_EVENT.setErrorInfo(p_event, 'ERROR');
 return 'ERROR';

end;
```

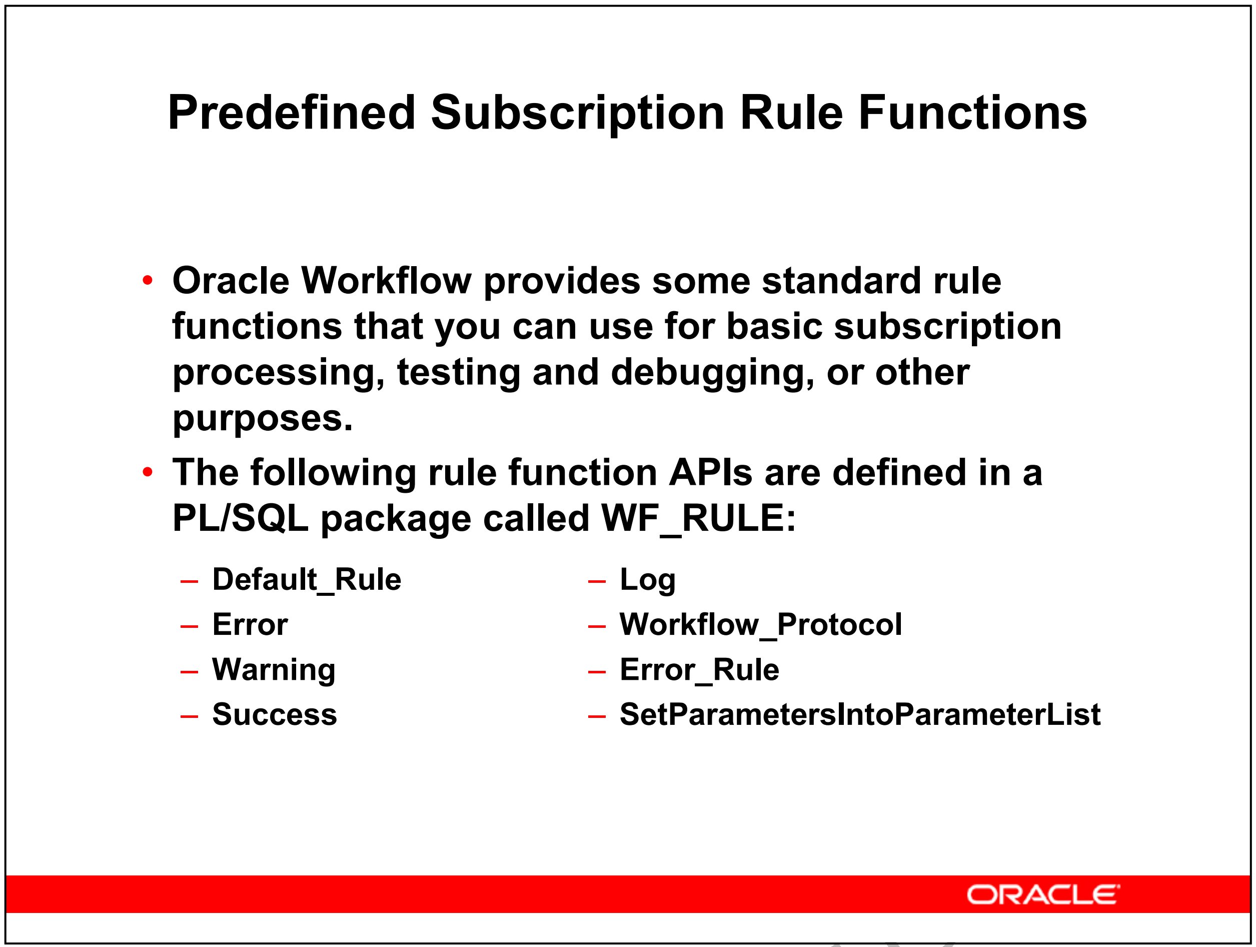
Arguments:

- p_subscription_ guid—The globally unique identifier for the subscription.
- p_event—The event message.

The function must return one of the following status codes:

- SUCCESS—The rule function completed successfully.
- WARNING—A warning condition occurred. The rule function reports a warning message using the Workflow Core error APIs and sets the warning information into the event message. The Event Manager places a copy of the event message on the WF_ERROR queue, but subscription processing continues.
- ERROR—An error occurred. The rule function reports an error message using the Workflow Core error APIs and sets the error information into the event message. The Event Manager halts subscription processing for this event, rolls back any subscriptions already executed for the event, and places the event message on the WF_ERROR queue.

For more information, refer to the Defining Procedures and Functions for Oracle Workflow chapter in the *Oracle Workflow Guide*.



Predefined Subscription Rule Functions

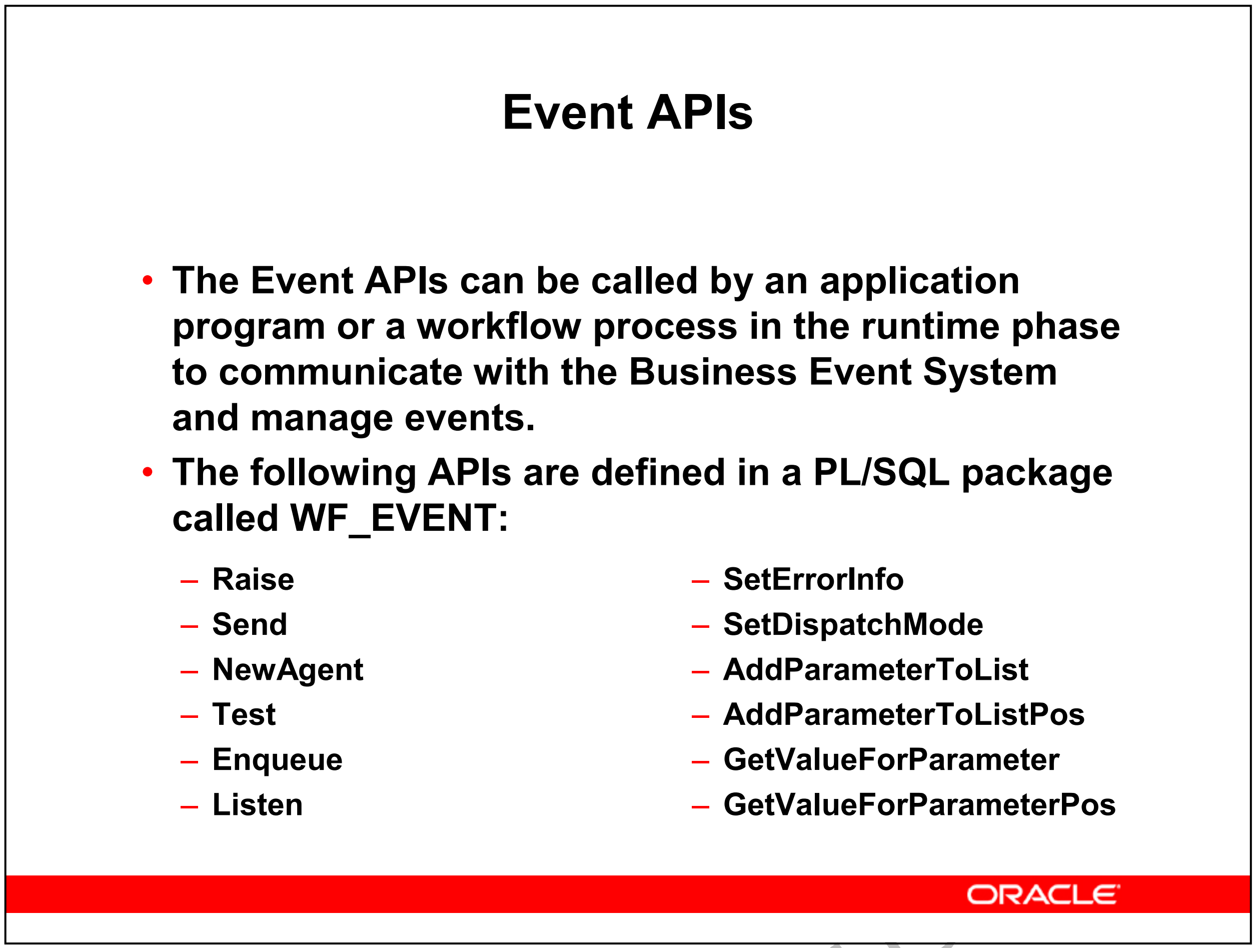
- Oracle Workflow provides some standard rule functions that you can use for basic subscription processing, testing and debugging, or other purposes.
- The following rule function APIs are defined in a PL/SQL package called WF_RULE:
  - Default_Rule
  - Error
  - Warning
  - Success
  - Log
  - Workflow_Protocol
  - Error_Rule
  - SetParametersIntoParameterList

ORACLE

Predefined Rule Functions

- Default_Rule—Performs default subscription processing on the event message when no rule function is specified for an event subscription, including:
  - Sending the event message to a workflow process, if specified in the subscription definition
  - Sending the event message to an agent, if specified in the subscription definition
- Error—Returns the status code ERROR and sets a specified error message into the event message.
- Warning—Returns the status code WARNING and sets a specified error message into the event message.
- Success—Returns the status code SUCCESS.
- Log—Outputs the contents of the event message to a SQL*Plus session using DBMS_OUTPUT.put_line.
- Workflow_Protocol—Sends the event message to the workflow process specified in the subscription, which should in turn send the event message to the inbound agent specified in the subscription.

- Error_Rule—Performs the same subscription processing as Default_Rule, but reraises any exception that is encountered.
- SetParametersIntoParameterList — Sets the parameter name and value pairs from the subscription parameters into the PARAMETER_LIST attribute of the event message, except for any parameter named ITEMKEY or CORRELATION_ID. For a parameter with one of these names, the function sets the CORRELATION_ID attribute of the event message to the parameter value.



Event APIs

- The Event APIs can be called by an application program or a workflow process in the runtime phase to communicate with the Business Event System and manage events.
- The following APIs are defined in a PL/SQL package called WF_EVENT:
  - Raise
  - Send
  - NewAgent
  - Test
  - Enqueue
  - Listen
  - SetErrorInfo
  - SetDispatchMode
  - AddParameterToList
  - AddParameterToListPos
  - GetValueForParameter
  - GetValueForParameterPos

ORACLE

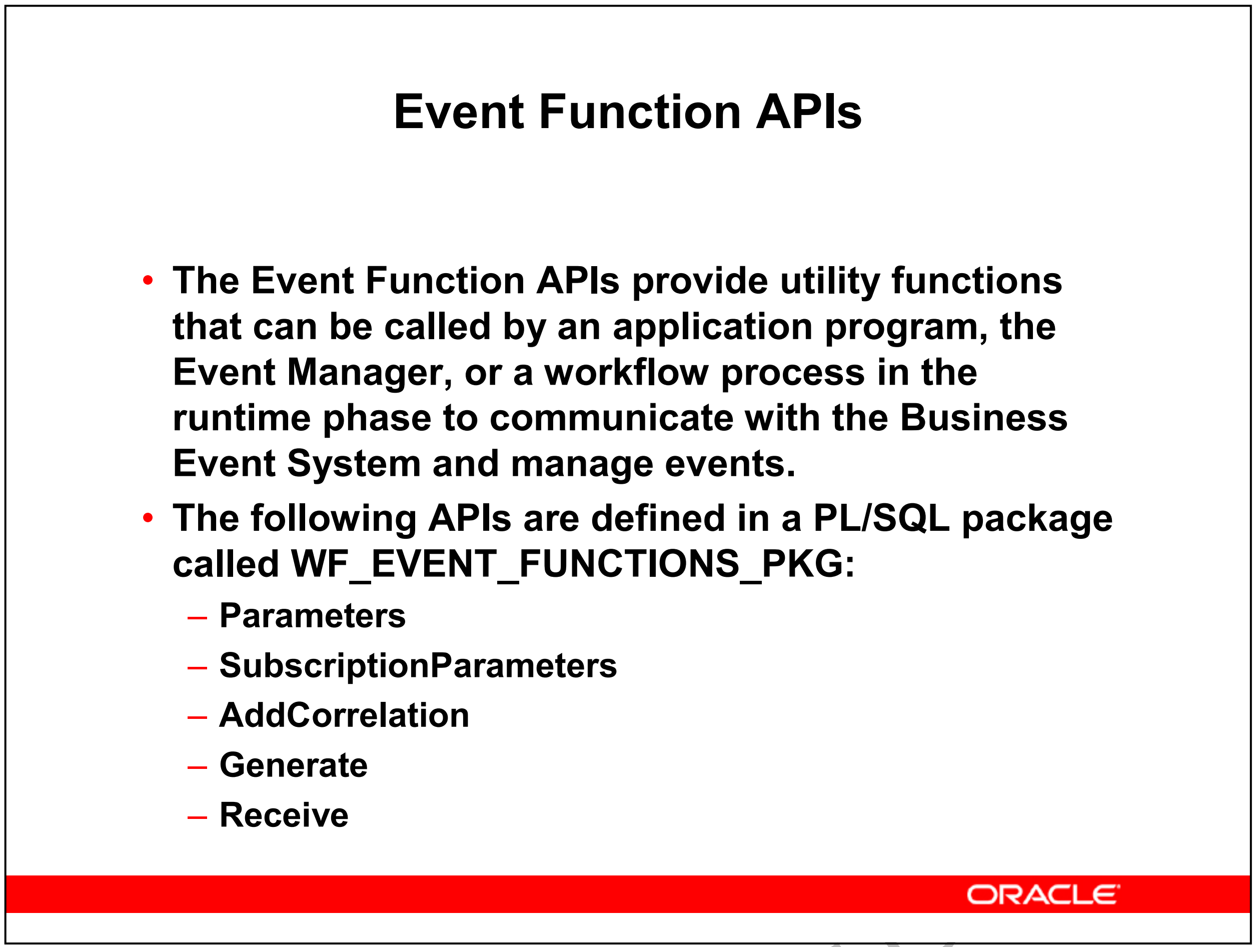
Event APIs

- Raise—Raises a local event to the Event Manager.
- Send—Sends an event message from one agent to another.
- NewAgent—Creates a WF_AGENT_T structure for the specified agent and sets the agent's system and name into the structure.
- Test—Tests for the most costly data requirement among subscriptions to an event.
- Enqueue—Enqueues an event message onto a queue associated with an outbound agent.
- Listen—Monitors an agent for inbound event messages and dequeues messages using the agent's queue handler.

  Note: If you are using the version of Oracle Workflow embedded in Oracle E-Business Suite, you can also use the "Workflow Agent Listener" concurrent program to listen for inbound event messages.
- SetErrorInfo—Retrieves error information from the error stack and sets it into the event message.
- SetDispatchMode—Sets the dispatch mode of the Event Manager to either deferred or synchronous subscription processing.

- AddParameterToList—Adds the specified parameter name and value pair to the end of the specified parameter list varray.
- AddParameterToListPos—Adds the specified parameter name and value pair to the end of the specified parameter list varray and returns the index for the position at which the parameter is stored within the varray.
- GetValueForParameter—Retrieves the value of the specified parameter from the specified parameter list varray.
- GetValueForParameterPos—Retrieves the value of the parameter stored at the specified position in the specified parameter list varray.

For more information, refer to the Oracle Workflow APIs chapter in the *Oracle Workflow Guide*.



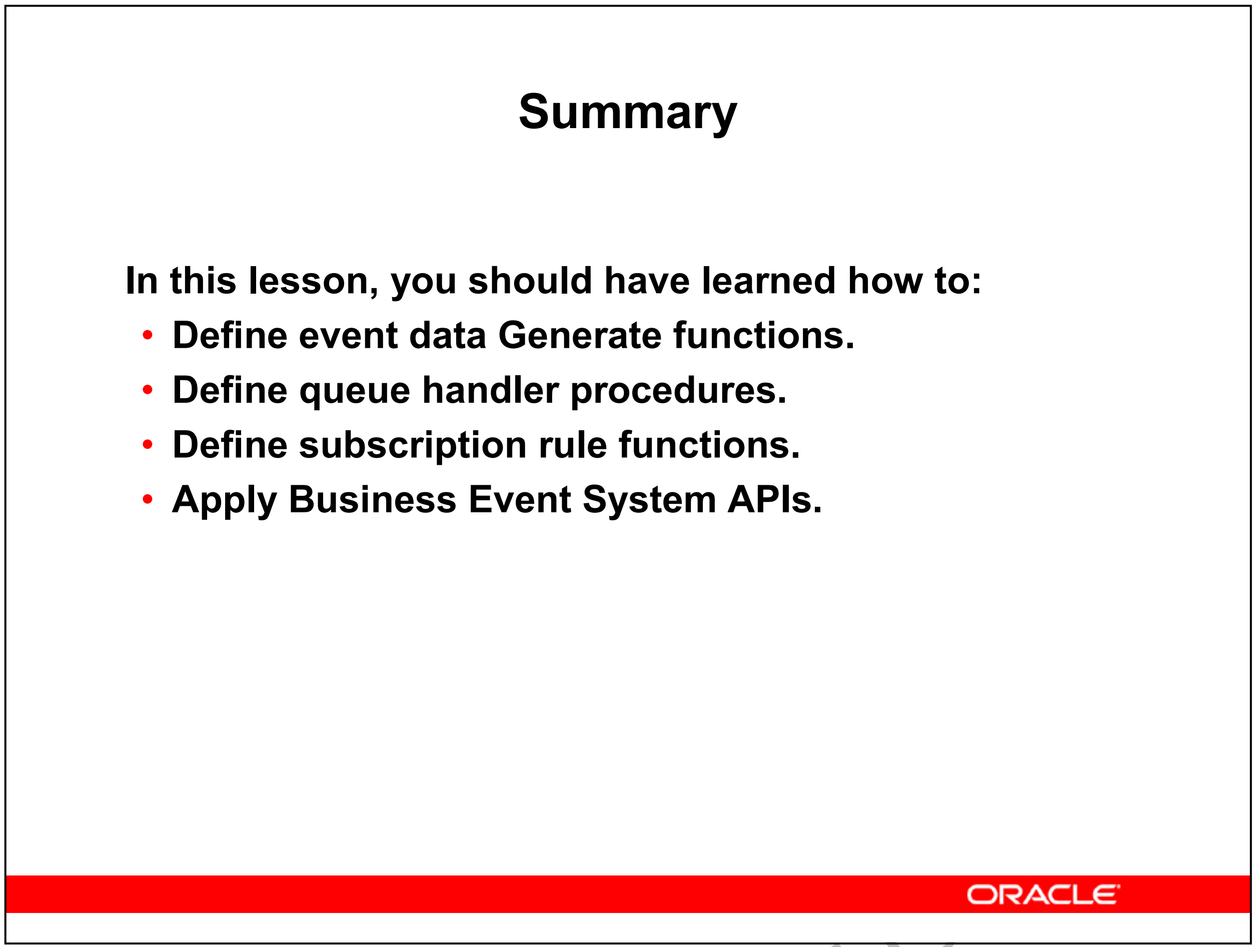
Event Function APIs

- The Event Function APIs provide utility functions that can be called by an application program, the Event Manager, or a workflow process in the runtime phase to communicate with the Business Event System and manage events.
- The following APIs are defined in a PL/SQL package called WF_EVENT_FUNCTIONS_PKG:
  - Parameters
  - SubscriptionParameters
  - AddCorrelation
  - Generate
  - Receive

Event Function APIs

- Parameters—Parses a string of text containing parameters and returns the parsed parameters in a varray.
- SubscriptionParameters—Returns the value for a parameter from a text string containing the parameters defined for an event subscription.
- AddCorrelation—Adds a correlation ID to an event message when used as a rule function for subscription processing.
- Generate—Generates the event data for events in the Seed Event Group.
- Receive—Receives Business Event System object definitions when used as a rule function for subscription processing and loads the definitions into the appropriate Business Event System tables.

For more information, refer to the Oracle Workflow APIs chapter in the *Oracle Workflow Guide*.



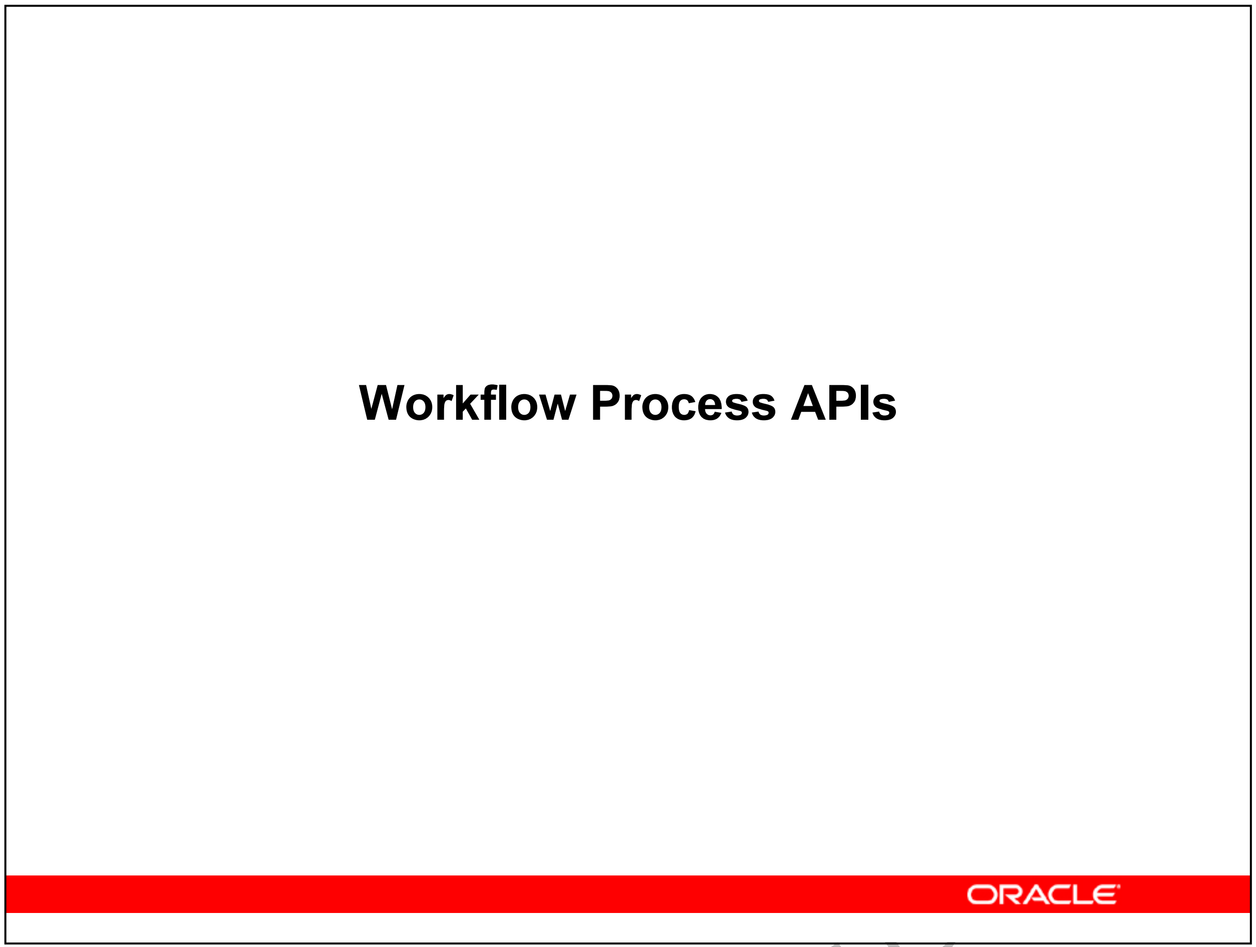
Summary

In this lesson, you should have learned how to:

- Define event data Generate functions.
- Define queue handler procedures.
- Define subscription rule functions.
- Apply Business Event System APIs.

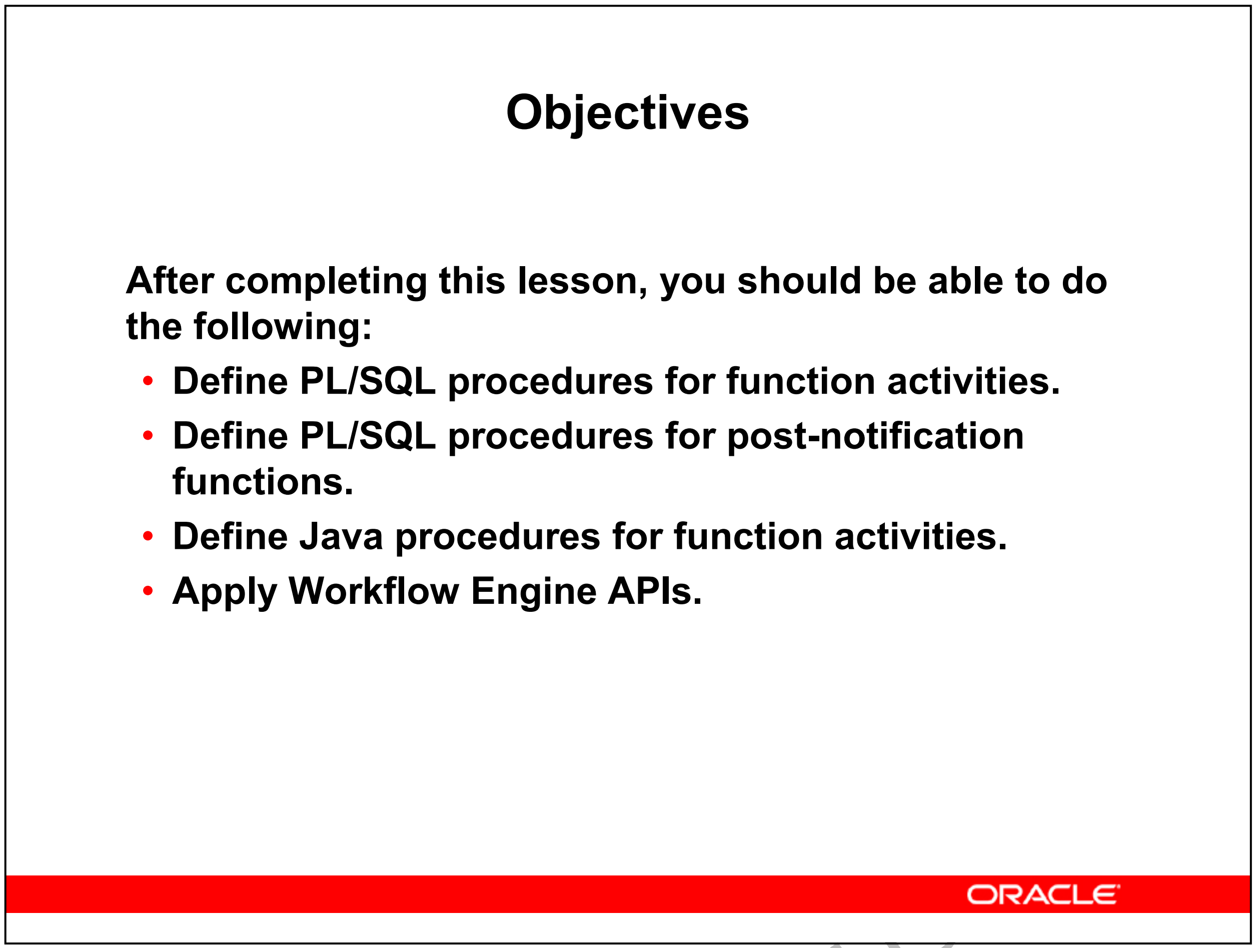
Workflow Process APIs

Chapter 21



Workflow Process APIs

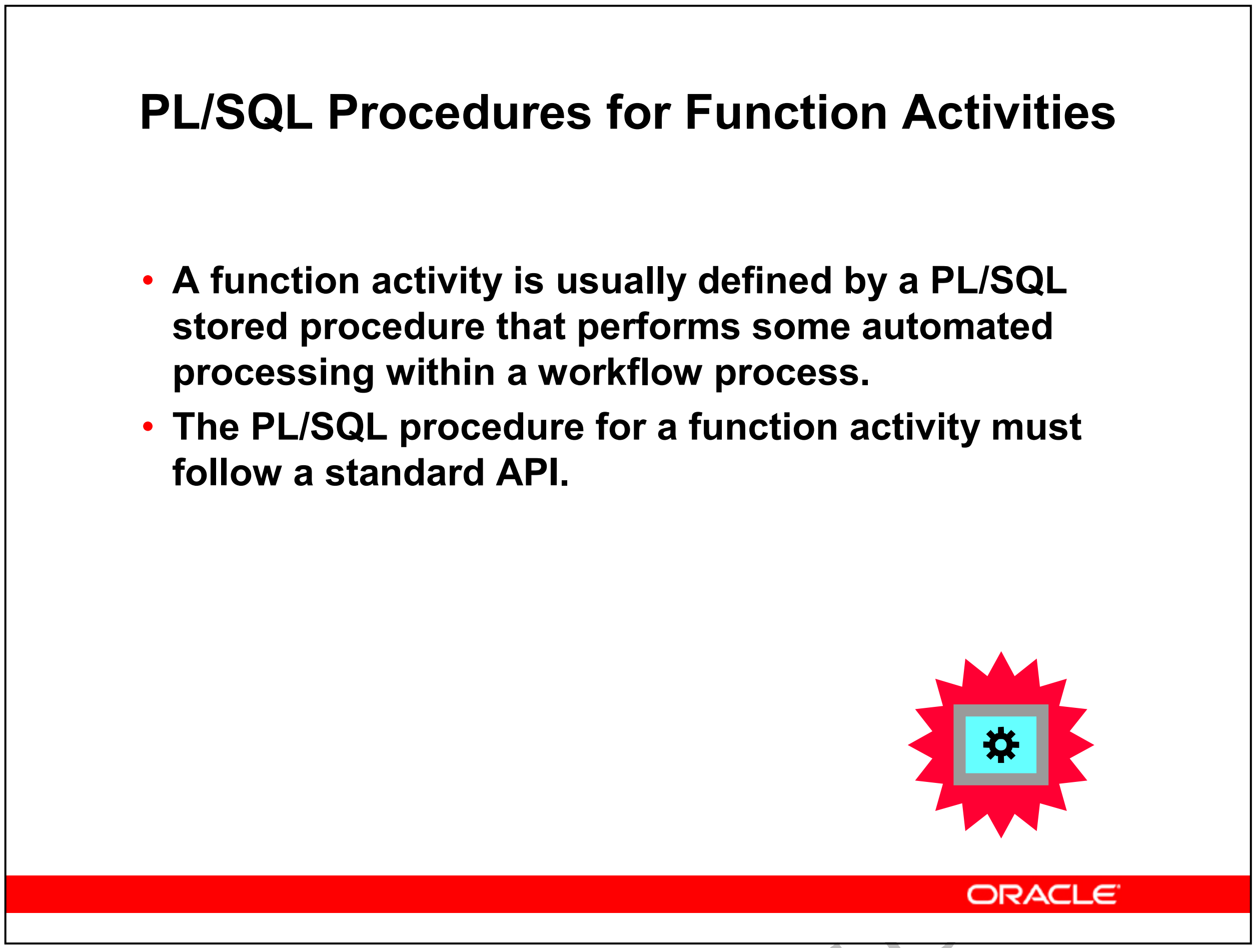
Objectives



Objectives

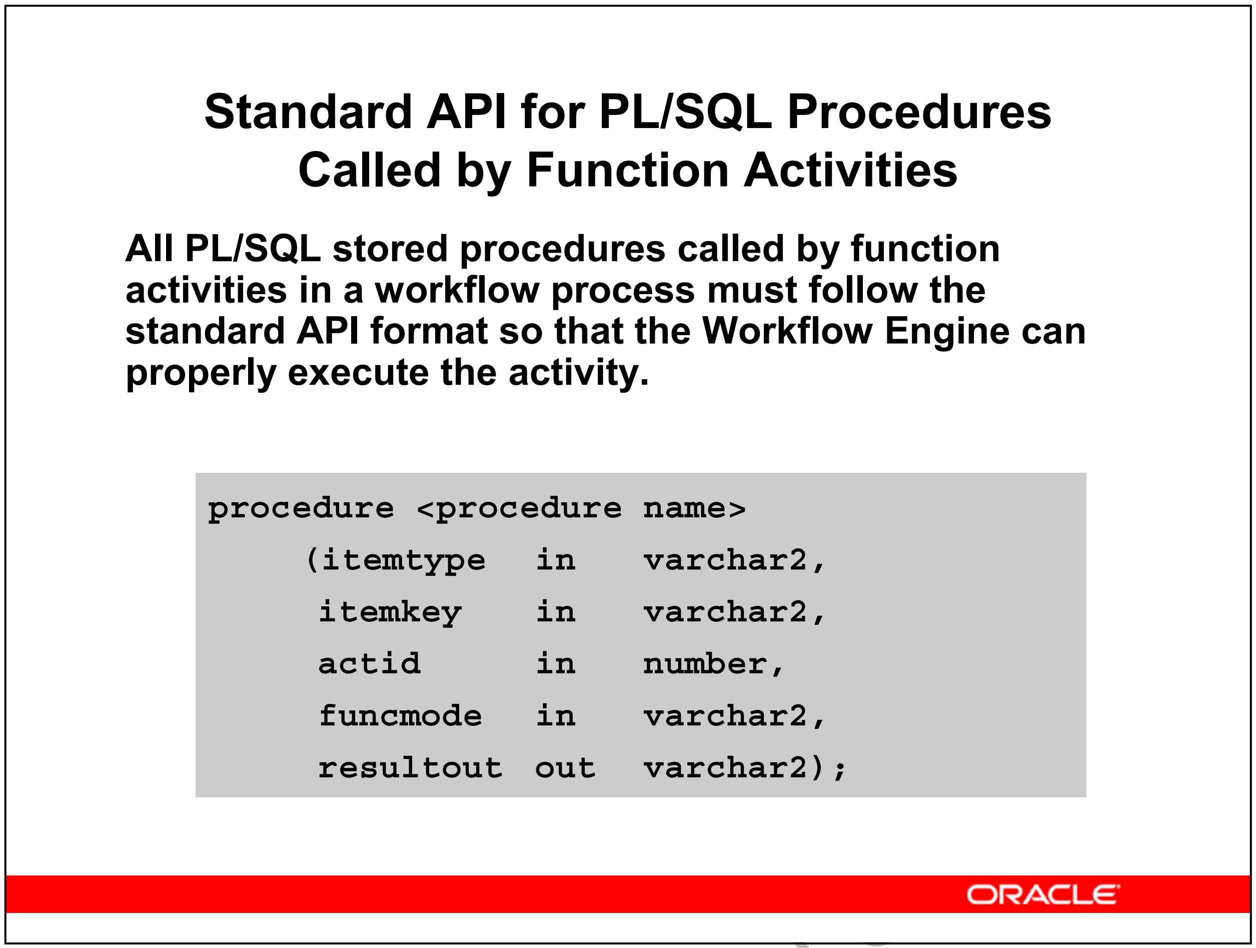
After completing this lesson, you should be able to do the following:

- **Define PL/SQL procedures for function activities.**
- **Define PL/SQL procedures for post-notification functions.**
- **Define Java procedures for function activities.**
- **Apply Workflow Engine APIs.**



PL/SQL Procedures for Function Activities

- A function activity is usually defined by a PL/SQL stored procedure that performs some automated processing within a workflow process.
- The PL/SQL procedure for a function activity must follow a standard API.



Standard API for PL/SQL Procedures Called by Function Activities

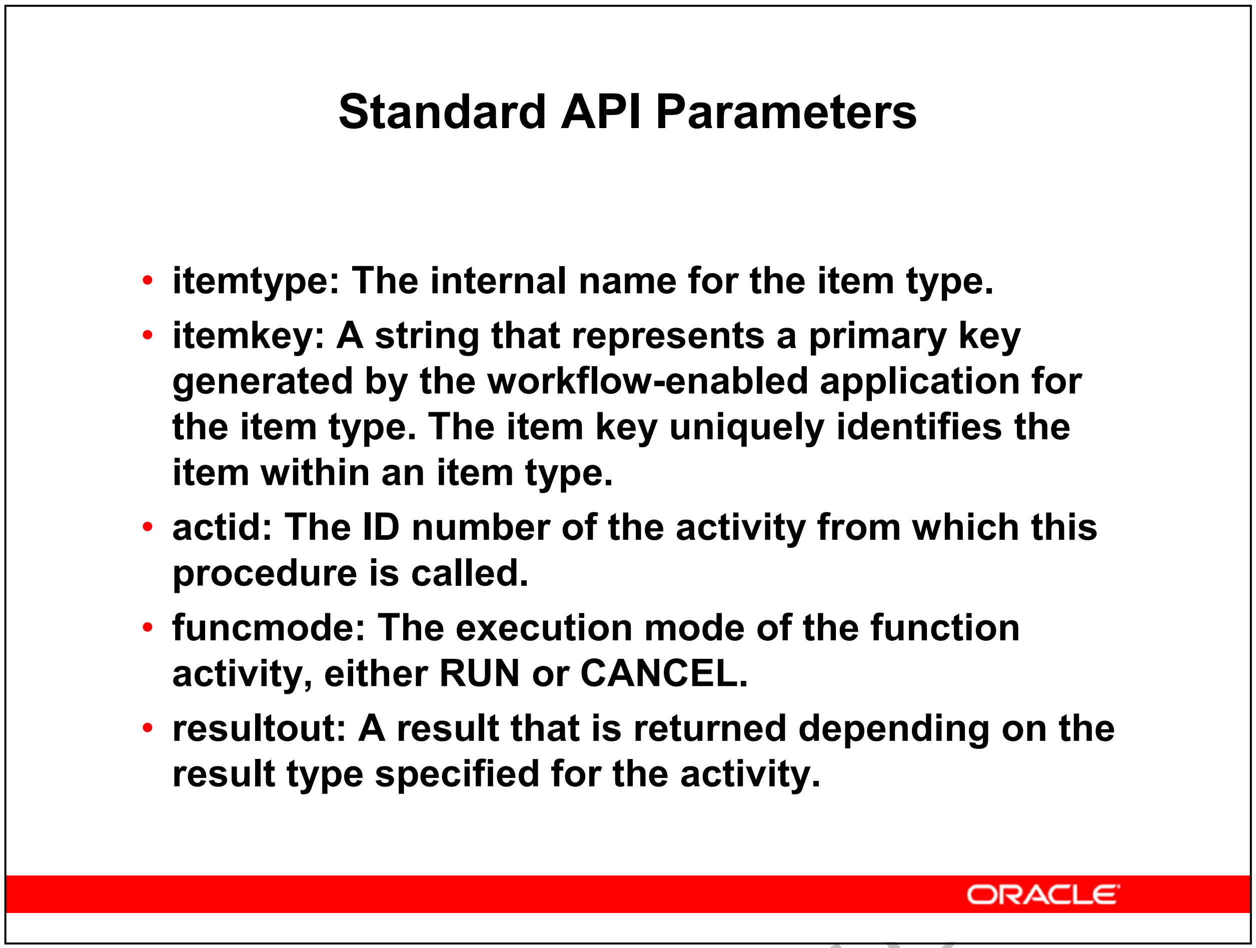
All PL/SQL stored procedures called by function activities in a workflow process must follow the standard API format so that the Workflow Engine can properly execute the activity.

```
procedure <procedure name>
     (itemtype   in    varchar2,
      itemkey    in    varchar2,
      actid      in    number,
      funcmode   in    varchar2,
      resultout  out   varchar2);
```

ORACLE

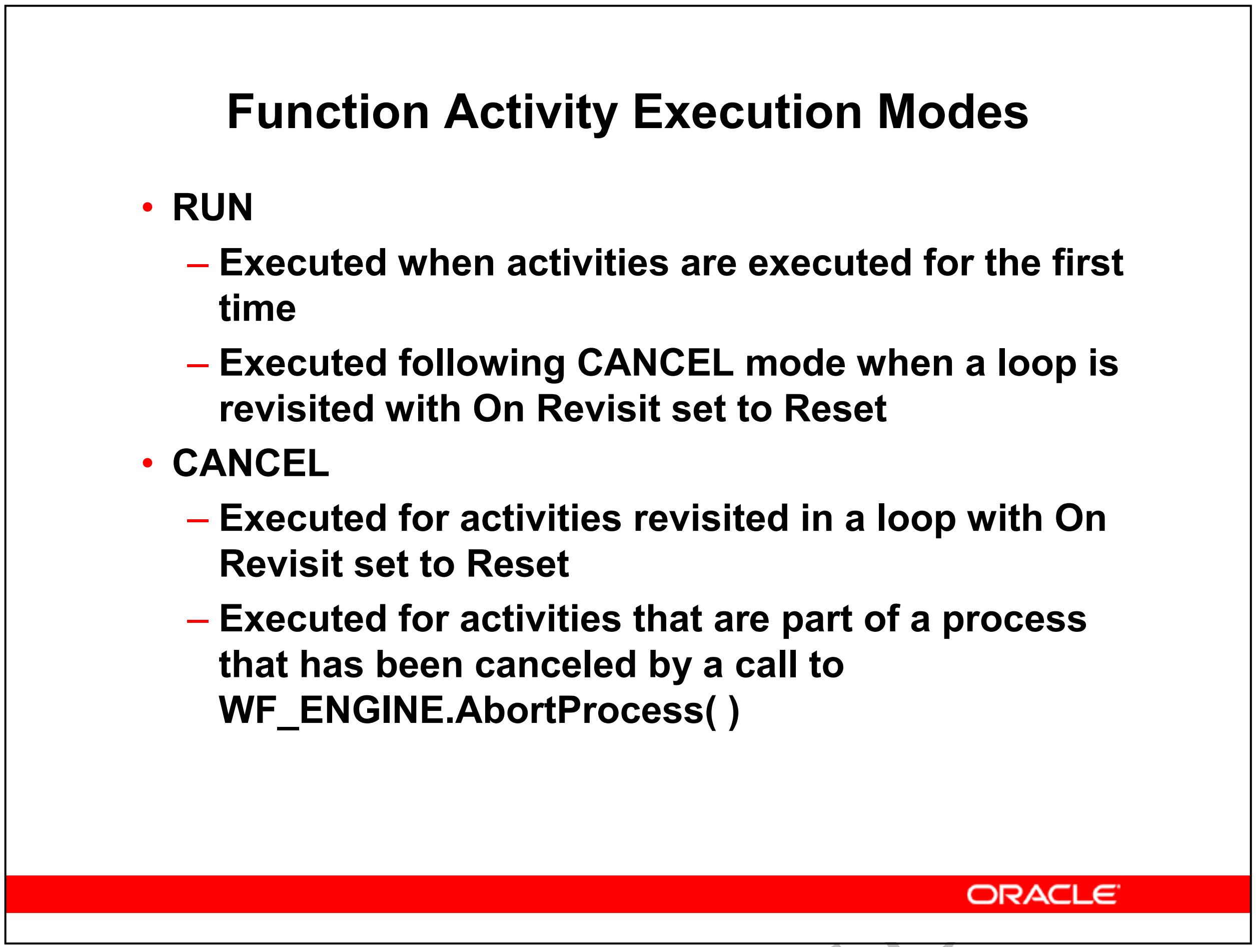
Standard API for PL/SQL Procedures Called by Function Activities

```
procedure <procedure name> (
     itemtype  in varchar2,
     itemkey   in varchar2,
     actid     in number,
     funcmode  in varchar2,
     resultout out varchar2) is
<local declarations>

begin
if (funcmode='RUN') then
     <your Run executable statements>
     resultout:='COMPLETE:<result>';
     return;
endif;
if (funcmode='CANCEL') then
     <your CANCEL executable statements>
     resultout:='COMPLETE';
     return;
endif;
...
exception
  when others then
  WF_CORE.CONTEXT('<package name>', '<procedure name>', <itemtype>,
                   <itemkey>, to_char(<actid>), <funcmode>);
  raise;
end <procedure name>;
```



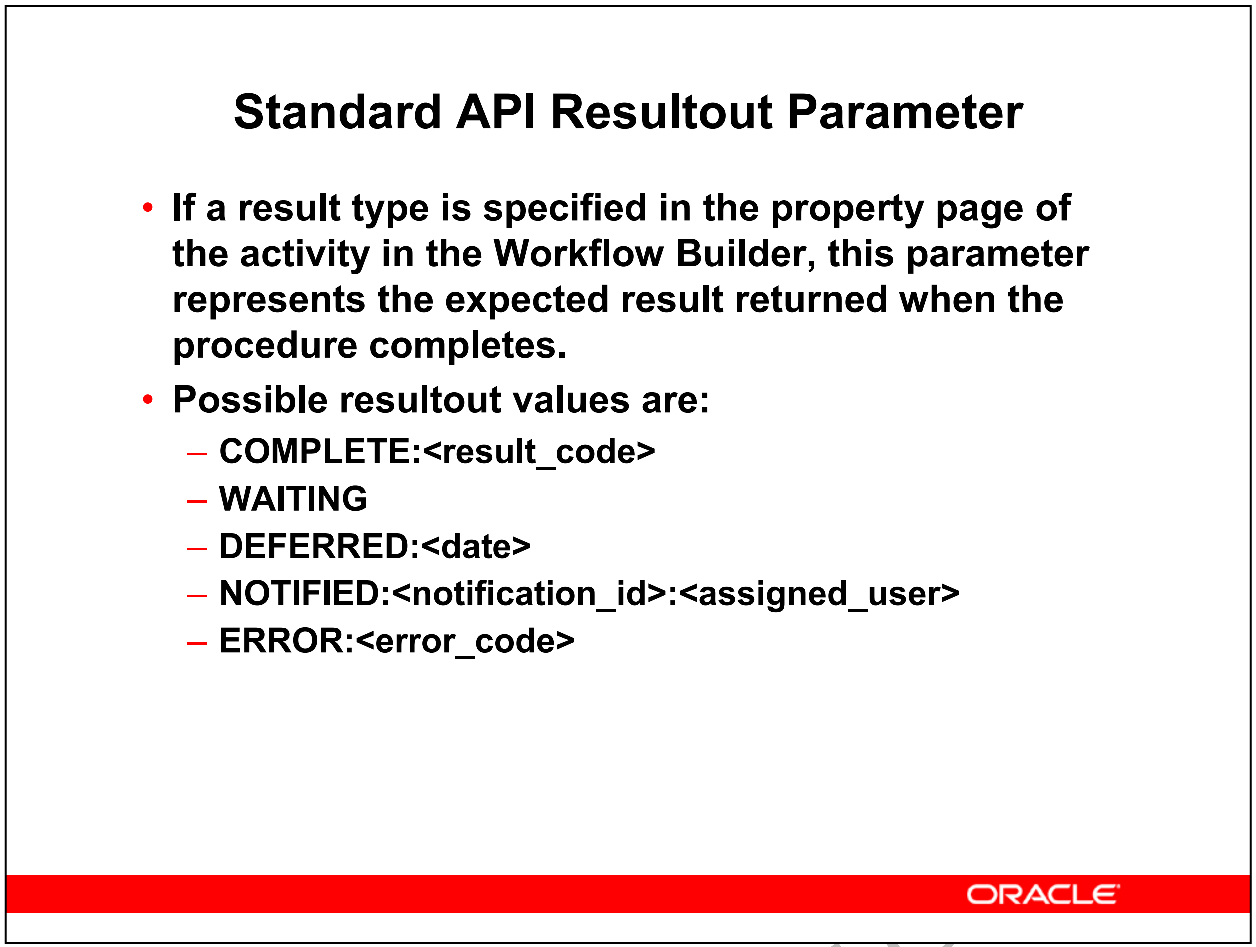
Standard API Parameters

- itemtype: The internal name for the item type.
- itemkey: A string that represents a primary key generated by the workflow-enabled application for the item type. The item key uniquely identifies the item within an item type.
- actid: The ID number of the activity from which this procedure is called.
- funcmode: The execution mode of the function activity, either RUN or CANCEL.
- resultout: A result that is returned depending on the result type specified for the activity.



Function Activity Execution Modes

- RUN
  - Executed when activities are executed for the first time
  - Executed following CANCEL mode when a loop is revisited with On Revisit set to Reset
- CANCEL
  - Executed for activities revisited in a loop with On Revisit set to Reset
  - Executed for activities that are part of a process that has been canceled by a call to WF_ENGINE.AbortProcess( )



Standard API Resultout Parameter

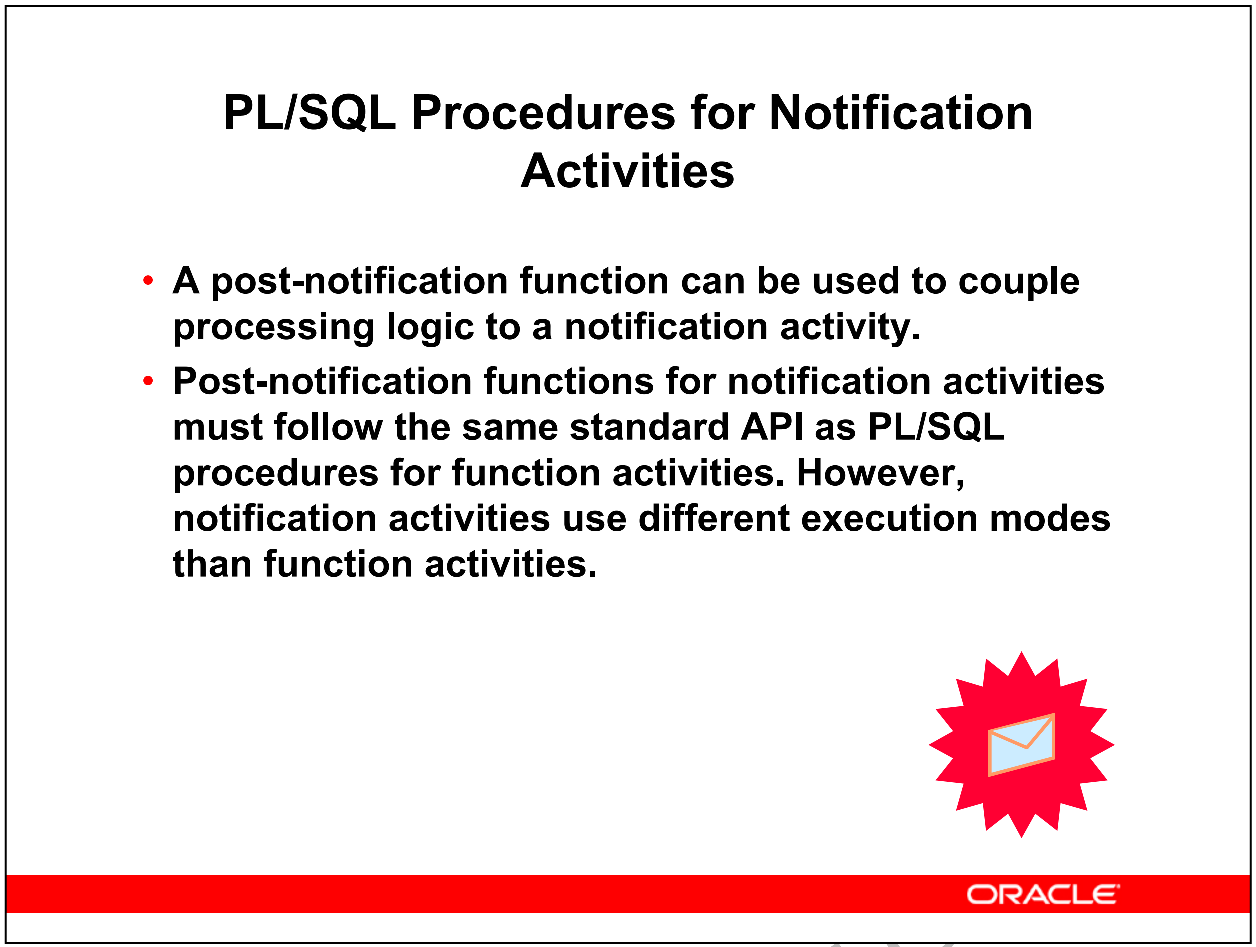
- If a result type is specified in the property page of the activity in the Workflow Builder, this parameter represents the expected result returned when the procedure completes.
- Possible resultout values are:
  - COMPLETE:<result_code>
  - WAITING
  - DEFERRED:<date>
  - NOTIFIED:<notification_id>:<assigned_user>
  - ERROR:<error_code>

Standard API Resultout Parameter

- **COMPLETE:<result_code>** The activity completes with the specified result code. The result code must match one of the result codes defined in the result type of the activity.
- **WAITING** The activity is pending, waiting for another activity to complete before it completes. An example is the Standard 'AND' activity.
- **DEFERRED:<date>** The activity is deferred to a background engine for execution until a given date. <date> must be of the format:

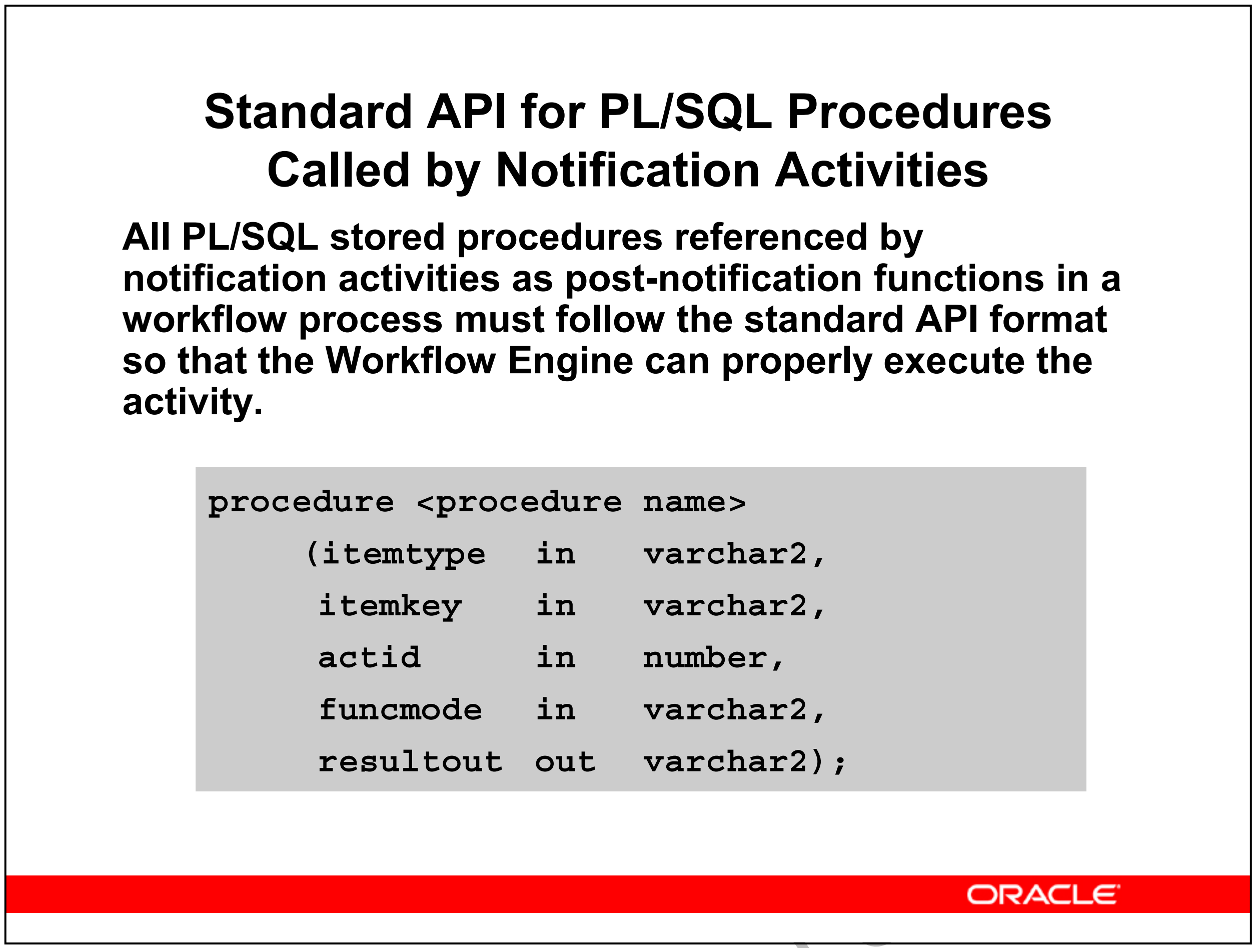
```
to_char(<date_string>, wf_engine.date_format)
```

- **NOTIFIED:<notification_id>:<assigned_user>** An external entity has been notified that an action must be performed. A notification ID and an assigned user can optionally be returned with this result. Note that the external entity must call CompleteActivity( ) to inform the Workflow Engine when the action completes.
- **ERROR:<error_code>** The activity encounters an error and returns the indicated error code.



PL/SQL Procedures for Notification Activities

- A post-notification function can be used to couple processing logic to a notification activity.
- Post-notification functions for notification activities must follow the same standard API as PL/SQL procedures for function activities. However, notification activities use different execution modes than function activities.



Standard API for PL/SQL Procedures Called by Notification Activities

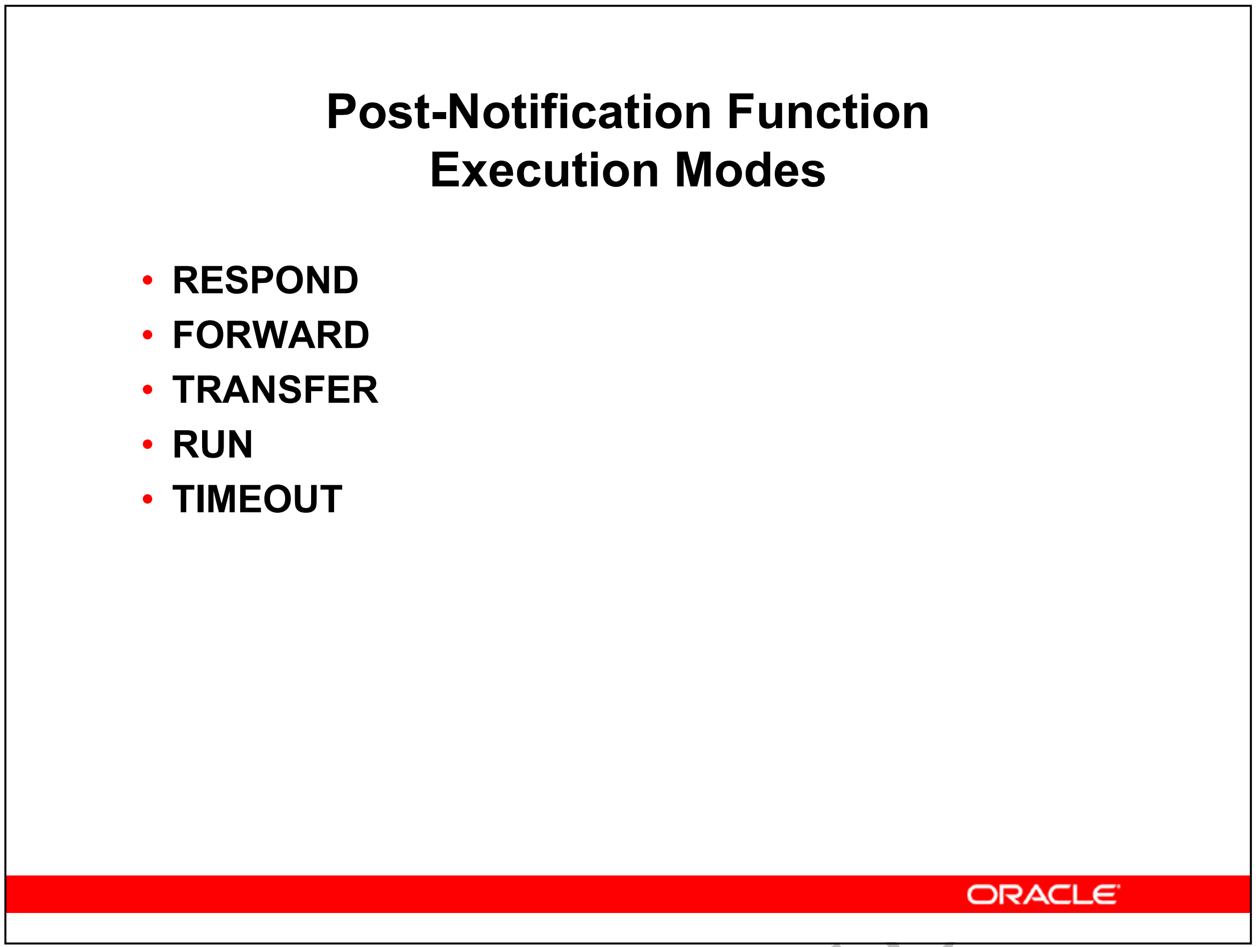
All PL/SQL stored procedures referenced by notification activities as post-notification functions in a workflow process must follow the standard API format so that the Workflow Engine can properly execute the activity.

```
procedure <procedure name>
     (itemtype   in    varchar2,
      itemkey    in    varchar2,
      actid      in    number,
      funcmode   in    varchar2,
      resultout  out   varchar2);
```

Standard API for PL/SQL Procedures Called by Notification Activities

```
procedure <procedure name> (
     itemtype    in varchar2,
     itemkey     in varchar2,
     actid       in number,
     funcmode    in varchar2,
     resultout   out varchar2) is
<local declarations>

begin
if (funcmode=‘RESPOND’) then
     <your RESPOND executable statements>
     resultout:=‘COMPLETE’;
     return;
endif;
if (funcmode=‘FORWARD’) then
     <your FORWARD executable statements>
     resultout:=‘COMPLETE’;
```

```
      return;
endif;
if (funcmode=‘TRANSFER’) then
      <your TRANSFER executable statements>
      resultout:=‘COMPLETE’;
      return;
endif;
if (funcmode=‘RUN’) then
      <your RUN executable statements>
      resultout:=‘COMPLETE:<result>‘;
      return;
endif;
if (funcmode=‘TIMEOUT’) then
      <your TIMEOUT executable statements>
      if (<condition_ok_to_proceed>) then
        resultout:=‘COMPLETE’;
      else
        resultout := wf_engine.eng_timedout;
      end if;
      return;
endif;
...
exception
  when others then
  WF_CORE.CONTEXT(’<package name>’, ’<procedure name>’, <itemtype>,
                  <itemkey>, to_char(<actid>), <funcmode>);
  raise;
end <procedure name>;
```



Post-Notification Function Execution Modes

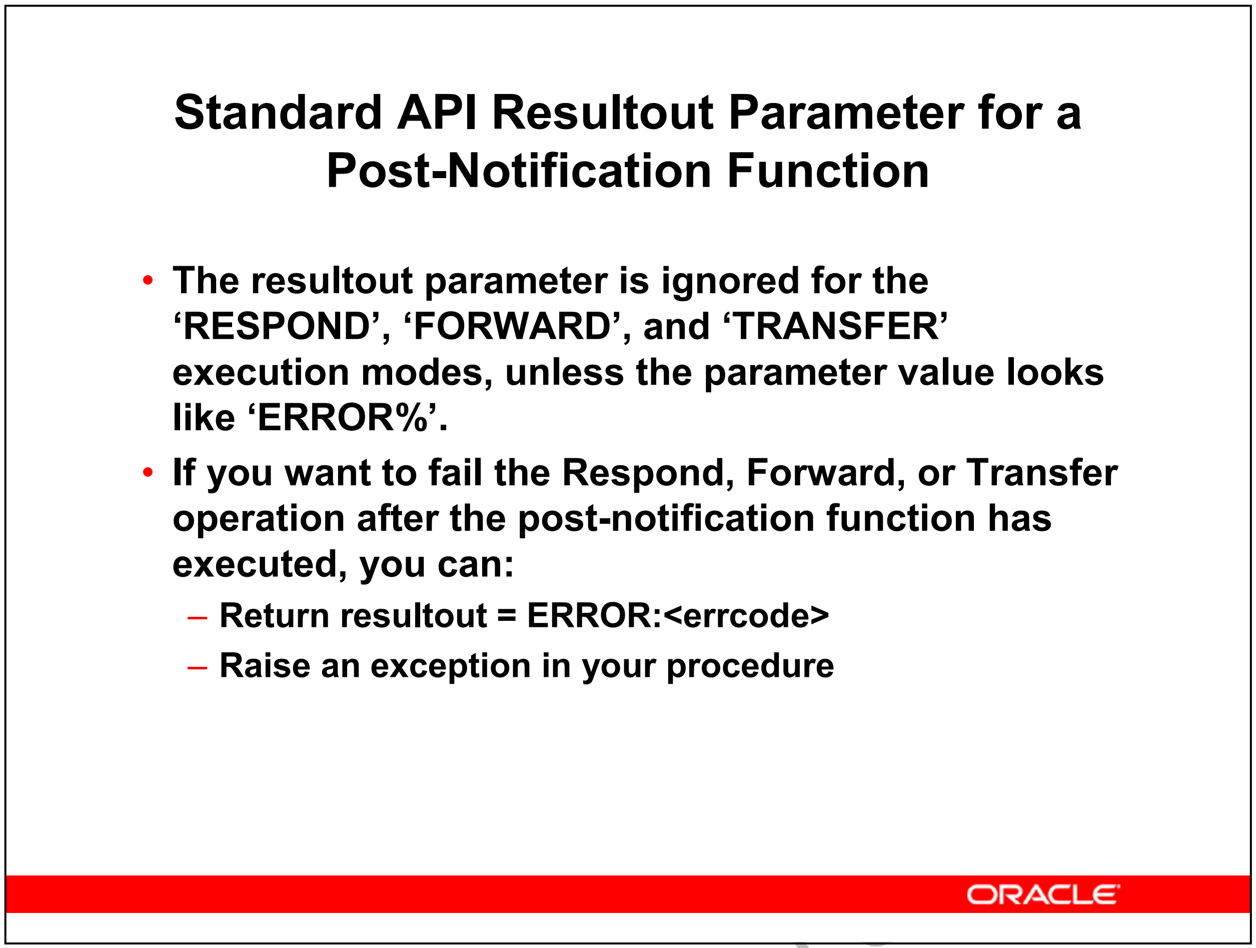
- RESPOND
- FORWARD
- TRANSFER
- RUN
- TIMEOUT

Post-Notification Function Modes

- RESPOND: The performer responded to the notification. The post-notification function can interpret the result and perform post-response processing. The function can reject the response with an error.
- FORWARD: The performer delegates the notification to another user. The function can audit or reject the delegation request with an error. This state corresponds to the Delegate Authority option in the Reassign Notification page.
- TRANSFER: The performer transfers the notification to another user. The function can audit or reject the transfer request with an error. This state corresponds to the Transfer Ownership option in the Reassign Notification web page.
- RUN: Called after RESPOND, FORWARD, or TRANSFER modes. The post-notification function in RUN mode can perform additional processing associated with the notification.
- TIMEOUT: If a notification activity does not complete within a certain period of time, then Oracle Workflow marks that activity as timed out and then cancels any notification associated with the timed out activity. The post-notification function in TIMEOUT mode can perform processing to determine an alternative result for the notification. For example, if the notification is a vote and some responses are received before the timeout event, the

post-notification function can determine whether there are enough votes received to determine the vote result.

Note: If the post-notification function in RESPOND, FORWARD or TRANSFER mode returns ERROR:<errcode> as a result or raises an exception, the Workflow Engine aborts the respond, forward, or transfer operation.



Standard API Resultout Parameter for a Post-Notification Function

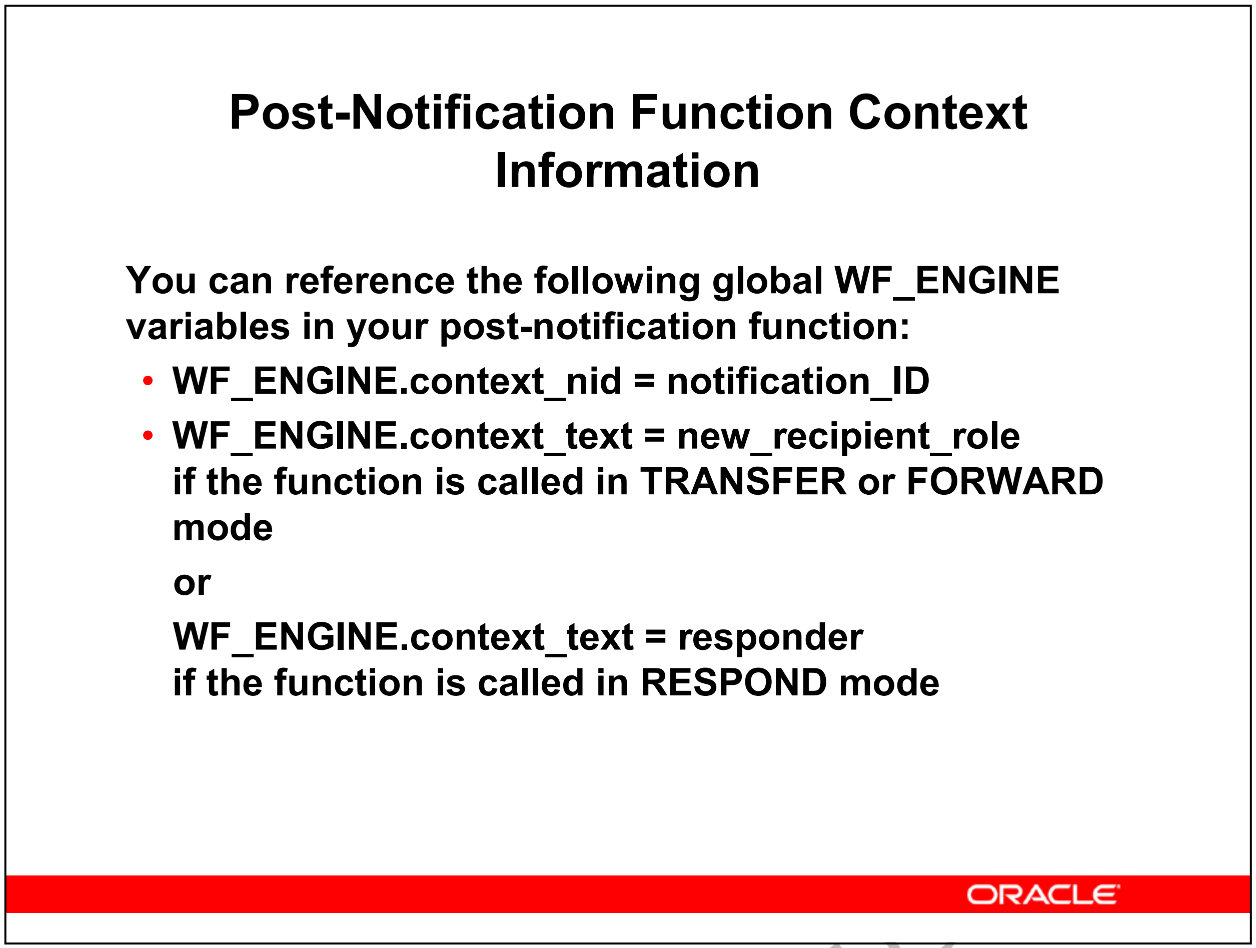
- **The resultout parameter is ignored for the ‘RESPOND’, ‘FORWARD’, and ‘TRANSFER’ execution modes, unless the parameter value looks like ‘ERROR%’.**
- **If you want to fail the Respond, Forward, or Transfer operation after the post-notification function has executed, you can:**
  - **Return resultout = ERROR:<errcode>**
  - **Raise an exception in your procedure**

ORACLE

Standard API Resultout Parameter for a Post-Notification Function

If you do not want the Respond, Forward or Transfer operation to occur after having executed your post-notification function, you can do one of two things:

- Return 'ERROR:<errcode>' in the resultout parameter to convert the result to a generic exception with the error code specified in the message.
- Raise an exception directly in your procedure with a more informative error message.



Post-Notification Function Context Information

You can reference the following global WF_ENGINE variables in your post-notification function:

- **WF_ENGINE.context_nid = notification_ID**
- **WF_ENGINE.context_text = new_recipient_role if the function is called in TRANSFER or FORWARD mode**

  or

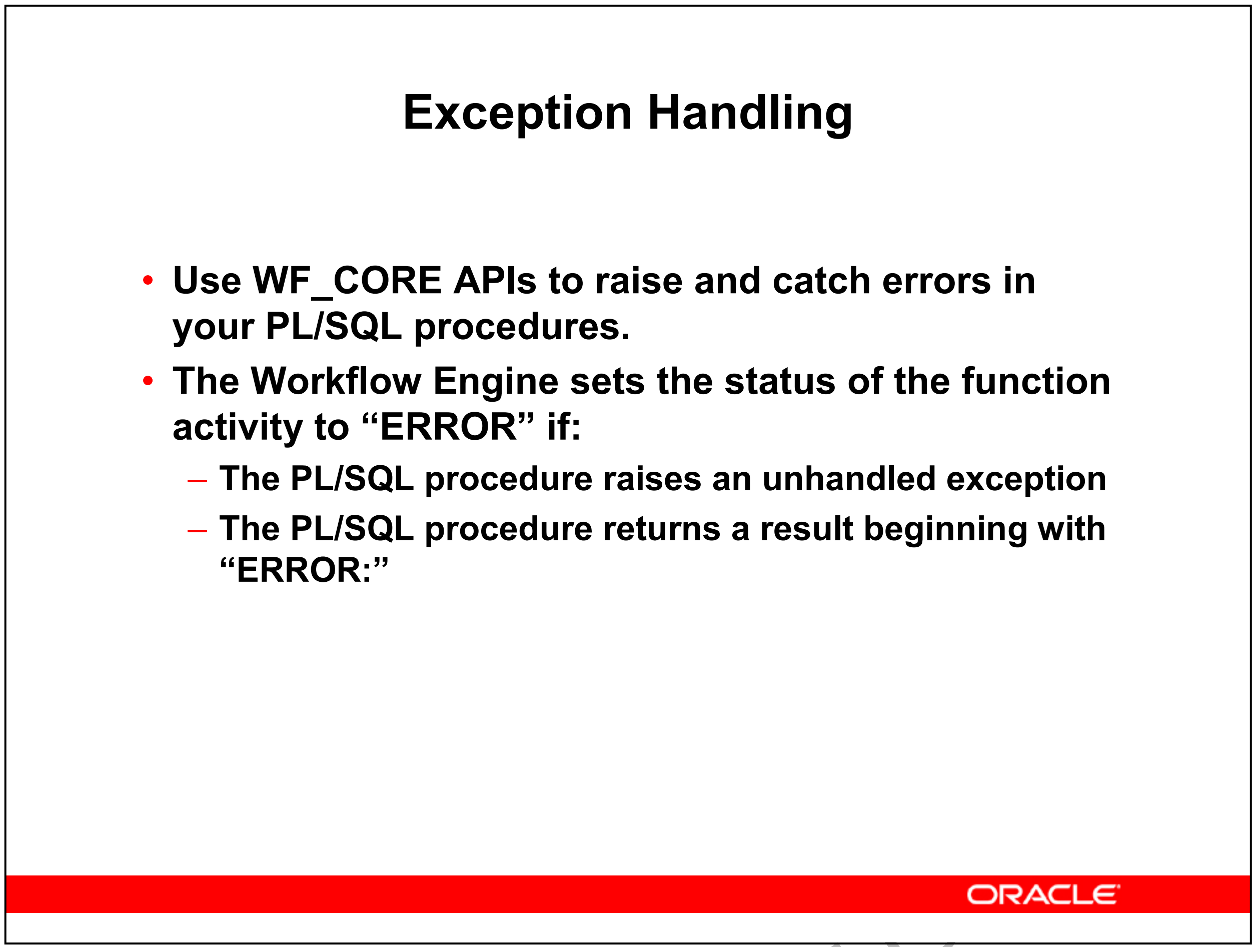
  WF_ENGINE.context_text = responder if the function is called in RESPOND mode

ORACLE

Post-Notification Function Context Information

For context_text, the value of *responder* varies depending on the notification interface the recipient uses to respond.

- If the recipient responds using the Notification Details web page, *responder* is set to the role name of the responder.
- If the recipient responds using e-mail, *responder* is set to "email:<responder_email_address>".



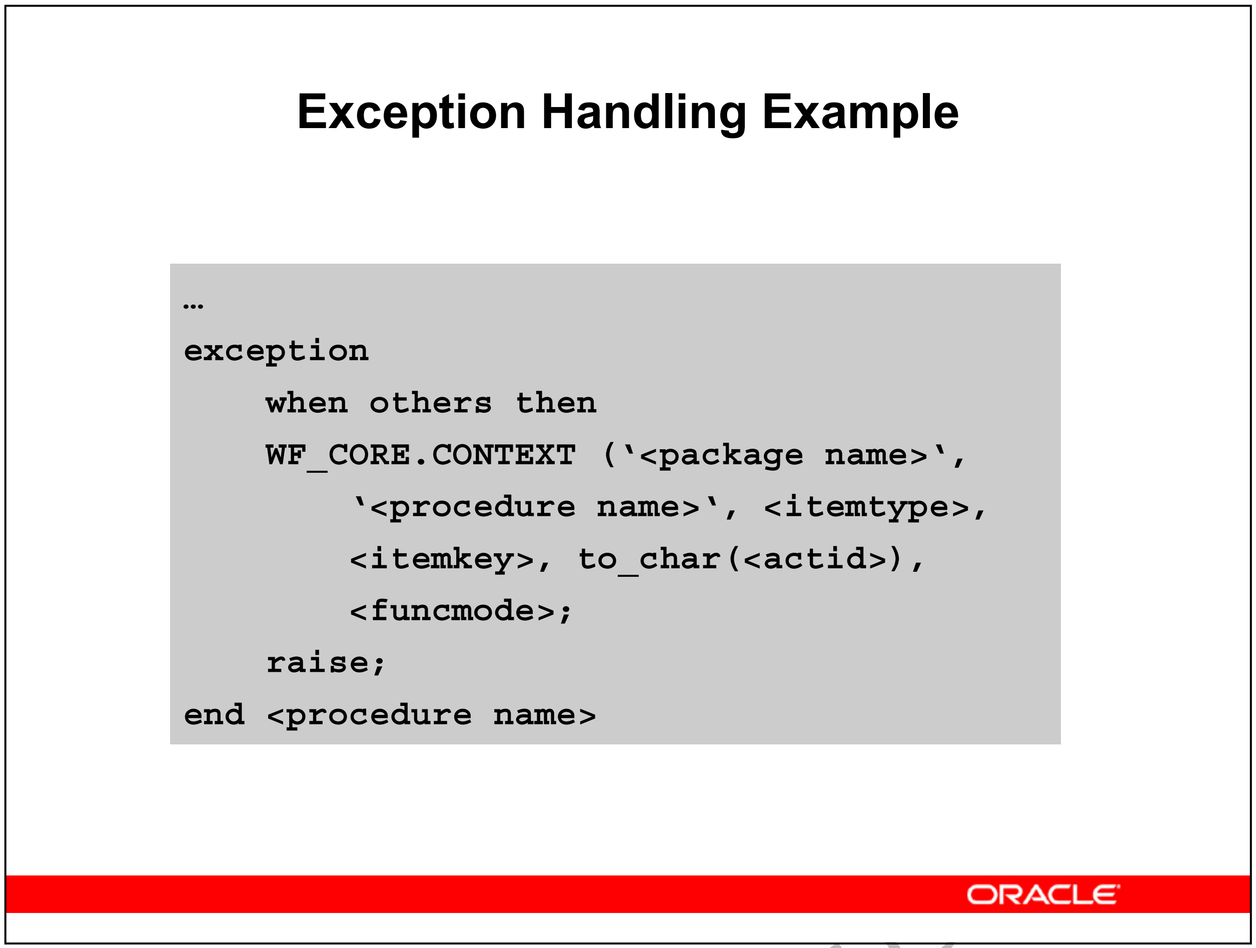
Exception Handling

- Use WF_CORE APIs to raise and catch errors in your PL/SQL procedures.
- The Workflow Engine sets the status of the function activity to "ERROR" if:
  - The PL/SQL procedure raises an unhandled exception
  - The PL/SQL procedure returns a result beginning with "ERROR:"

Exception Handling

If an activity encounters an error, information about the error is stored in the following columns in the WF_ITEM_ACTIVITY_STATUSES table, which are viewable from the Workflow Monitor.

- ERROR_NAME
- ERROR_MESSAGE
- ERROR_STACK



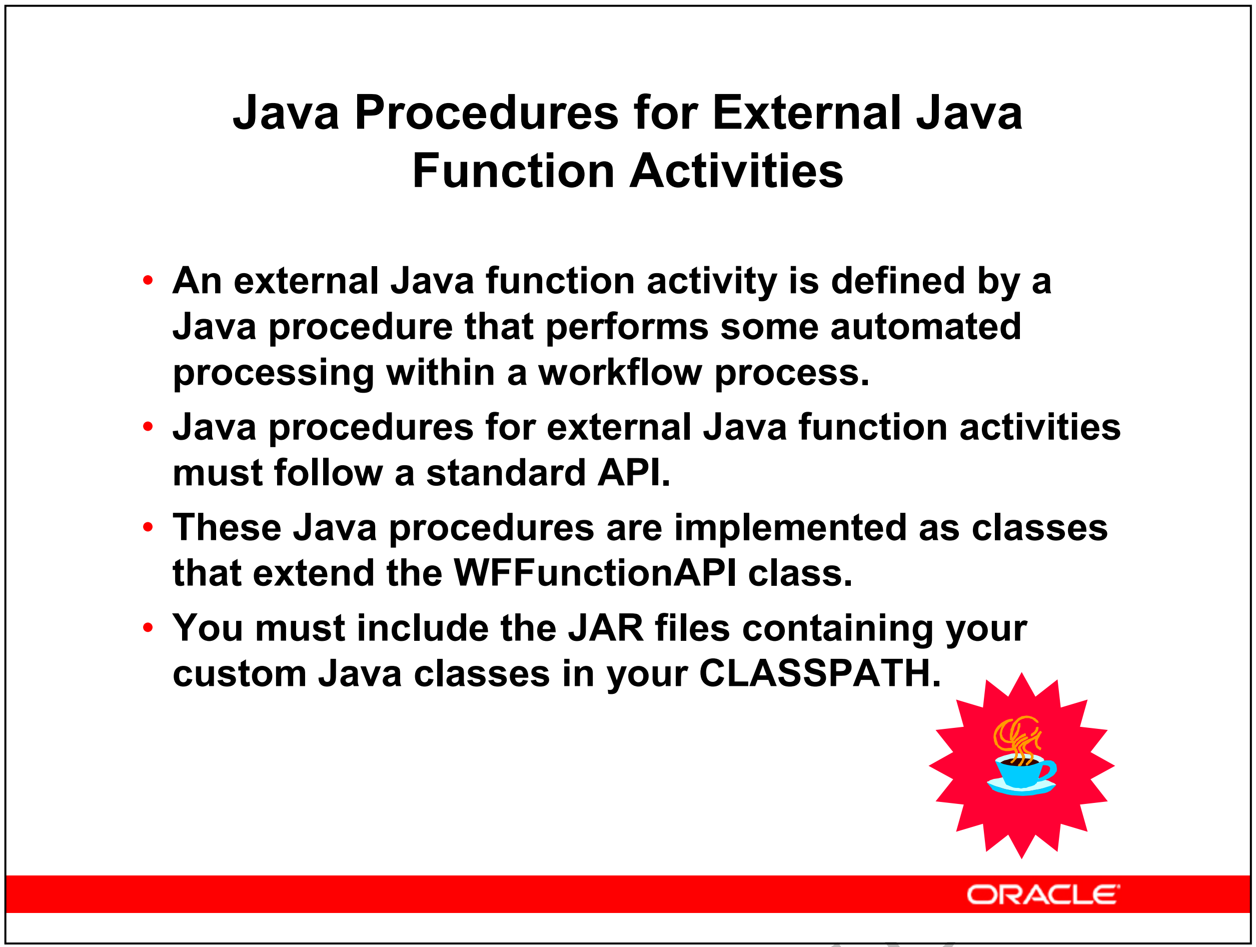
Exception Handling Example

```
…
exception
    when others then
    WF_CORE.CONTEXT (‘<package name>‘,
        ‘<procedure name>‘, <itemtype>,
        <itemkey>, to_char(<actid>),
        <funcmode>;
    raise;
end <procedure name>
```

ORACLE

Exception Handling Example

- WF_CORE.CONTEXT adds an entry to the error stack to provide context information that helps locate the source of an error.
- Use this example exception handler construct in your PL/SQL procedures to facilitate debugging workflow function activities.
- If a call to ‘procedure name’ fails with an unhandled exception, the column ERROR_STACK in the WF_ITEM_ACTIVITY_STATUSES table records the call to ‘procedure name’ and its arguments.



Java Procedures for External Java Function Activities

- An external Java function activity is defined by a Java procedure that performs some automated processing within a workflow process.
- Java procedures for external Java function activities must follow a standard API.
- These Java procedures are implemented as classes that extend the WFFunctionAPI class.
- You must include the JAR files containing your custom Java classes in your CLASSPATH.

External Java Function Activities

A Java procedure for an external Java function activity must have the following standard API:

```
package oracle.apps.fnd.wf;

import java.io.*;
import java.sql.*;
import java.math.BigDecimal;
import oracle.sql.*;
import oracle.jdbc.driver.*;

import oracle.apps.fnd.common.*;
import oracle.apps.fnd.wf.engine.*;
import oracle.apps.fnd.wf.*;
```

```
public class className extends WFFunctionAPI {
  public boolean execute(WFContext pWCtx){
    ErrorStack es = pWCtx.getWFErrorStack();
      try
      {

        WFAttribute lAAttr = new WFAttribute();
        WFAttribute lIAttr = new WFAttribute();

        loadActivityAttributes(pWCtx);
        loadItemAttributes(pWCtx);

        lAAttr = getActivityAttr("AATTR");
        lIAttr = getItemAttr("IATTR");

        <your executable statements>

        lIAttr.value((Object)"NEWVALUE");
        setItemAttrValue(pWCtx, lIAttr);

      }
        catch (Exception e)
        {
          es.addMessage("WF","WF_FN_ERROR");
          es.addToken("MODULE",this.getClass().getName());
          es.addToken("ITEMTYPE",itemType);
          es.addToken("ITEMKEY",itemKey);
          es.addToken("ACTID",actID.toString());
          es.addToken("FUNCMODE",funcMode);
          es.addToken("ERRMESSAGE",e.getMessage());
          return false;
        }

      return true;
  }
}
```

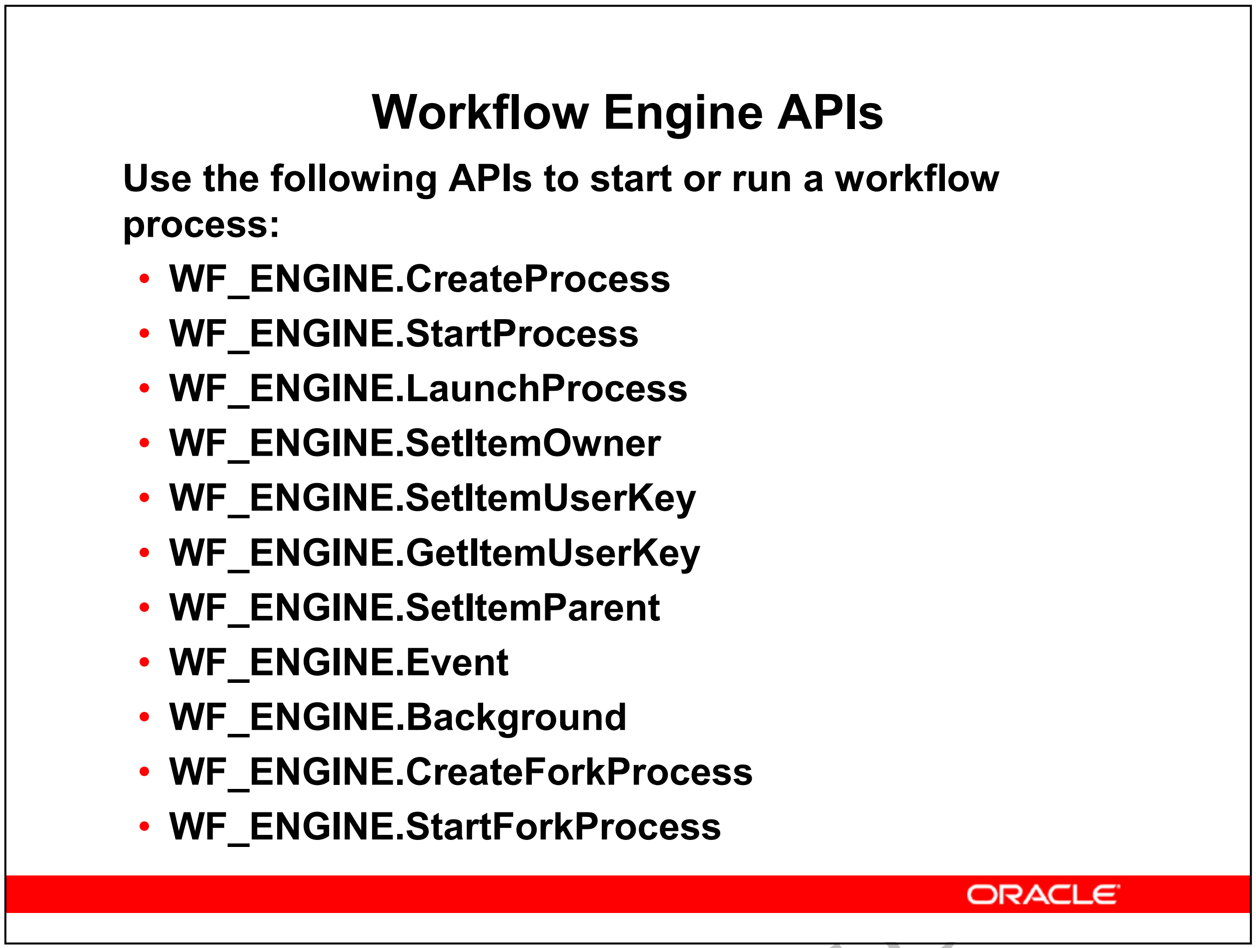
The parameters available as class variables are:

- itemType—The internal name for the item type.

- itemKey—A string that represents a primary key generated by the workflow-enabled application for the item type. The string uniquely identifies the item within an item type.
- ActID—The ID number of the activity from which this procedure is called.
- funcMode—The execution mode of the activity. Currently the only supported mode for external Java function activities is the 'RUN' mode.
- resultOut—If a result type is specified in the Activities properties page for the activity in the Oracle Workflow Builder, this parameter represents the expected result that is returned when the procedure completes.

  Note: Unlike the resultout for a PL/SQL procedure called by a function activity, the resultOut for a Java procedure does not include a status code. In the Java API, only the result type value is required. The status of the activity will be set automatically by the Workflow Engine depending on whether there is a value in the errorStack variable.

For more information, refer to the Defining Procedures and Functions for Oracle Workflow chapter in the *Oracle Workflow Guide*.



Workflow Engine APIs

Use the following APIs to start or run a workflow process:

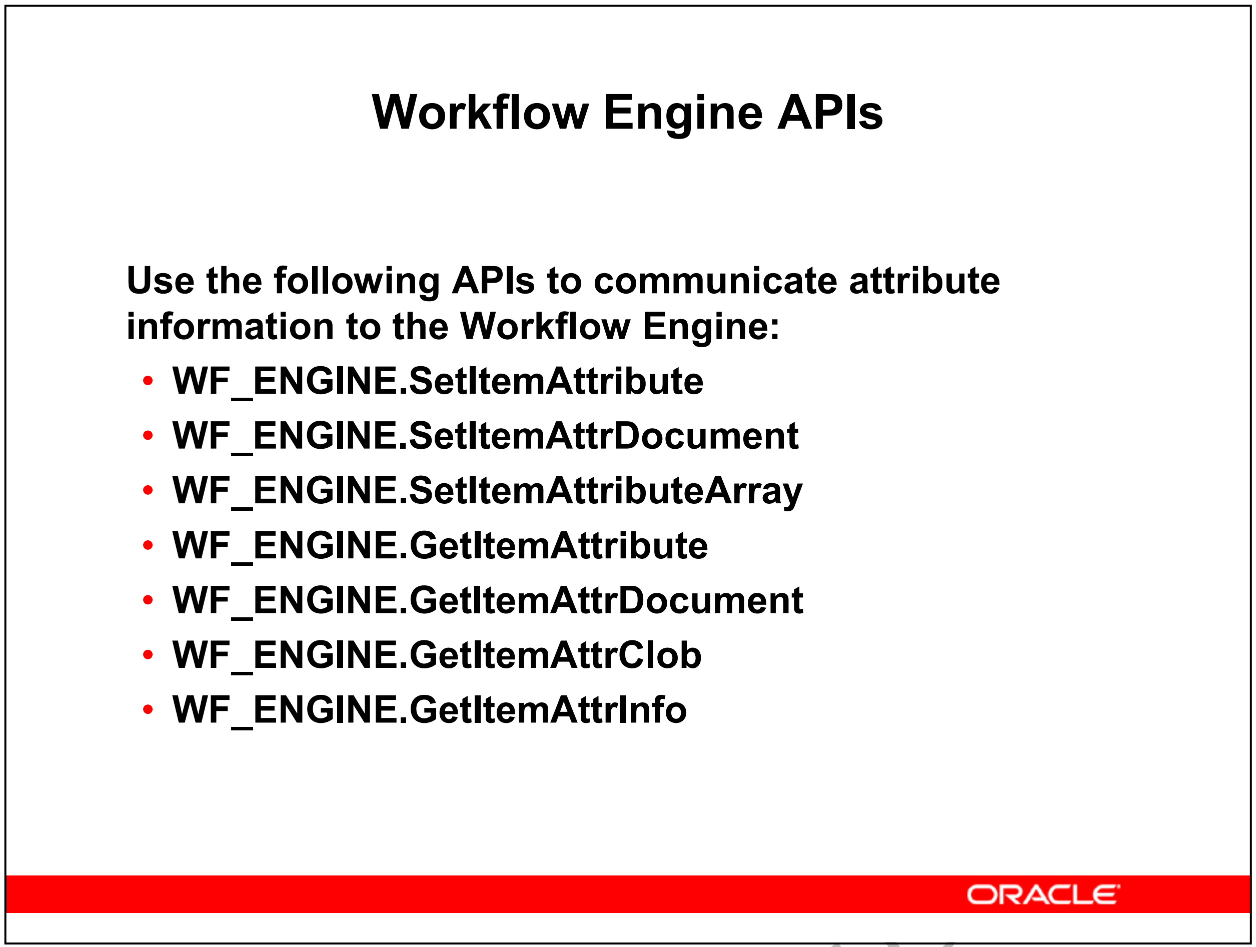
- **WF_ENGINE.CreateProcess**
- **WF_ENGINE.StartProcess**
- **WF_ENGINE.LaunchProcess**
- **WF_ENGINE.SetItemOwner**
- **WF_ENGINE.SetItemUserKey**
- **WF_ENGINE.GetItemUserKey**
- **WF_ENGINE.SetItemParent**
- **WF_ENGINE.Event**
- **WF_ENGINE.Background**
- **WF_ENGINE.CreateForkProcess**
- **WF_ENGINE.StartForkProcess**

Workflow Engine APIs

The following APIs start or run a workflow process:

- WF_ENGINE.CreateProcess creates a new runtime process for a work item.
- WF_ENGINE.StartProcess begins execution of the specified process.
- WF_ENGINE.LaunchProcess launches a specified process by creating the new runtime process and beginning its execution.
- WF_ENGINE.SetItemOwner sets the owner of an existing item.
- WF_ENGINE.SetItemUserKey sets a user-friendly identifier for an item.
- WF_ENGINE.GetItemUserKey returns the user-friendly identifier assigned to an item.
- WF_ENGINE.SetItemParent defines the parent/child relationship for master/detail processes.
- WF_ENGINE.Event receives an event from the Business Event System into a workflow process.
- WF_ENGINE.Background runs a background engine to process deferred and timed out activities and stuck processes.

- WF_ENGINE.CreateForkProcess forks a runtime process by creating a new process that is a copy of the original.
- WF_ENGINE.StartForkProcess begins execution of the specified new forked process.



Workflow Engine APIs

Use the following APIs to communicate attribute information to the Workflow Engine:

- **WF_ENGINE.SetItemAttribute**
- **WF_ENGINE.SetItemAttrDocument**
- **WF_ENGINE.SetItemAttributeArray**
- **WF_ENGINE.GetItemAttribute**
- **WF_ENGINE.GetItemAttrDocument**
- **WF_ENGINE.GetItemAttrClob**
- **WF_ENGINE.GetItemAttrInfo**

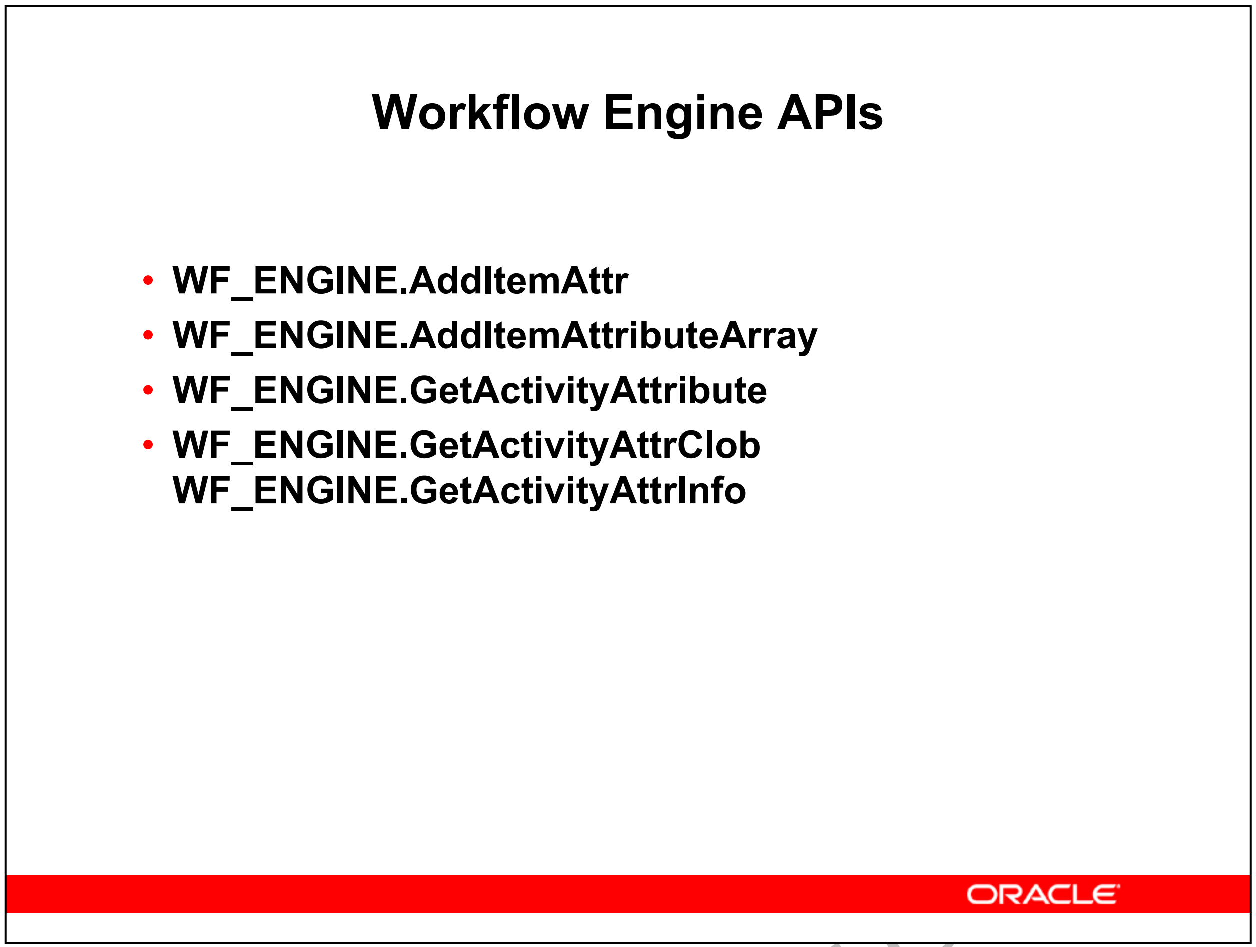
ORACLE

Workflow Engine APIs

The following APIs communicate attribute information to the Workflow Engine:

- WF_ENGINE.SetItemAttrText, WF_ENGINE.SetItemAttrNumber, WF_ENGINE.SetItemAttrDate, and WF_ENGINE.SetItemAttrEvent set the value of an item type attribute in a process.
- WF_ENGINE.SetItemAttrDocument sets the value of an item attribute of type document to a document identifier.
- WF_ENGINE.SetItemAttrTextArray, WF_ENGINE.SetItemAttrNumberArray, and WF_ENGINE.SetItemAttrDateArray set the values of an array of item type attributes in a process.
- WF_ENGINE.GetItemAttrText, WF_ENGINE.GetItemAttrNumber, WF_ENGINE.GetItemAttrDate, and WF_ENGINE.GetItemAttrEvent return the value of an item type attribute in a process.
- WF_ENGINE.GetItemAttrDocument returns the document identifier of an item attribute of type document.
- WF_ENGINE.GetItemAttrClob returns the value of an item type attribute in a process as a character large object (CLOB).

- WF_ENGINE.GetItemAttrInfo returns information about an item attribute, such as its type and format.

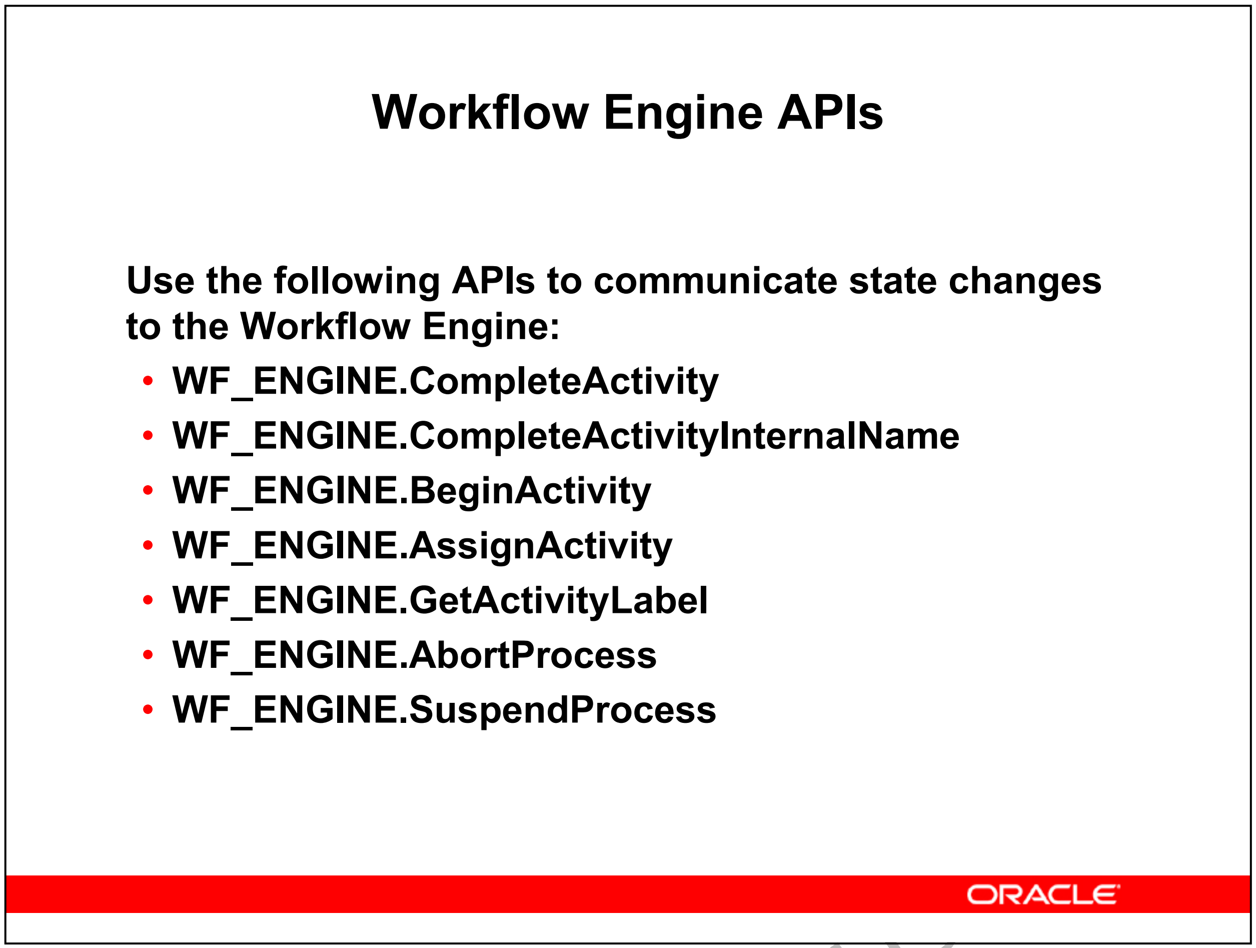


Workflow Engine APIs

- WF_ENGINE.AddItemAttr
- WF_ENGINE.AddItemAttributeArray
- WF_ENGINE.GetActivityAttribute
- WF_ENGINE.GetActivityAttrClob
  WF_ENGINE.GetActivityAttrInfo

Workflow Engine APIs

- WF_ENGINE.AddItemAttr adds a new item attribute to the runtime process.
- WF_ENGINE.AddItemAttrTextArray, WF_ENGINE.AddItemAttrNumberArray, and WF_ENGINE.AddItemAttrDateArray add an array of new item type attributes to the runtime process.
- WF_ENGINE.GetActivityAttrText, WF_ENGINE.GetActivityAttrNumber, WF_ENGINE.GetActivityAttrDate, and WF_ENGINE.GetActivityAttrEvent return the value of an activity attribute in a process.
- WF_ENGINE.GetActivityAttrClob returns the value of an activity attribute in a process as a character large object (CLOB).
- WF_ENGINE.GetActivityAttrInfo returns information about an activity attribute, such as its type and format.



Workflow Engine APIs

Use the following APIs to communicate state changes to the Workflow Engine:

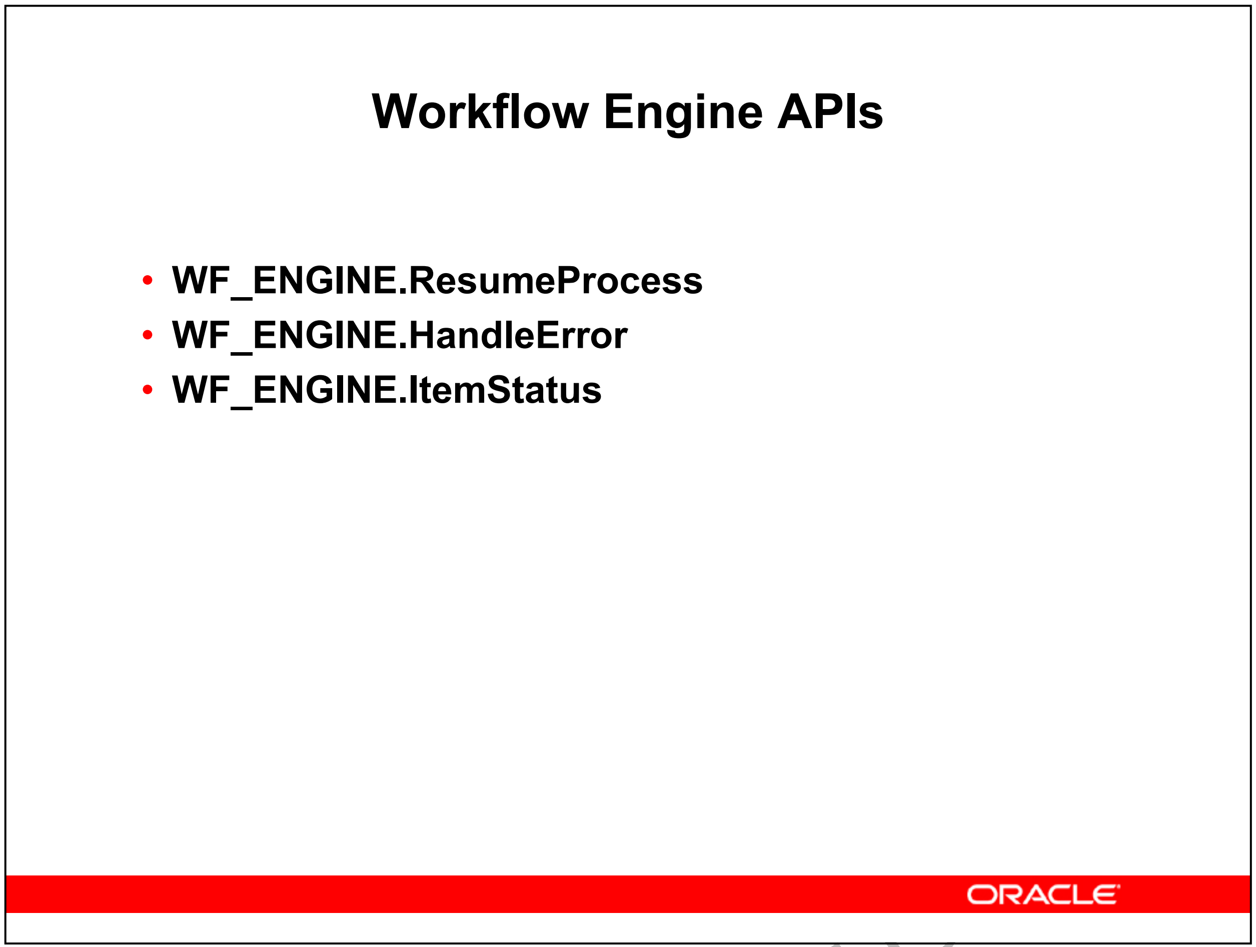
- **WF_ENGINE.CompleteActivity**
- **WF_ENGINE.CompleteActivityInternalName**
- **WF_ENGINE.BeginActivity**
- **WF_ENGINE.AssignActivity**
- **WF_ENGINE.GetActivityLabel**
- **WF_ENGINE.AbortProcess**
- **WF_ENGINE.SuspendProcess**

ORACLE

Workflow Engine APIs

The following APIs communicate state changes to the Workflow Engine:

- WF_ENGINE.CompleteActivity notifies the engine that the specified activity has been completed for the item, identifying the activity by the activity node label name.
- WF_ENGINE.CompleteActivityInternalName notifies the engine that the specified activity has been completed for the item, identifying the activity by its internal name.
- WF_ENGINE.BeginActivity determines if the specified activity can currently be performed and raises an exception if it cannot.
- WF_ENGINE.AssignActivity assigns an activity to another performer.
- WF_ENGINE.GetActivityLabel returns the instance label of an activity, given the internal activity instance identification.
- WF_ENGINE.AbortProcess aborts process execution and cancels outstanding notifications.
- WF_ENGINE.SuspendProcess suspends process execution so that users cannot transition items to new activities.



Workflow Engine APIs

- WF_ENGINE.ResumeProcess
- WF_ENGINE.HandleError
- WF_ENGINE.ItemStatus

ORACLE

Workflow Engine APIs

- WF_ENGINE.ResumeProcess returns a suspended process to normal execution status.
- WF_ENGINE.HandleError handles any activity that has encountered an error. This API can also be called for any arbitrary activity in a process to roll back part of the process to that activity.
- WF_ENGINE.ItemStatus returns the status and results for the root process of the specified item instance.

Practice - Defining PL/SQL Procedures for Function Activities

Overview

In this practice, you will define PL/SQL procedures for the Update Vacation Schedule and Approver Same as Requestor? function activities in the Vacation Proposal process. The PL/SQL procedure for the Update Vacation Schedule activity should update a schedule of planned vacations, and the PL/SQL procedure for the Approver Same as Requestor? Activity should check whether the approver is the same as the requestor. In the Defining a Function Activity and Branching on a Function Activity Result practices, you ran these function activities with predefined sample procedures. In this practice, you will develop your own procedures to use in place of the predefined procedures.

- Define a PL/SQL procedure for the Update Vacation Schedule function activity.
- Define a PL/SQL procedure for the Approver Same as Requestor? function activity.
- Test the PL/SQL procedures by running the process.

Note: Because many students access the system and create objects during this course, you need a way to distinguish between the objects created by you and by your classmates. Therefore, you will be assigned a terminal number by your instructor. Use this number as a prefix wherever you see *XX* included in the name of something you are defining. In this way, you can ensure that the definitions you create are unique.

Assumptions

- The instructor will provide you with the connect string for the class database and the username and password of the Oracle Workflow database account.
- The instructor will provide you with the username and password of a user with workflow administrator privileges. The workflow administrator is defined in the Global Workflow Preferences page.
- For standalone Oracle Workflow, the instructor will provide you with the URL for the Oracle Workflow home page. The URL is <*webagent*>/wfa_html.home, where <*webagent*> is the base URL of the web agent configured for Oracle Workflow in your Web server.
- For Oracle Workflow embedded in Oracle E-Business Suite, the instructor will provide you with the name of a Workflow administrator responsibility. The username you use to log in should have this responsibility assigned to it.
- The instructor will provide you with the names of users that you can assign as the requestor and approver in the Vacation Proposal process.

Tasks

1. Define a PL/SQL procedure for the Update Vacation Schedule function activity.

   Solution:

   1. Your PL/SQL procedure will update a table named WFVAC*XX*_VACATION_SCHEDULE. If you have not already done so, copy and edit the sample table creation script named wfvacxxc.sql. Open a copy of the sample file and replace all instances of XX with your own terminal number. Save the file and rename it by replacing *xx* with your terminal number. Then log in to SQL*Plus using the database username, password, and connect string provided by the instructor, and run the table creation script.

      This script creates a vacation schedule table named WFVAC*XX*_VACATION_SCHEDULE. The table includes the following columns:
      – REQUESTOR - varchar2(30)
      – APPROVER - varchar2(30)
      – FROM_DATE - date
      – TO_DATE - date

   2. Using Wordpad, write SQL scripts for the specification and body of a PL/SQL package, named wfvac*xx*s.sql and wfvac*xx*b.sql, respectively. Name the package WFVAC*XX* and create a procedure named SCHEDULE_UPDATE that records an approved vacation proposal.
      – If the function activity is called by the Workflow Engine in RUN mode, insert the requestor, approver, and vacation dates into your vacation schedule table.
      – If the function activity is called by the Workflow Engine in CANCEL mode, delete the appropriate requestor, approver, and vacation date row from your vacation schedule table.

      Note: By using the same names for your PL/SQL package and procedure as the predefined sample solutions, you can avoid having to modify the function activity that you previously defined. If you use different package and procedure names, you must update the function activity in the Workflow Builder to reference the new names before you can run the process to test your procedure.

   3. Use the standard Workflow API for the SCHEDULE_UPDATE procedure.

```
procedure <procedure_name>
     (itemtype  in  varchar2,
      itemkey   in  varchar2,
      actid     in  number,
      funcmode  in  varchar2,
      resultout out varchar2)
```

   4. In the SCHEDULE_UPDATE procedure, include logic for the RUN funcmode that:
      – Retrieves the values for the requestor, approver, vacation from date, and vacation to date using API calls to WF_ENGINE.GetItemAttrText and WF_ENGINE.GetItemAttrDate

- Inserts a row into your WFVAC*XX*_VACATION_SCHEDULE table using the appropriate item attribute values
- Sets resultout to wf_engine.eng_completed||':'||wf_engine.eng_null
- Returns

5. In the SCHEDULE_UPDATE procedure, include logic for the CANCEL funcmode that:
   - Retrieves the values for the requestor, approver, vacation from date, and vacation to date using API calls to WF_ENGINE.GetItemAttrText and WF_ENGINE.GetItemAttrDate
   - Deletes the appropriate row from your WFVAC*XX*_VACATION_SCHEDULE table based on these item attribute values
   - Sets resultout to wf_engine.eng_completed||':'||wf_engine.eng_null
   - Returns

6. You can look at the sample package specification and body scripts provided with this course to see a sample solution procedure. The sample scripts are named wfvacxxs.sql and wfvacxxb.sql, respectively. They contain all the sample PL/SQL procedures for all practices in this course. The procedure specific to this task is WFVACXX.SCHEDULE_UPDATE.

2. Define a PL/SQL procedure for the Approver Same as Requestor? function activity.

Solution:

7. Using Wordpad, open the SQL scripts you created for the specification and body of a PL/SQL package, named wfvac*xx*s.sql and wfvac*xx*b.sql, respectively. In the WFVAC*XX* package, create a new procedure named CHECK_APPROVER that checks whether the approver is the same as the requestor. The procedure should have the same name as the internal name of the function activity that will call it.

   Note: By using the same names for your PL/SQL package and procedure as the predefined sample solutions, you can avoid having to modify the function activity that you previously defined. If you use different package and procedure names, you must update the function activity in the Workflow Builder to reference the new names before you can run the process to test your procedure.

8. Use the standard Workflow API for the CHECK_APPROVER procedure.

```
procedure <procedure_name>
     (itemtype  in  varchar2,
      itemkey   in  varchar2,
      actid     in  number,
      funcmode  in  varchar2,
      resultout out varchar2)
```

9. In the CHECK_APPROVER procedure, include logic for the RUN funcmode.
   - Retrieve the values for the requestor and approver using API calls to WF_ENGINE.GetItemAttrText

- If requestor <> approver, set resultout to wf_engine.eng_completed||':'||'N'
- If requestor = approver, set resultout to wf_engine.eng_completed||':'||'Y'
- Return

10. In the CHECK_APPROVER procedure, do not include any special logic for the CANCEL funcmode. Set resultout to wf_engine.eng_completed||':'||wf_engine.eng_null to indicate that the funcmode is not implemented.

11. You can look at the sample package specification and body scripts provided with this course to see a sample solution procedure. The sample scripts are named wfvacxxs.sql and wfvacxxb.sql, respectively. They contain all the sample PL/SQL procedures for all practices in this course. The procedure specific to this practice is WFVACXX.CHECK_APPROVER.

12. After completing your scripts, log in to SQL*Plus using the database username, password, and connect string provided by the instructor, and run the package specification and package body scripts in that order.

3. Test the PL/SQL procedures by running the process.

Solution:

13. If you used the recommended package and procedure names, which are the same as those in the sample solutions, you should not need to modify the function activities you defined previously. However, you can use the Oracle Workflow Builder to open your wfvac*XX*.wft data store, verify the procedures assigned to the Update Vacation Schedule and Approver Same as Requestor? function activities, and update them if necessary. If you make any modifications, choose Save from the File menu to save your work to your workflow definition file, and then choose Save As from the File menu and save your item type to the class database, using the database username, password, and connect string provided by the instructor.

14. Use a web browser to connect to the Oracle Workflow home page with the URL provided by the instructor for standalone Oracle Workflow, or to a Workflow administrator responsibility provided by the instructor for Oracle Workflow embedded in Oracle E-Business Suite. Log in as a user with workflow administrator privileges.

15. Use the Launch Processes page to launch your workflow process and test your work. You can use the Notifications Worklist to view the notifications sent by the process and approve the vacation proposal, and use the Workflow Monitor to review the status of the process.

16. To test the CHECK_APPROVER procedure, run the process and enter the same role for the requestor and the approver. The process should end with a result of Reject.

    Then, run the process again and enter different roles for the requestor and the approver. The process should proceed to send the Vacation Proposal notification to the approver.

17. To test the RUN mode logic for the SCHEDULE_UPDATE procedure, approve the vacation proposal. Use SQL*Plus to verify that the appropriate row was inserted into your WFVAC*XX*_VACATION_SCHEDULE table. Enter the following command:

```
select * from WFVACXX_VACATION_SCHEDULE;
```

18. To test the CANCEL mode logic for the SCHEDULE_UPDATE procedure, review the diagram for the completed work item in the Workflow Monitor.
    – Rewind the work item to the Start node by selecting the Start node and clicking the Expedite button. Then select Retry and click the OK button.
    – Use SQL*Plus to verify that the appropriate row was removed from your WFVAC*XX*_VACATION_SCHEDULE table. Enter the following command:

    ```
    select * from WFVACXX_VACATION_SCHEDULE;
    ```

Practice - Defining a Post-Notification Function

Overview

In this practice, you will define a post-notification function to validate action taken on the Vacation Proposal notification in the Vacation Proposal item type you created in the Creating a Workflow Process practice.

- Define the PL/SQL procedure for the post-notification function.
- Assign the post-notification function to the notification activity.

Note: Because many students access the system and create objects during this course, you need a way to distinguish between the objects created by you and by your classmates. Therefore, you will be assigned a terminal number by your instructor. Use this number as a prefix wherever you see *XX* included in the name of something you are defining. In this way, you can ensure that the definitions you create are unique.

Note: In order to use the sample solution scripts provided for these practices, you must enter the internal names for all objects you define exactly as shown in the instructions. Otherwise, you must modify the sample code to reference the object names you define.

Assumptions

- The instructor will provide you with the connect string for the class database and the username and password of the Oracle Workflow database account.
- The instructor will provide you with the username and password of a user with workflow administrator privileges. The workflow administrator is defined in the Global Workflow Preferences page.
- For standalone Oracle Workflow, the instructor will provide you with the URL for the Oracle Workflow home page. The URL is <*webagent*>/wfa_html.home, where <*webagent*> is the base URL of the web agent configured for Oracle Workflow in your Web server.
- For Oracle Workflow embedded in Oracle E-Business Suite, the instructor will provide you with the name of a Workflow administrator responsibility. The username you use to log in should have this responsibility assigned to it.
- The instructor will provide you with the names of users that you can assign as the requestor, approver, and new recipient in the Vacation Proposal process.

Tasks

1. Define the PL/SQL procedure for the post-notification function.

Solution:

1. Using Wordpad, open the SQL scripts you created for the specification and body of a PL/SQL package, named wfvacxxs.sql and wfvacxxb.sql, respectively. In the WFVACXX package, create a new procedure named NTF_VACATION_PROPOSAL that validates action taken on the Vacation Proposal notification. The procedure should have the same name as the internal name of the notification activity that it validates.
   - If the approver rejects the vacation proposal, alternate vacation dates must be provided, and the alternate from date must be prior to the alternate to date.
   - The approver is allowed to delegate (forward) the notification to another role.
   - The approver is not allowed to transfer the notification to another role.

2. Use the standard Workflow API for the NTF_VACATION_PROPOSAL procedure.

```
procedure <procedure_name>
     (itemtype  in  varchar2,
      itemkey   in  varchar2,
      actid     in  number,
      funcmode  in  varchar2,
      resultout out varchar2)
```

3. In the NTF_VACATION_PROPOSAL procedure, include logic for the RESPOND funcmode.
   - Get the notification ID (nid) from the engine context variable wf_engine.context_nid
   - Retrieve the response result using the API call WF_NOTIFICATION.GetAttrText(nid, 'RESULT')
   - If the response is APPROVED, no further validation is needed; set resultout to wf_engine.eng_completed||':'||<ntf result retrieved> and return
   - If the response is REJECTED, retrieve the alternate from and to dates using the API call WF_NOTIFICATION.GetAttrDate(nid, <attribute_name>)
   - If the alternate dates are valid, set result out to wf_engine.eng_completed||':'||<ntf result retrieved> and return
   - If the alternate dates are missing or if the alternate from date is after the alternate to date, raise an exception using the API call WF_CORE.Raise(<your message>) and set resultout to wf_engine.eng_error||':'||wf_engine.eng_null and return

4. In the NTF_VACATION_PROPOSAL procedure, include logic for the TRANSFER funcmode.
   - Raise an exception using the API call WF_CORE.Raise(<your message>)
   - Set resultout to wf_engine.eng_error||':'||wf_engine.eng_null and return

5. In the NTF_VACATION_PROPOSAL procedure, do not include any special logic for the FORWARD, RUN, or TIMEOUT funcmodes. For each of these funcmodes, set resultout to wf_engine.eng_null to indicate that the funcmode is not implemented and return.

6. You can optionally copy and edit the sample package specification and body scripts instead of coding your own scripts. The sample scripts are named wfvacxxs.sql and wfvacxxb.sql, respectively. They contain all the sample PL/SQL procedures for all practices in this course. The procedure specific to this practice is WFVACXX.NTF_VACATION_PROPOSAL. Open a copy of each sample file and replace all instances of XX with your own terminal number. Save the files and rename them by replacing *xx* with your terminal number.

   Note: If you have already copied, edited, and run the complete sample package specification and body scripts containing all the sample procedures for all practices, and you did not overwrite the sample package with your own package in the previous practice, you do not need to repeat these steps. However, if you created your own package of the same name as the sample package in the Defining PL/SQL Procedures for Function Activities practice and loaded it to the database, you must reload the sample package before you can use the sample procedures in your process.

7. Log in to SQL*Plus using the database username, password, and connect string provided by the instructor. Run the package specification and package body scripts in that order.

2. Assign the post-notification function to the notification activity.

   Solution:

8. Start the Oracle Workflow Builder.

9. From the File menu, choose Open to open the wfvac*XX*.wft data store you defined in the Creating a Workflow Process practice.

10. In the navigator tree, select your item type.

11. Open the property pages for the Vacation Proposal notification.

12. In the Function Name field, enter the package and procedure name WFVAC*XX*.NTF_VACATION_PROPOSAL for the activity. The function type defaults to PL/SQL.

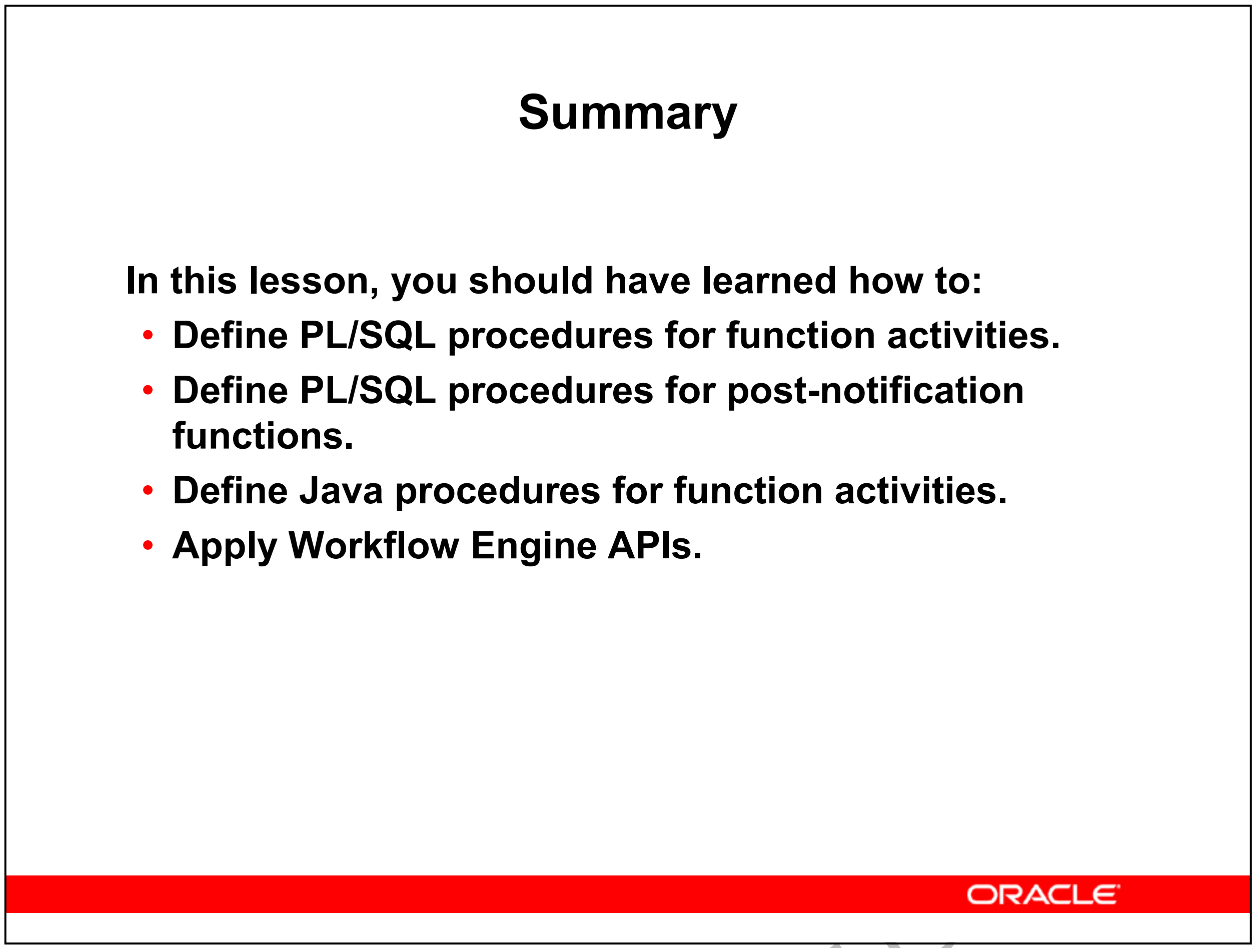
13. In the Navigator window, click the Verify button to verify your workflow.

14. From the File menu, choose Save to save your work to your workflow definition file.

15. From the File menu, choose Save As and save your item type to the class database, using the database username, password, and connect string provided by the instructor.

16. Use a web browser to connect to the Oracle Workflow home page with the URL provided by the instructor for standalone Oracle Workflow, or to a Workflow administrator responsibility provided by the instructor for Oracle Workflow embedded in Oracle E-Business Suite. Log in as a user with workflow administrator privileges.

17. Use the Launch Processes page to launch your workflow process and test your work. You can use the Notifications Worklist to view the notifications sent by the process, and use the Workflow Monitor to review the status of the process.
    - Run the process and approve the vacation proposal. The response should be successful.
    - Run the process and reject the vacation proposal notification, but do not enter any alternate dates. You should receive an error message stating that you must provide alternate dates.
    - Reject the vacation proposal notification with an alternate to date prior to the alternate from date. You should receive an error message stating that the alternate from date must be prior to the alternate to date.
    - Reject the vacation proposal with valid alternate dates. The response should be successful.
    - Run the process and choose the Reassign button in the Notification Details page to reassign the notification. Select a user name and choose the Transfer option to attempt to transfer the notification to that user. You should receive an error message stating that transfers are not allowed.
    - Choose the Delegate option to delegate the notification to the selected user. The reassignment should be successful.



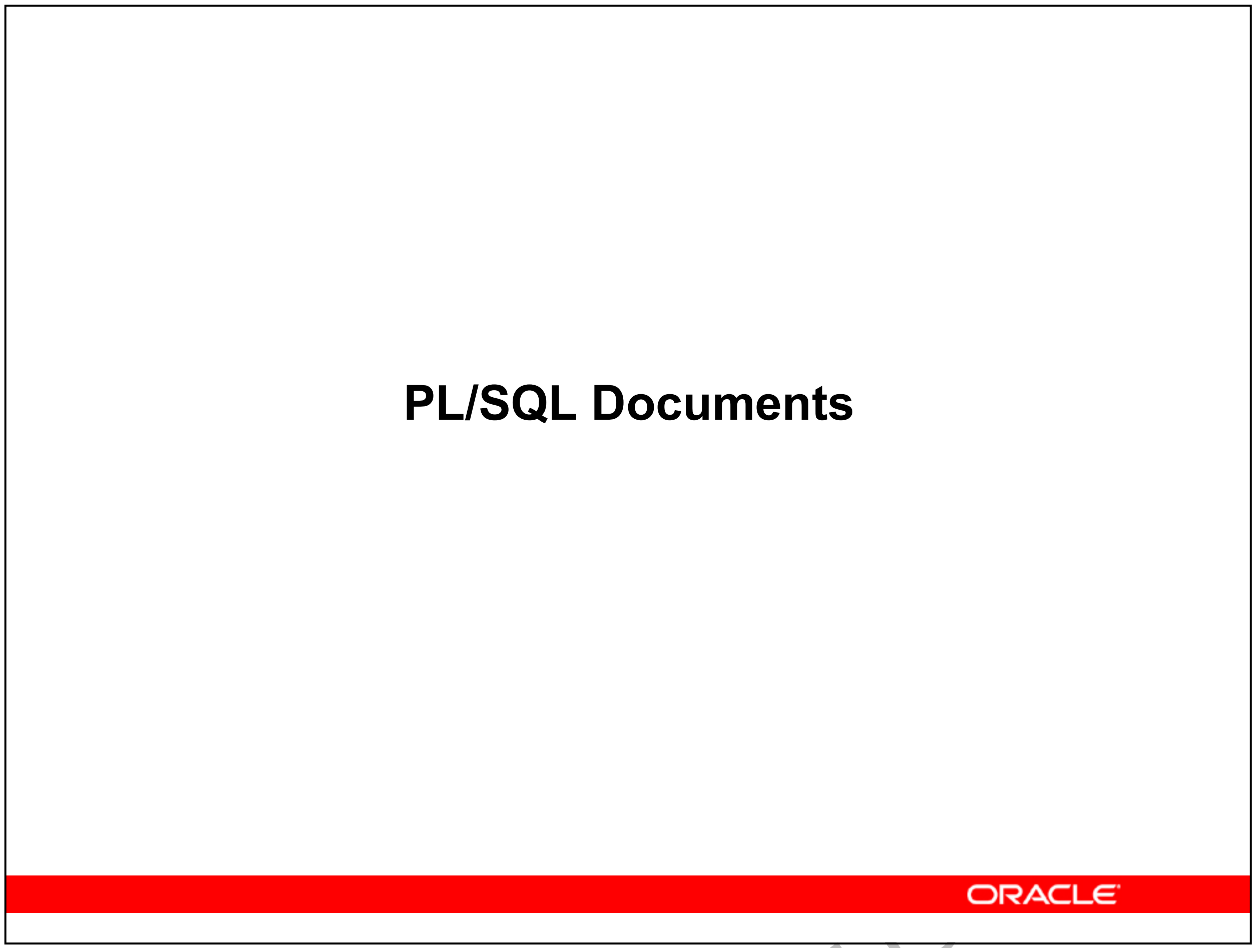
Summary

In this lesson, you should have learned how to:

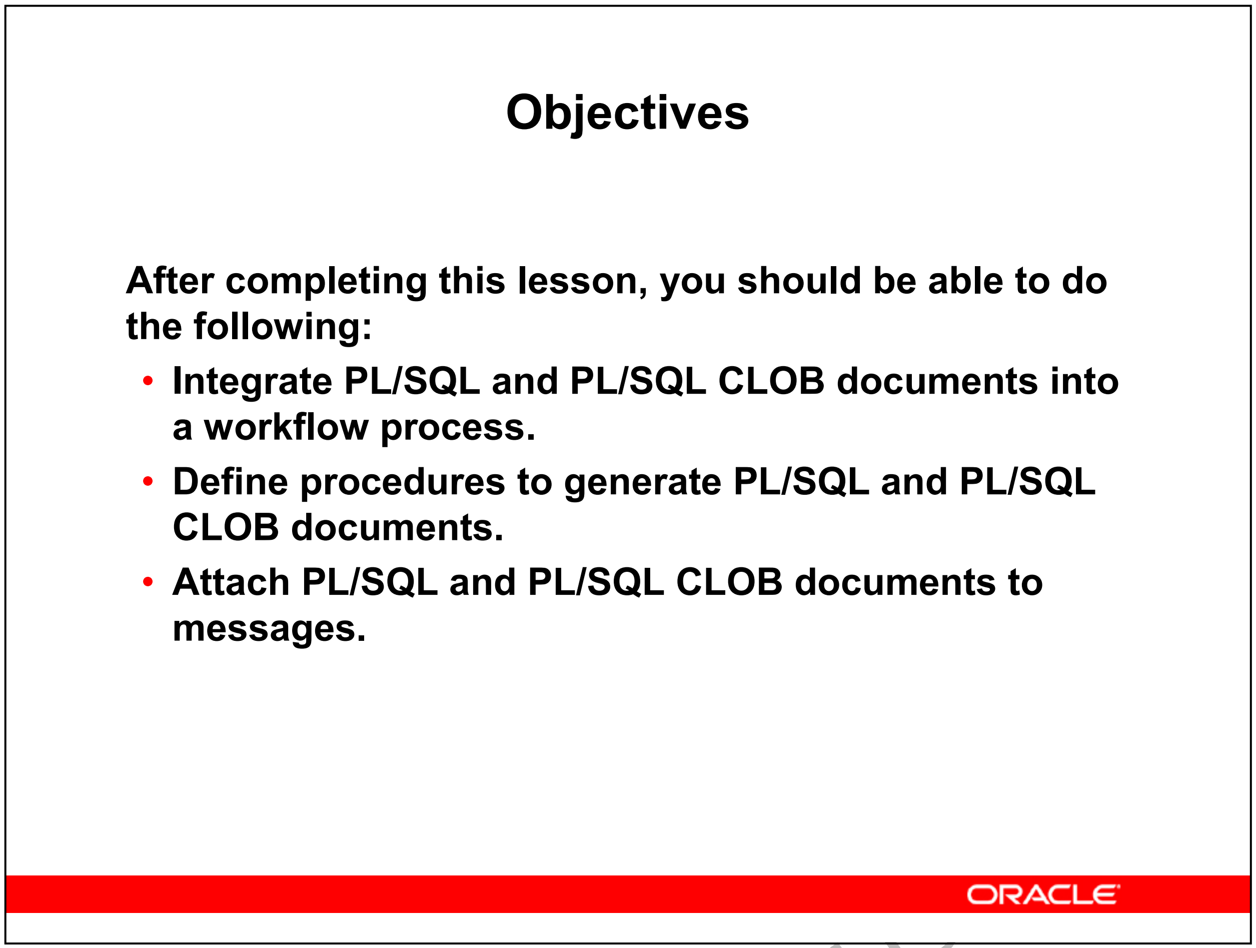
- Define PL/SQL procedures for function activities.
- Define PL/SQL procedures for post-notification functions.
- Define Java procedures for function activities.
- Apply Workflow Engine APIs.

PL/SQL Documents

Chapter 22



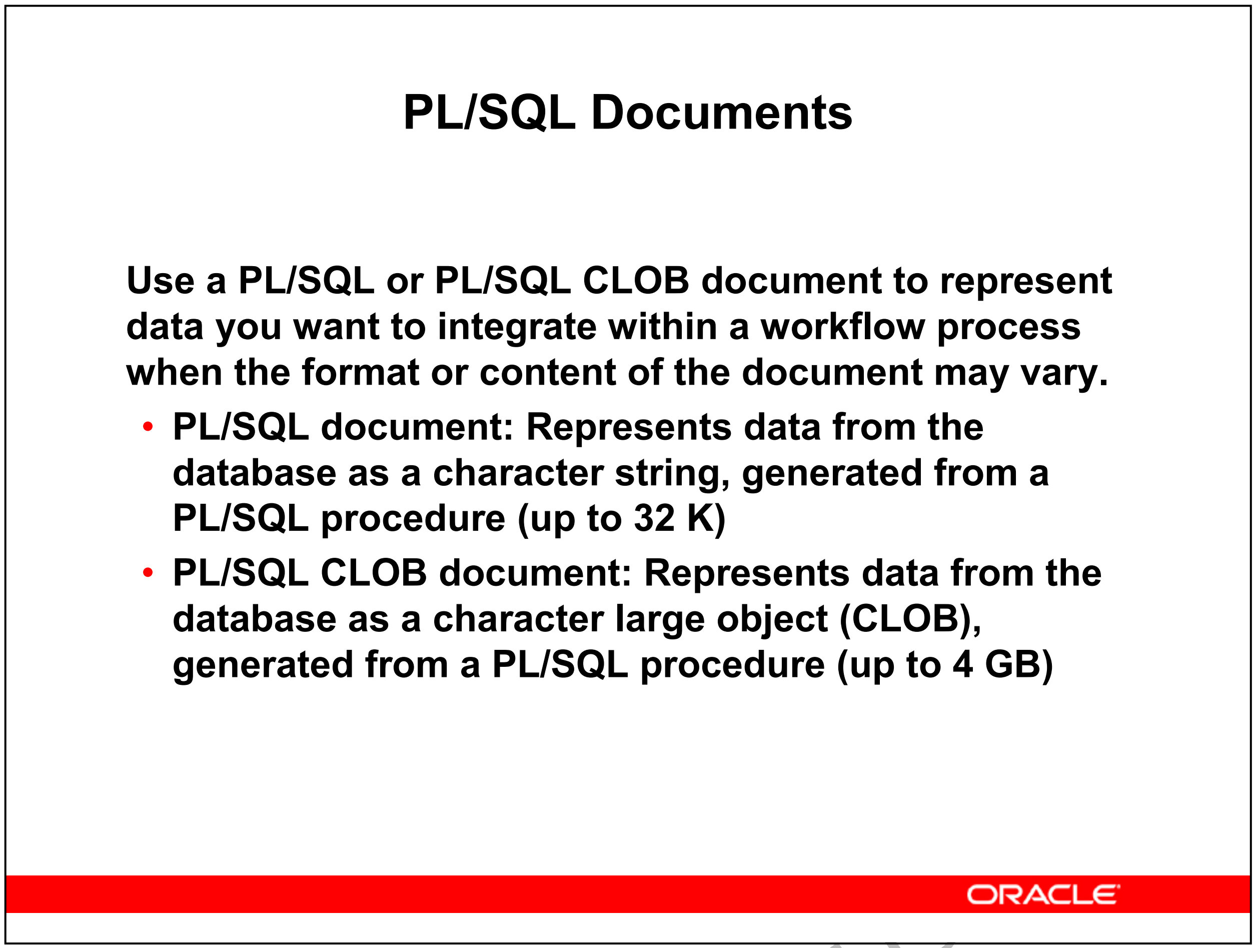
PL/SQL Documents



Objectives

After completing this lesson, you should be able to do the following:

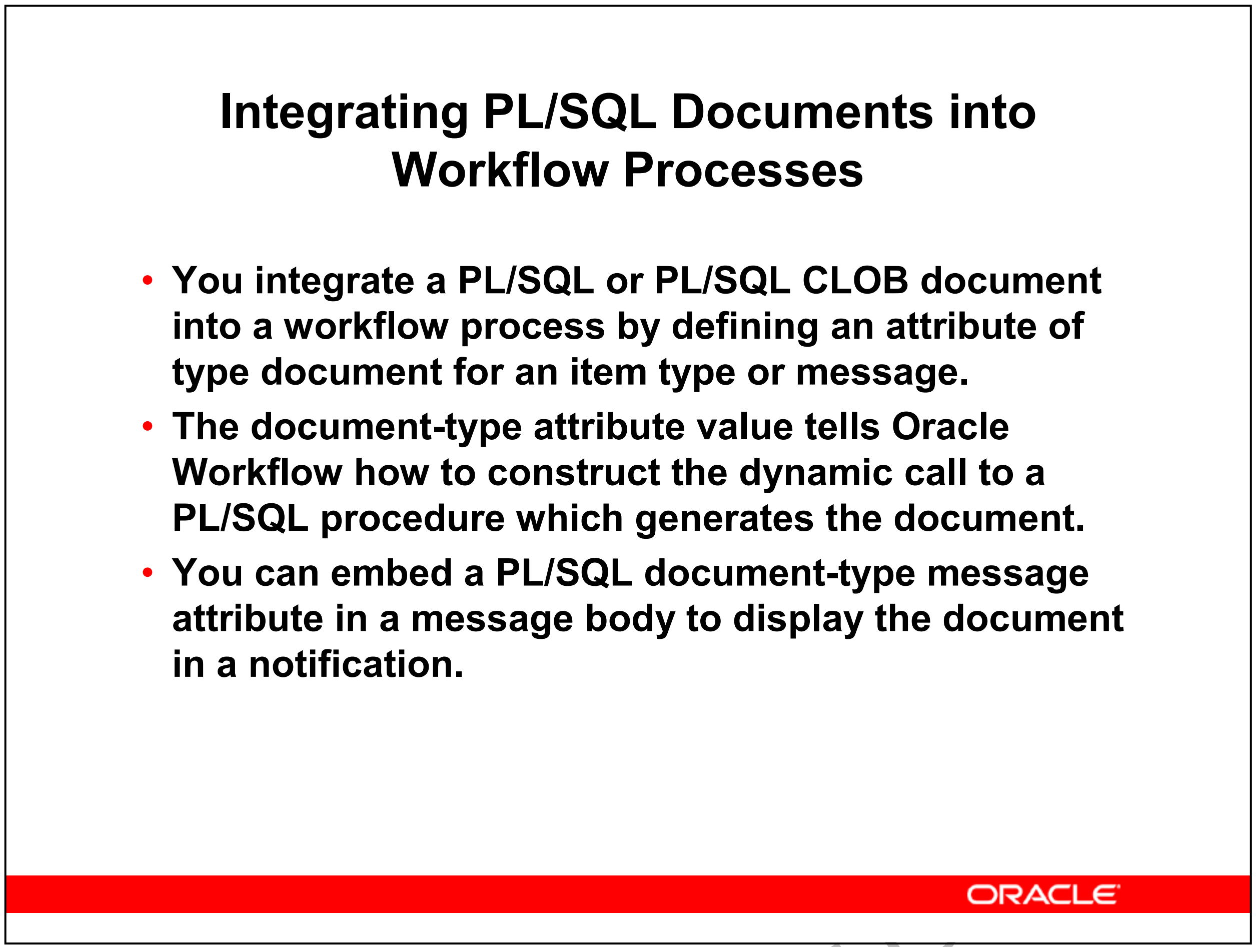
- Integrate PL/SQL and PL/SQL CLOB documents into a workflow process.
- Define procedures to generate PL/SQL and PL/SQL CLOB documents.
- Attach PL/SQL and PL/SQL CLOB documents to messages.



PL/SQL Documents

Use a PL/SQL or PL/SQL CLOB document to represent data you want to integrate within a workflow process when the format or content of the document may vary.

- PL/SQL document: Represents data from the database as a character string, generated from a PL/SQL procedure (up to 32 K)
- PL/SQL CLOB document: Represents data from the database as a character large object (CLOB), generated from a PL/SQL procedure (up to 4 GB)



Integrating PL/SQL Documents into Workflow Processes

- You integrate a PL/SQL or PL/SQL CLOB document into a workflow process by defining an attribute of type document for an item type or message.
- The document-type attribute value tells Oracle Workflow how to construct the dynamic call to a PL/SQL procedure which generates the document.
- You can embed a PL/SQL document-type message attribute in a message body to display the document in a notification.

Setting the Document Attribute Value

A PL/SQL document-type attribute should take a value of the form:

`PLSQL:<procedure>/<document_identifier>`

A PL/SQL CLOB document-type attribute should take a value of the form:

`PLSQLCLOB:<procedure>/<document_identifier>`

<procedure> should be replaced with the PL/SQL package and procedure name in the form package.procedure.

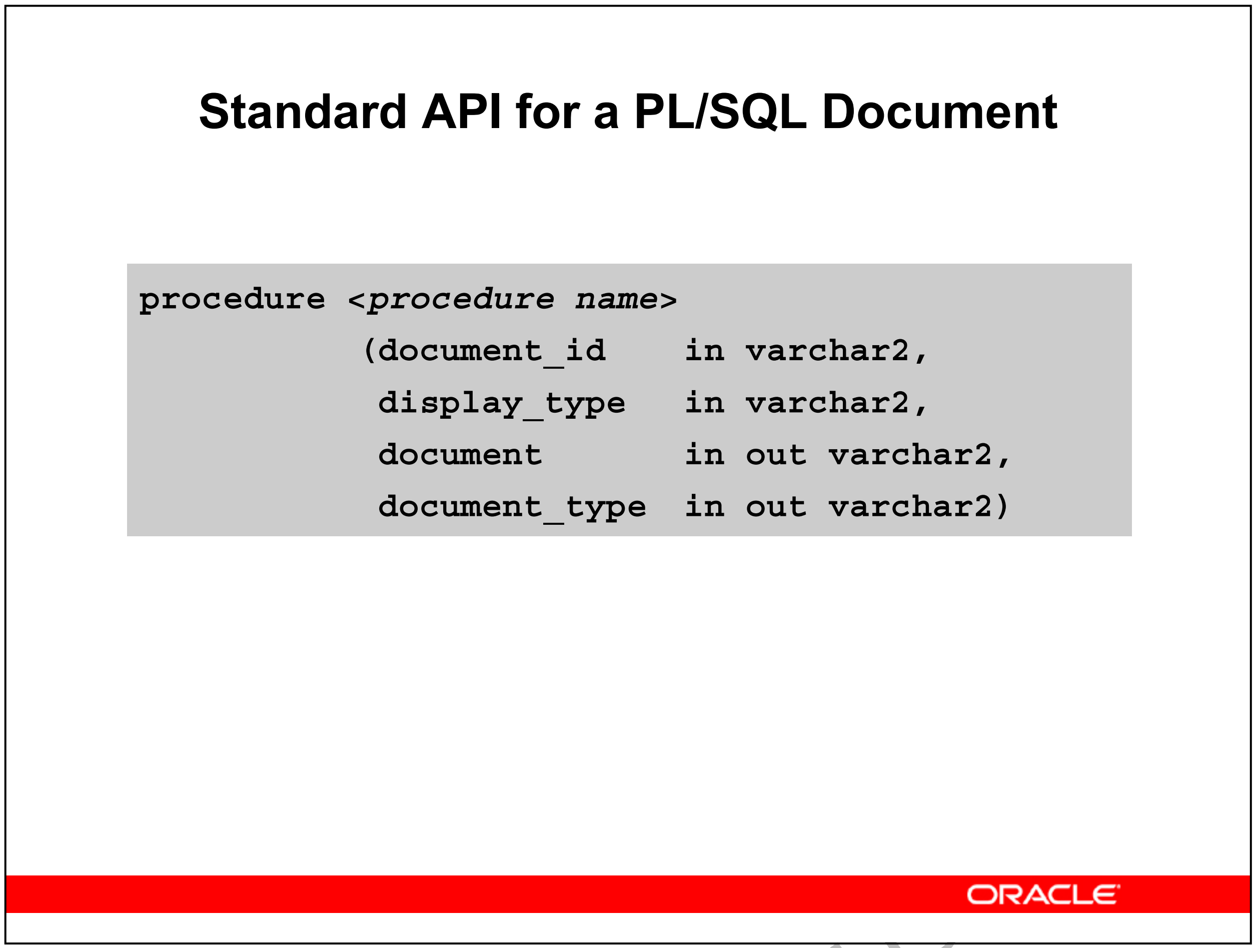
<document_identifier> should be replaced with the PL/SQL argument string that you want to pass directly to the procedure. The argument string should identify the document. For example: plsql:po_wf.show_req/2034

If you want to generate the PL/SQL argument string dynamically, create another item attribute and specify '&item_attribute' in place of the PL/SQL argument string.

For example: plsql:po_wf.show_req/&POREQ_NUMBER

Before any activity that references this other item attribute gets executed, call the WF_ENGINE.SetItemAttribute API to dynamically set the PL/SQL argument string value.

Note: The maximum length of the data that a PL/SQL document can contain is 32 kilobytes. If you expect your document to exceed 32 Kb, you should use a PL/SQL CLOB document to hold the data instead.



Standard API for a PL/SQL Document

```
procedure <procedure name>
            (document_id    in varchar2,
             display_type   in varchar2,
             document       in out varchar2,
             document_type  in out varchar2)
```

ORACLE

Standard API for a PL/SQL Document

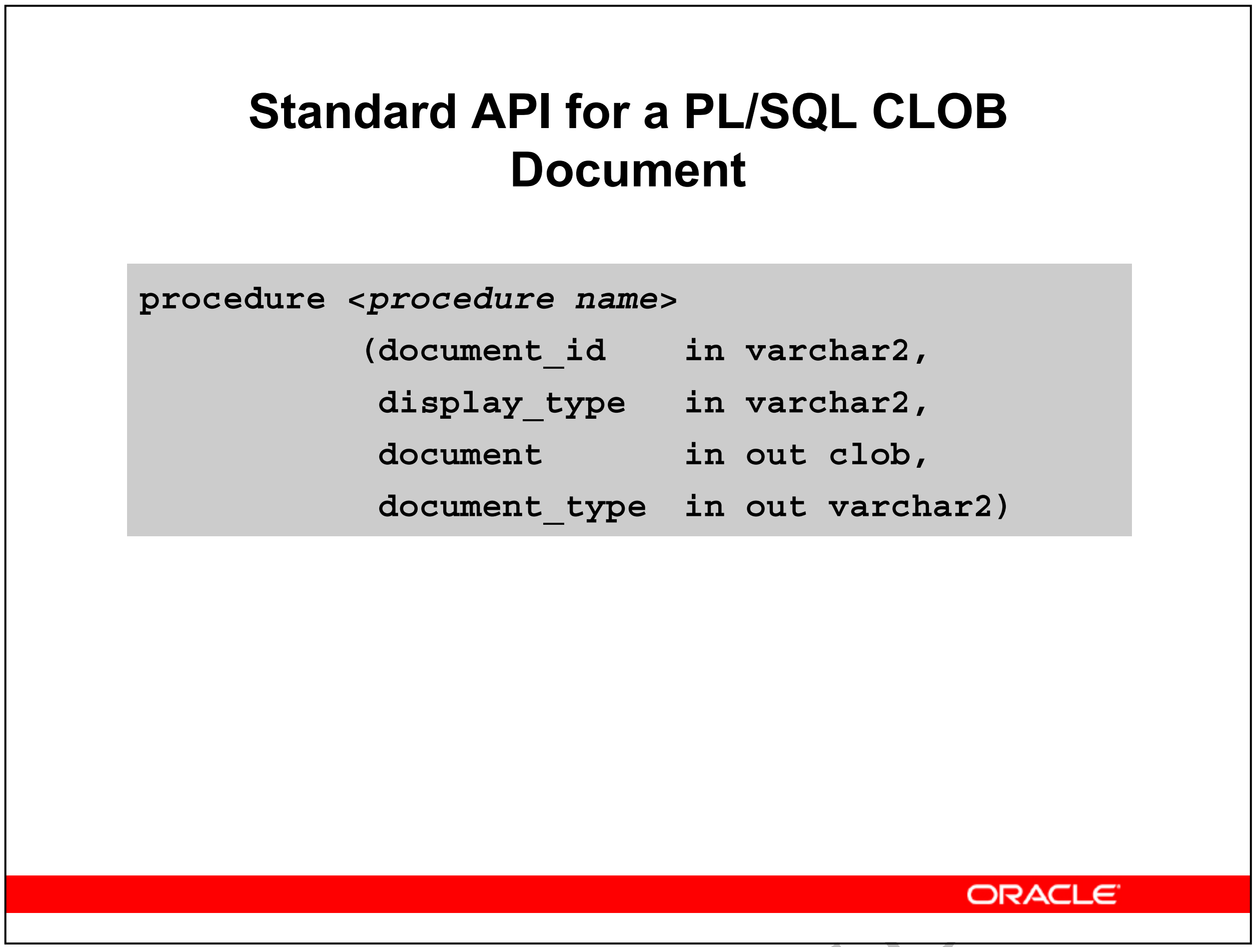
The procedure for generating a PL/SQL document must follow a standard API format. The parameters of the procedure are as follows:

- document_id: A string that uniquely identifies a document. Equivalent to the document identifier string you specify in the document-type attribute value (PLSQL:<procedure>/<document_identifier>).
- display_type: Represents how the document is displayed in the notification (also referred to as the "content type" or "requested type"):
  - text/plain: The document is embedded inside a plain text representation of the notification as viewed from an e-mail message. The entire mail message must be less than or equal to 32K.
  - text/html: The document is embedded inside an HTML representation of the notification as viewed from the Notification Details web page or the HTML attachment to an e-mail message. The entire HTML representation of the document must be less than or equal to 32K and should not include top level HTML tags like <HTML> or <BODY>, because the HTML page that the document is being inserted into already contains these tags. If the tags are included, Oracle Workflow removes them for you

when the document attribute is referenced in a message body. The procedure can alternatively generate a plain text document as the Notification System can automatically surround plain text with the appropriate HTML tags to preserve formatting.

  - ‘ ‘: The document is presented as a separate attachment to the notification. Any content type may be returned.

- document: Outbound text buffer where up to 32K of document text is returned.
- document type: Outbound text buffer where the document content type is returned. Also referred to as the returned type. If no type is supplied, then “text/plain” is assumed.



Standard API for a PL/SQL CLOB Document

```
procedure <procedure name>
          (document_id    in varchar2,
           display_type   in varchar2,
           document       in out clob,
           document_type  in out varchar2)
```

Standard API for a PL/SQL CLOB Document

The procedure for generating a PL/SQL CLOB document must follow a standard API format. The parameters of the procedure are as follows:

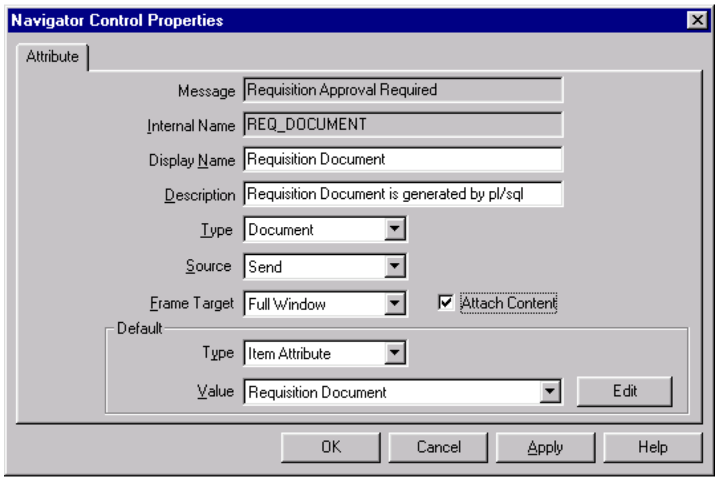
- document_id: A string that uniquely identifies a document. Equivalent to the document identifier string you specify in the document-type attribute value (PLSQLCLOB:<procedure>/<document_identifier>).
- display_type: Represents how the document is displayed in the notification (also referred to as the “content type” or “requested type”):
  - text/plain: The document is embedded inside a plain text representation of notification.
  - text/html: The document is embedded inside an HTML representation of the notification as viewed from the Notification Details web page or the HTML attachment to an e-mail message. The HTML representation of the document should not include top level HTML tags like <HTML> or <BODY>, because the HTML page that the document is being inserted into already contains these tags. If the tags are included, Oracle Workflow removes them for you when the document attribute is referenced in a message body. The procedure can alternatively generate a plain text document as the

Notification System can automatically surround plain text with the appropriate HTML tags to preserve formatting.

  - ‘ ‘: The document is presented as a separate attachment to the notification. Any content type may be returned.

- document: Outbound text buffer where the document text is returned.
- document type: Outbound text buffer where the document content type is returned. Also referred to as the returned type. If no type is supplied, then “text/plain” is assumed.

Attaching PL/SQL Documents to Messages



Message	Requisition Approval Required
Internal Name	REQ_DOCUMENT
Display Name	Requisition Document
Description	Requisition Document is generated by pl/sql
Type	Document
Source	Send
Frame Target	Full Window
Attach Content	✓
Default Type	Item Attribute
Default Value	Requisition Document

Attaching PL/SQL Documents to Messages

When defining a message attribute in the Oracle Workflow Builder, if your message attribute is a Send attribute and is of type Document, you can check Attach Content to attach the content of the attribute to the notification message. When you view your notification from the Notification Details web page, you see a document icon following the notification message body that displays the contents of the attached message attribute when you click it. If you view your notification from e-mail, the presentation of the attachment will vary depending on your e-mail notification preference setting.

Note: You can attach, as well as embed (by token substitution) PL/SQL and PL/SQL CLOB document attributes in the notification message and are not limited to one or the other.

Practice - Using a PL/SQL Document Attribute

Overview

In this practice, you will implement a PL/SQL Document attribute in the Vacation Proposal item type you created in the Creating a Workflow Process practice. The PL/SQL document will show the scheduled vacation for a vacation proposal requestor as stored in your WFVAC*XX*_VACATION_SCHEDULE table.

- Define the PL/SQL procedure for the PL/SQL document.
- Define an item attribute of type PL/SQL document.
- Define a notification that includes the PL/SQL document in its message.

Note: Because many students access the system and create objects during this course, you need a way to distinguish between the objects created by you and by your classmates. Therefore, you will be assigned a terminal number by your instructor. Use this number as a prefix wherever you see *XX* included in the name of something you are defining. In this way, you can ensure that the definitions you create are unique.

Note: In order to use the sample solution scripts provided for these practices, you must enter the internal names for all objects you define exactly as shown in the instructions. Otherwise, you must modify the sample code to reference the object names you define.

Assumptions

- The instructor will provide you with the connect string for the class database and the username and password of the Oracle Workflow database account.
- The instructor will provide you with the username and password of a user with workflow administrator privileges. The workflow administrator is defined in the Global Workflow Preferences page.
- For standalone Oracle Workflow, the instructor will provide you with the URL for the Oracle Workflow home page. The URL is <*webagent*>/wfa_html.home, where <*webagent*> is the base URL of the web agent configured for Oracle Workflow in your Web server.
- For Oracle Workflow embedded in Oracle E-Business Suite, the instructor will provide you with the name of a Workflow administrator responsibility. The username you use to log in should have this responsibility assigned to it.
- The instructor will provide you with the names of users that you can assign as the requestor and approver in the Vacation Proposal process.

Tasks

1. Define the PL/SQL procedure for the PL/SQL document.

Solution:

1. Your PL/SQL procedure will select from a table named WFVAC*XX*_VACATION_SCHEDULE. If you have not already done so, copy and edit the sample table creation script named wfvacxxc.sql. Open a copy of the sample file and replace all instances of XX with your own terminal number. Save the file and rename it by replacing *xx* with your terminal number. Then log in to SQL*Plus using the database username, password, and connect string provided by the instructor, and run the table creation script.

   This script creates a vacation schedule table named WFVAC*XX*_VACATION_SCHEDULE. The table includes the following columns:
   – REQUESTOR - varchar2(30)
   – APPROVER - varchar2(30)
   – FROM_DATE - date
   – TO_DATE - date

2. Using Wordpad, open the SQL scripts you created for the specification and body of a PL/SQL package, named wfvacxxs.sql and wfvacxxb.sql, respectively. In the WFVACXX package, create a new procedure named VACATION_SCHEDULED that creates a document displaying the vacation scheduled for the current requestor.

3. Use the standard PL/SQL Document API for the VACATION_SCHEDULED procedure.

```
procedure <procedure_name>
     (document_id   in     varchar2,
      display_type  in     varchar2,
      document      in out varchar2,
      document_type in out varchar2,)
```

4. In the VACATION_SCHEDULED procedure, use the requestor username as the value of the document_id to create a document displaying the vacation scheduled for the current requestor.
   – Create a cursor to select all vacation schedule rows for the requestor.

```
cursor vacation_schedule
     (xrequestor in varchar2) is
     select approver_username, from_date, to_date
     from WFVACXX_VACATION_SCHEDULE
     where requestor_username in
        (select name from wf_users
         where display_name = xrequestor)
     order by from_date, to_date;
```

Note: The select statement from the wf_users view is used only for illustrative purposes in this practice. When writing your own procedures, you must not write SQL statements that join to the WF_USERS, WF_ROLES, or WF_USER_ROLES views; you should only access the Workflow directory service using the directory service APIs provided in the WF_DIRECTORY package.

- Build a document text buffer. You can choose to create a document specific to the display _type (text/html, text/plain, or null), or choose to always create an html or plain text document.
- Include SQL date formatting on the from_date and to_date to improve the appearance of the date fields in your document.
- Use the Workflow API WF_DIRECTORY.GetRoleDisplayName to show the approver display name instead of the approver username in your document.
- Set the document_type based on the document you created, text/html or text/plain.

5. You can optionally copy and edit the sample package specification and body scripts instead of coding your own scripts. The sample scripts are named wfvacxxs.sql and wfvacxxb.sql, respectively. They contain all the sample PL/SQL procedures for all practices in this course. The procedure specific to this practice is WFVACXX.VACATION_SCHEDULED. Open a copy of each sample file and replace all instances of XX with your own terminal number. Save the files and rename them by replacing *xx* with your terminal number.

   Note: If you have already copied, edited, and run the complete sample package specification and body scripts containing all the sample procedures for all practices, you do not need to repeat these steps.

6. Log in to SQL*Plus using the database username, password, and connect string provided by the instructor. Run the package specification and package body scripts in that order.

2. Define an item attribute of type PL/SQL document.

   Solution:

7. Start the Oracle Workflow Builder.

8. From the File menu, choose Open to open the wfvac*XX*.wft data store you defined in the Creating a Workflow Process practice.

9. In the navigator tree, select your item type.

10. From the Edit menu, choose New > Attribute.

11. Define the following properties for the item attribute:
    - Internal Name: VACATION_SCHEDULE_DOC
    - Display Name: Vacation Schedule Document
    - Type: Document
    - Frame Target: Full Window
    - Default Value: PLSQL:WFVAC*XX*.VACATION_SCHEDULED/&REQUESTOR

Choose OK.

3. Define a notification that includes the PL/SQL document in its message.

   Solution:

   12. From the Edit menu, choose New > Message.

   13. Define a Vacation Schedule message that informs the requestor of his or her scheduled vacation dates using the PL/SQL document. To embed the contents of the PL/SQL document within the message, include the message attribute token &VACATION_SCHEDULE_DOC in the message body.

   14. Drag and drop the Requestor item attribute, the Vacation Schedule Document item attribute, and any other necessary item attributes onto the Vacation Schedule message to create the corresponding message attributes with those item attributes as their default values.

       Note: The &REQUESTOR portion of the default value for the PL/SQL document message attribute will be token replaced with the runtime value of the REQUESTOR message attribute. Even if the REQUESTOR attribute is not used in the message body, it must still be defined as a message attribute to enable the token substitution in the PL/SQL document message attribute.

   15. To attach the contents of the PL/SQL document to the message, open the property pages for the Vacation Schedule Document message attribute, select a frame target for the document, and select the Attach Content check box.

   16. From the Edit menu, choose New > Notification.

   17. Define the following properties for the notification activity:
       – Internal Name: VACATION_SCHEDULE
       – Display Name: Vacation Schedule

   18. Choose the icon NTF_INFO.ICO for the notification activity.

   19. In the Navigator window, drag and drop the Vacation Schedule message onto the Vacation Schedule notification to assign that message to the activity.

   20. Open the process diagram window for the Vacation Proposal process.

   21. Delete the transition between the Update Vacation Schedule node and the Raise Vacation Scheduled Event node.

   22. Drag and drop the Vacation Schedule notification into the process diagram, positioning it between the Update Vacation Schedule node and the Raise Vacation Scheduled Event node.

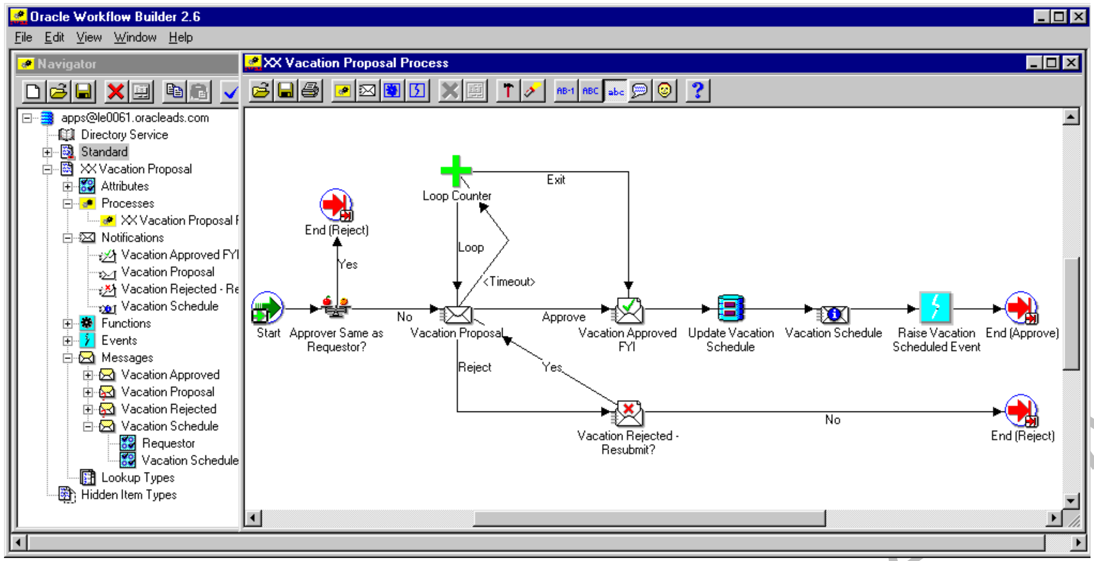
23. Draw transitions from the Update Vacation Schedule node to the Vacation Schedule node and from the Vacation Schedule node to the Raise Vacation Scheduled Event node.

24. Double-click the Vacation Schedule node and choose the Node tab. Set the performer for the node to the Requestor item attribute.

25. In the Navigator window, click the Verify button to verify your workflow.

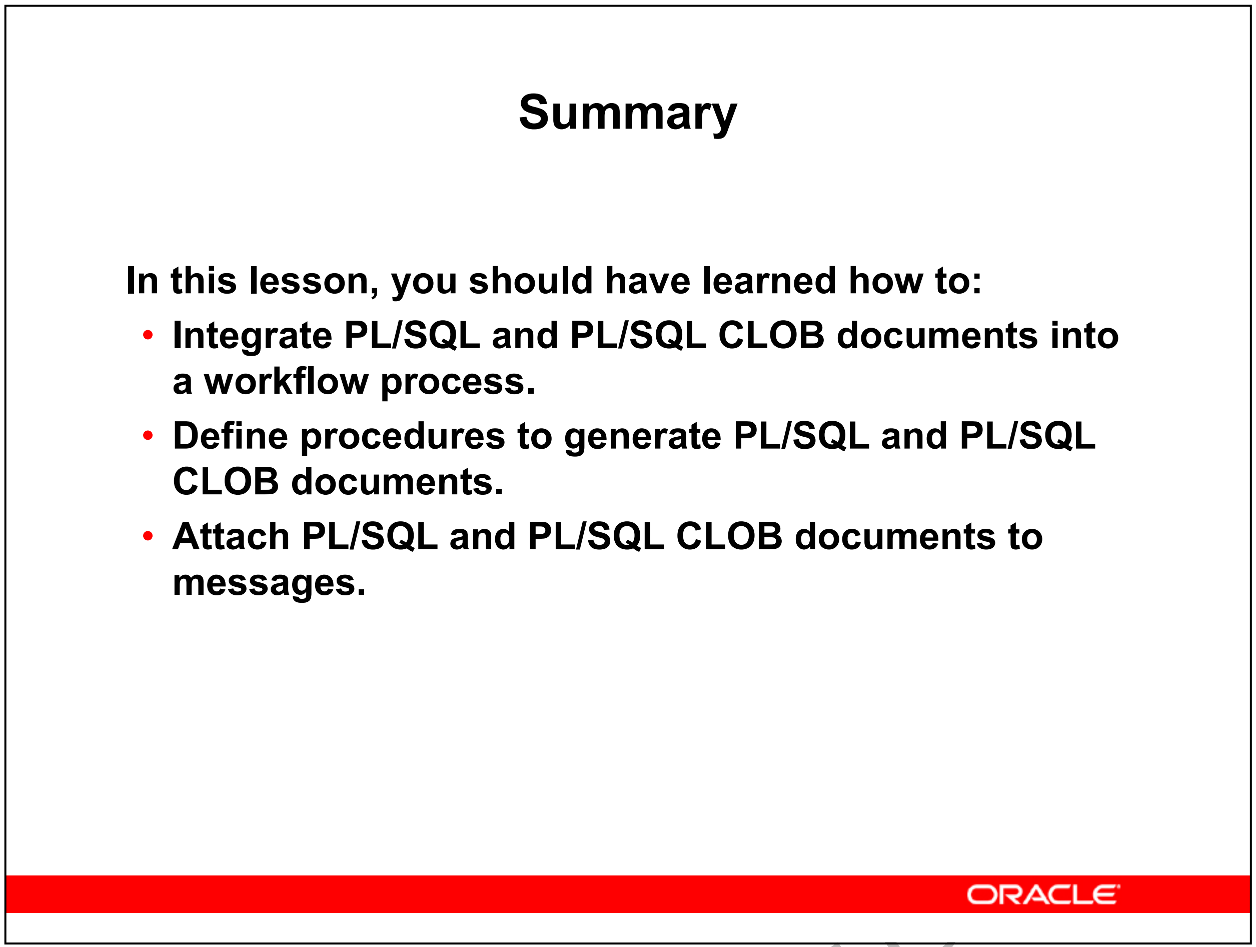
26. From the File menu, choose Save to save your work to your workflow definition file.

27. From the File menu, choose Save As and save your item type to the class database, using the database username, password, and connect string provided by the instructor.



28. Use a web browser to connect to the Oracle Workflow home page with the URL provided by the instructor for standalone Oracle Workflow, or to a Workflow administrator responsibility provided by the instructor for Oracle Workflow embedded in Oracle E-Business Suite. Log in as a user with workflow administrator privileges.

29. Use the Launch Processes page to launch your workflow process and test your work. You can use the Notifications Worklist to view the notifications sent by the process, and use the Workflow Monitor to review the status of the process.
    – Run the process and approve the vacation proposal.
    – Use the Notifications Worklist to view the Vacation Schedule notification and review the vacation schedule document. A PL/SQL document is generated with a display type of text/html when the notification is viewed through the Notification Details web page.
    – Correct any formatting or processing errors in your PL/SQL procedure, if necessary, while leaving the Notification Details web page open. Then use the reload option in your web browser to display the updated version of the Vacation Schedule notification details.



Summary

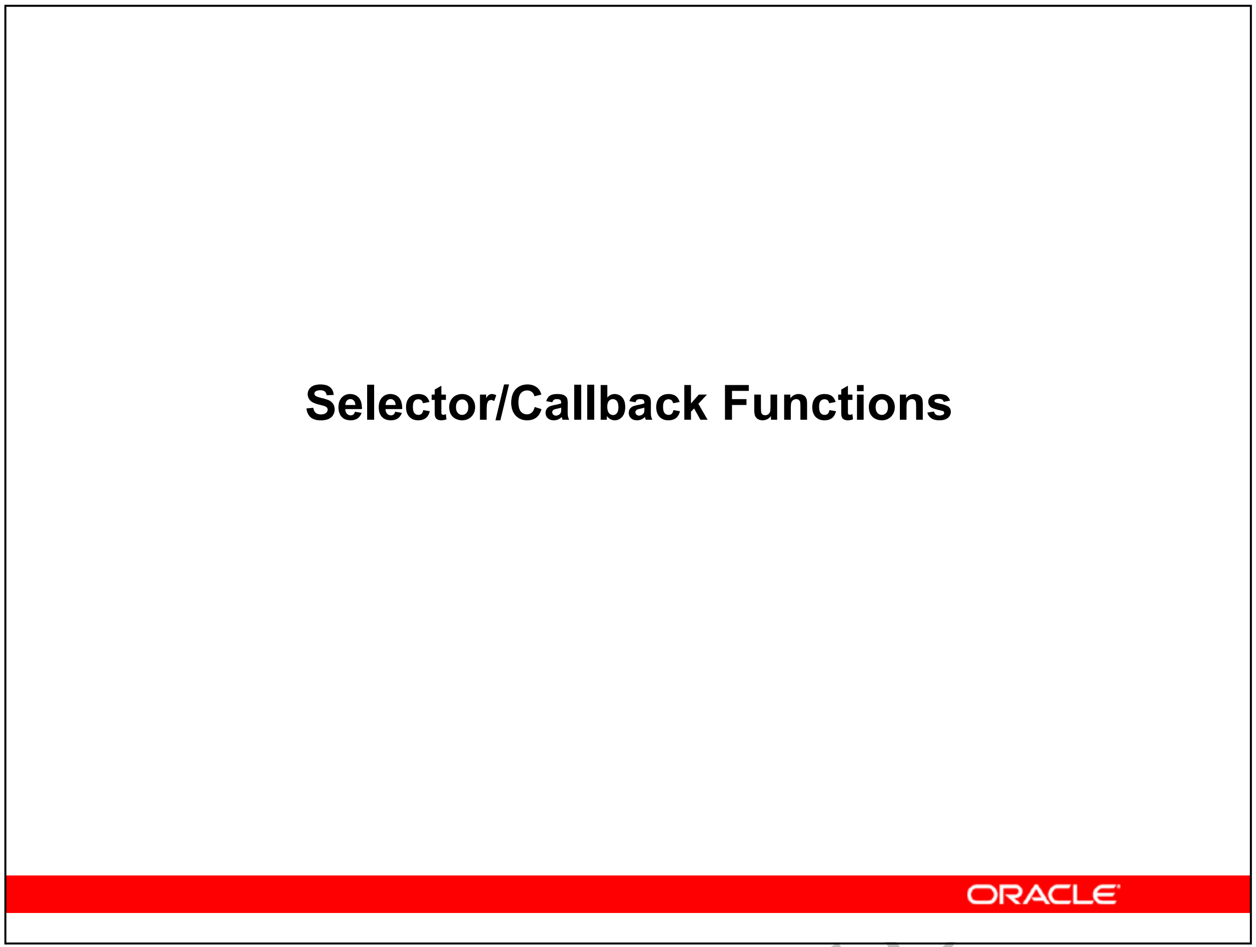
In this lesson, you should have learned how to:

- Integrate PL/SQL and PL/SQL CLOB documents into a workflow process.
- Define procedures to generate PL/SQL and PL/SQL CLOB documents.
- Attach PL/SQL and PL/SQL CLOB documents to messages.

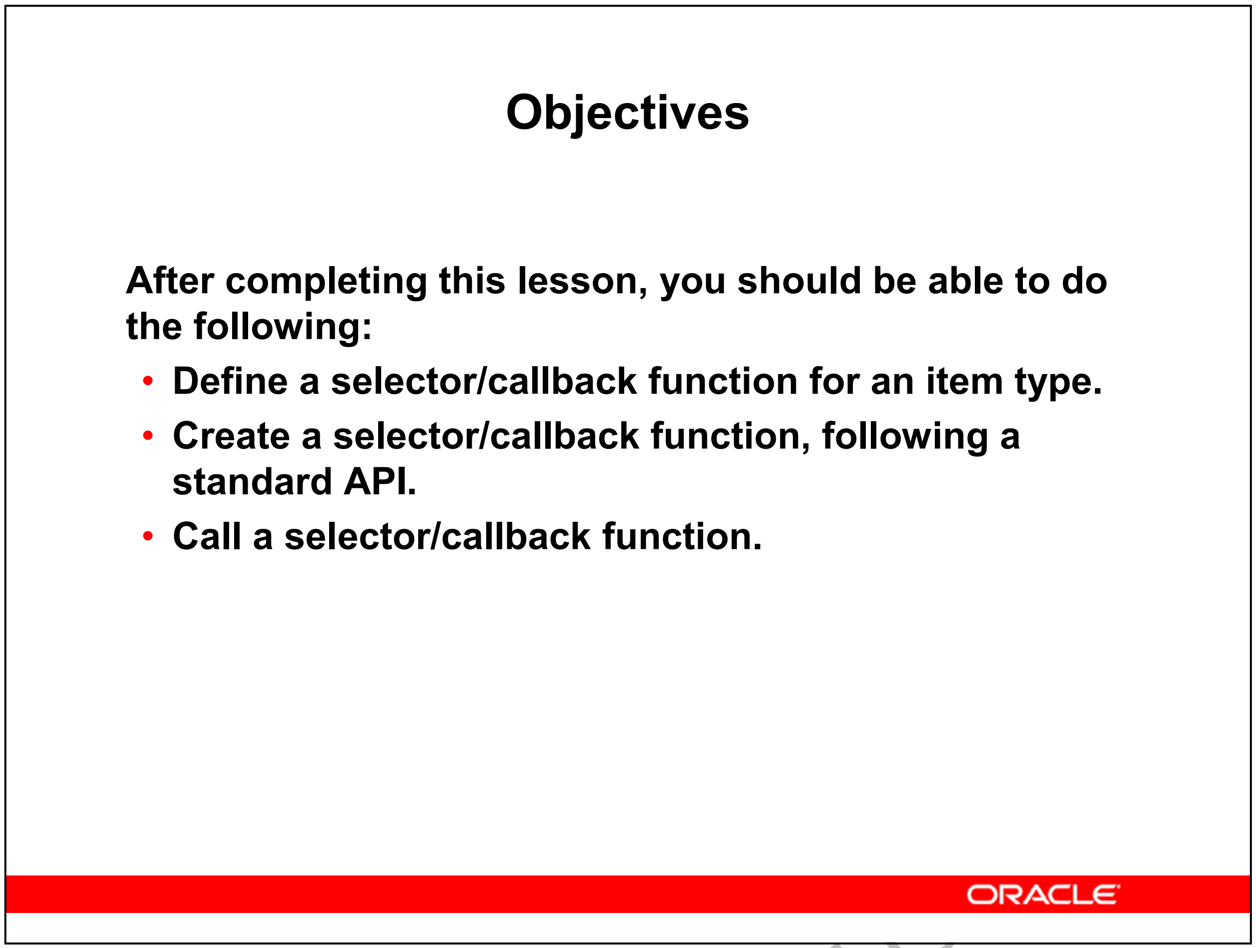
ORACLE

Selector/Callback Functions

Chapter 23



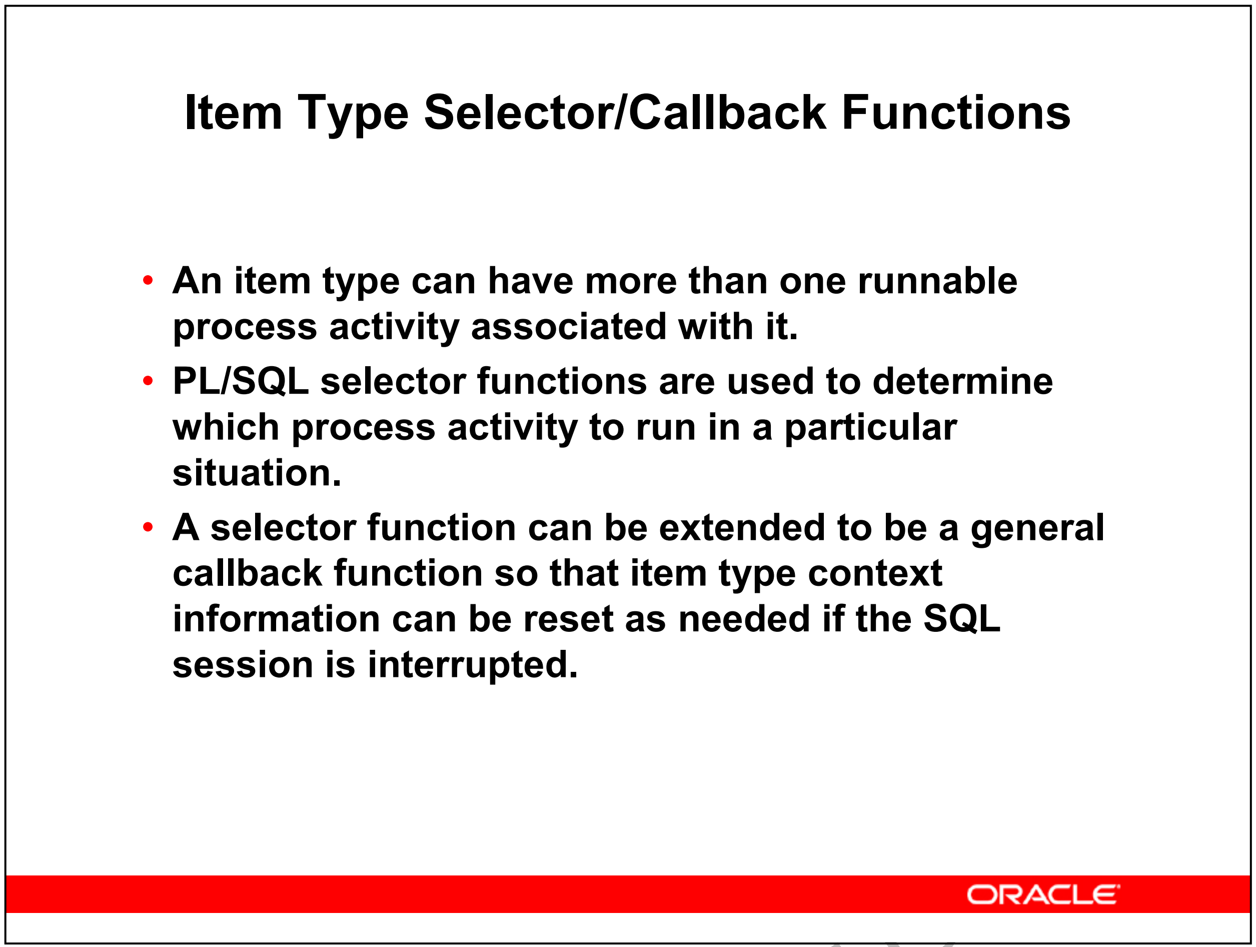
Selector/Callback Functions



Objectives

After completing this lesson, you should be able to do the following:

- Define a selector/callback function for an item type.
- Create a selector/callback function, following a standard API.
- Call a selector/callback function.



Item Type Selector/Callback Functions

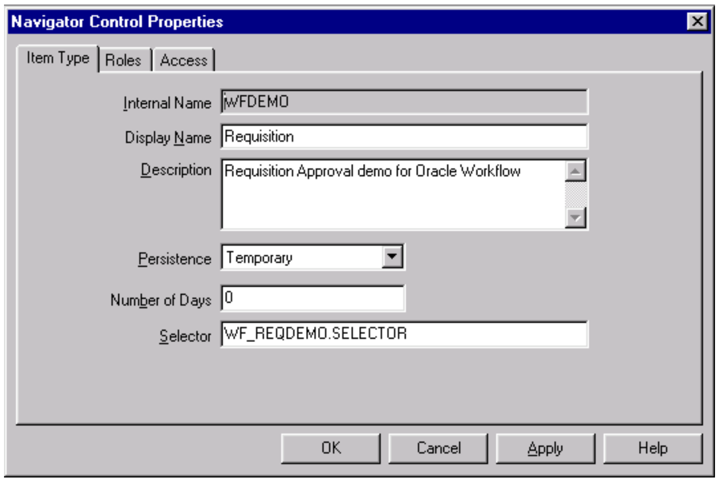
- **An item type can have more than one runnable process activity associated with it.**
- **PL/SQL selector functions are used to determine which process activity to run in a particular situation.**
- **A selector function can be extended to be a general callback function so that item type context information can be reset as needed if the SQL session is interrupted.**

Item Type Selector Functions

A selector function is a PL/SQL procedure that automatically identifies the specific process definition to execute when a workflow is initiated for a particular item type, but no process name is provided. If your item type has more than one runnable process activity associated with it, define a PL/SQL selector function that determines which process activity to run in a particular situation. For example, you may have two different requisition approval process activities associated with the same item type. The process that Oracle Workflow executes may vary depending on how and where the requisition originates. Your selector function would determine which process would be appropriate in any given situation. This functionality gives you more flexibility to determine dynamically at runtime which process to execute.

You can also extend the selector function to be a general callback function so that item type context information can be reset as needed if the SQL session is interrupted during the execution of a process. This is particularly important in the Oracle E-Business Suite scenario when you view a notification from the Notification Details web page and attempt to launch a form that is associated with the notification. Oracle Workflow calls the selector/callback function for your item type in 'TEST_CTX' mode to test the Oracle E-Business Suite context before turning the form launch over to the Oracle Application Object Library function security system. In 'TEST_CTX' mode, the selector/callback function can perform whatever logic is necessary to determine whether it is appropriate to launch the form.

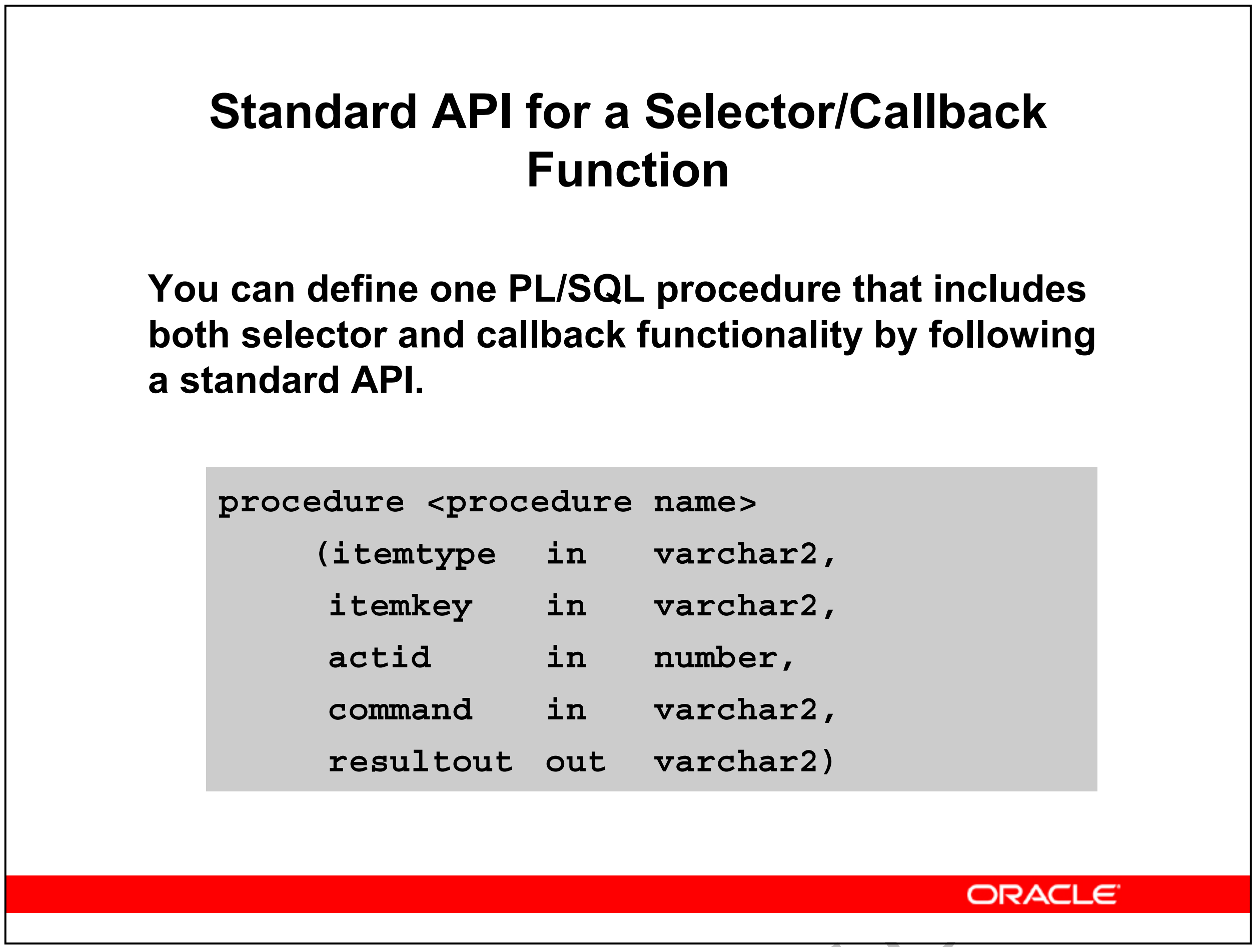
Defining a Selector/Callback Function for an Item Type



Internal Name	WFDEMO
Display Name	Requisition
Description	Requisition Approval demo for Oracle Workflow
Persistence	Temporary
Number of Days	0
Selector	WF_REQDEMO.SELECTOR

Defining a Selector/Callback Function for an Item Type

You can associate a selector function with an item type in the item type property page when you define the item type in the Oracle Workflow Builder.



Standard API for a Selector/Callback Function

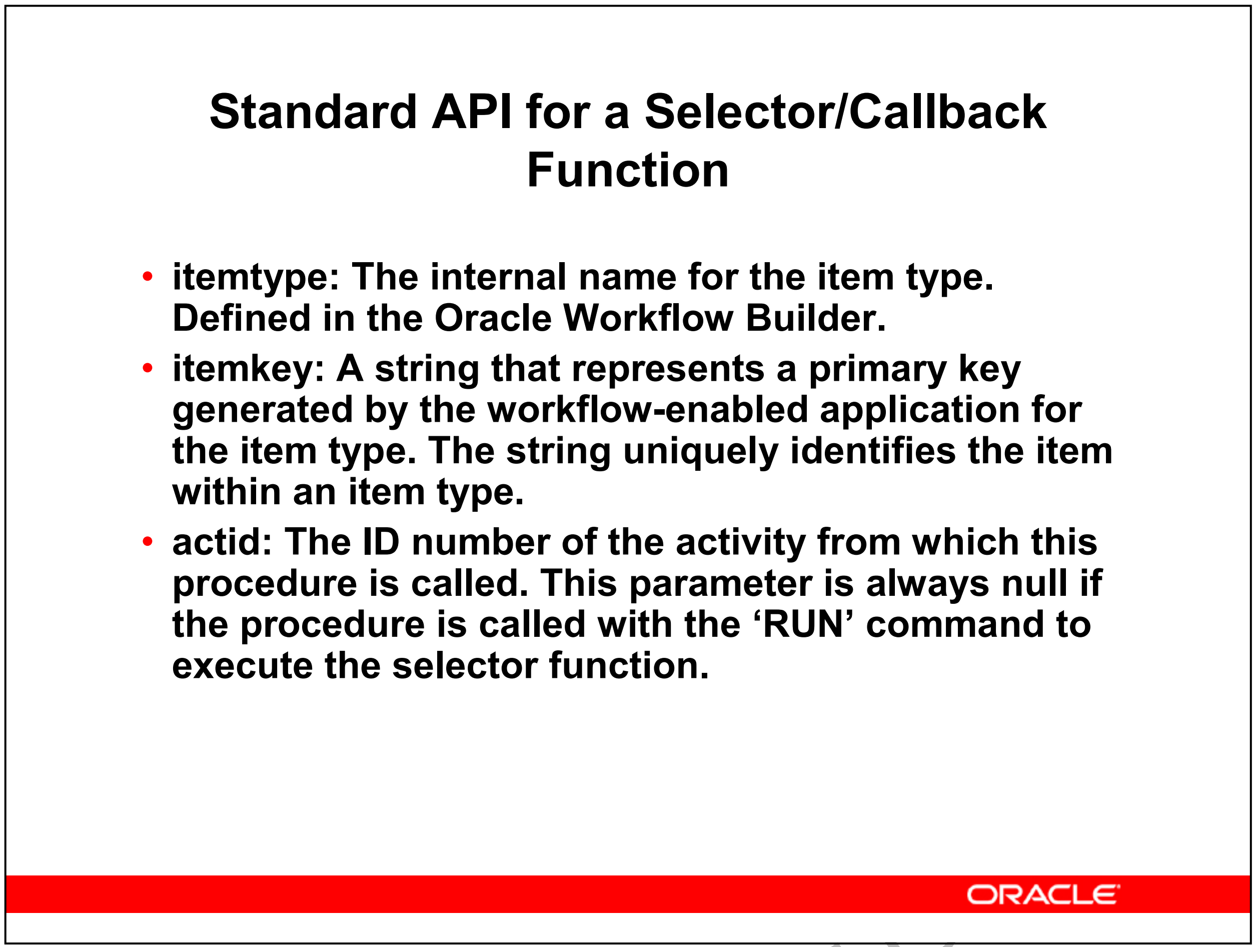
You can define one PL/SQL procedure that includes both selector and callback functionality by following a standard API.

```
procedure <procedure name>
     (itemtype   in    varchar2,
      itemkey    in    varchar2,
      actid      in    number,
      command    in    varchar2,
      resultout  out   varchar2)
```

Standard API for Selector/Callback Function

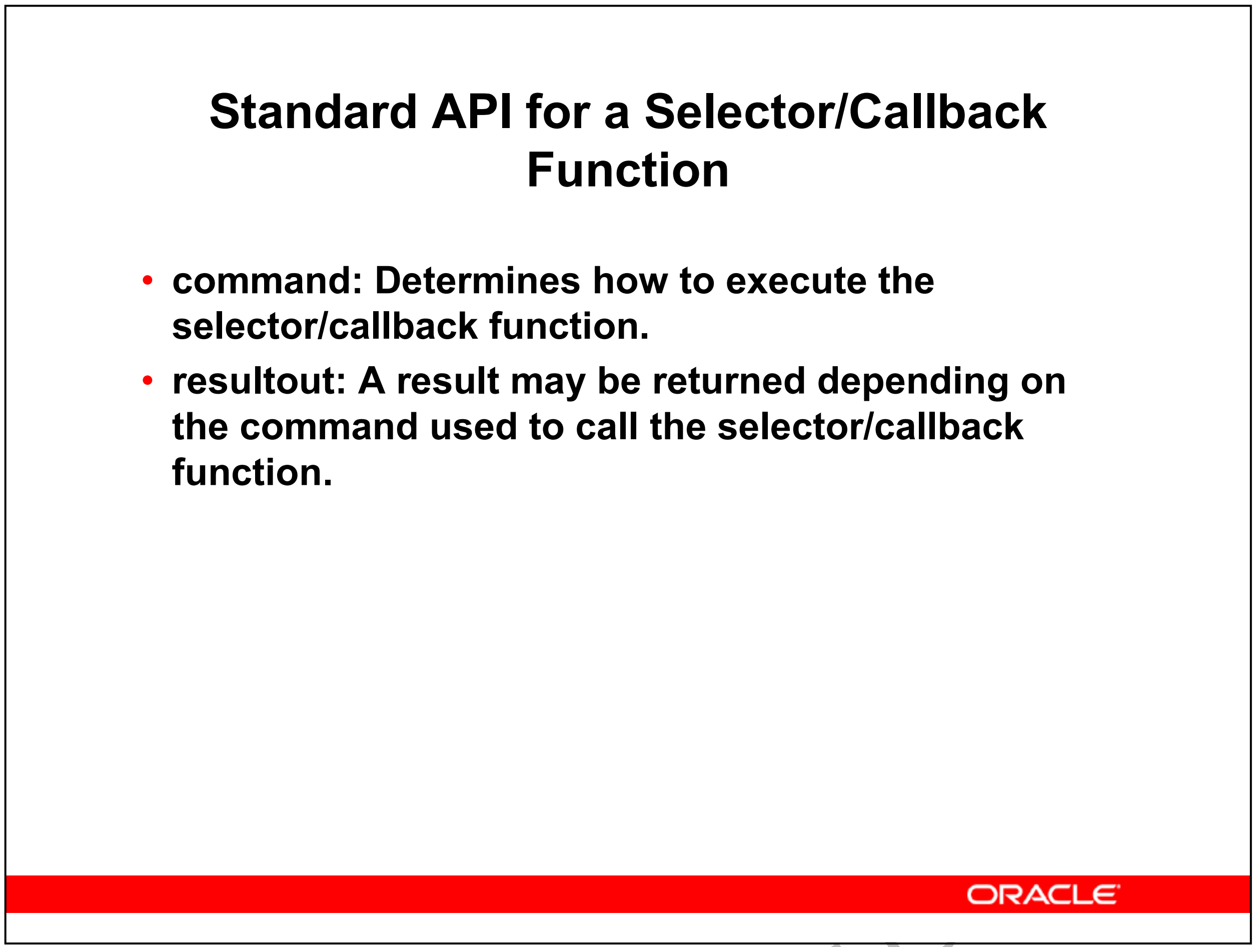
```
procedure <procedure name>(
     itemtype    in varchar2,
     itemkey     in varchar2,
     actid       in number,
     command     in varchar2,
     resultout   out varchar2) is
<local declarations>
begin
if (command = 'RUN') then
     <your RUN executable statements>
     resultout:='<Name of process to run>';
     return;
endif;
if (command = 'SET_CTX') then
     <your executable statements for establishing context information>
     resultout:=' ';
     return;
endif;
if (command = 'TEST_CTX') then
     <your executable statements for testing validity of current context
```

```
       information>
      resultout:=‘<TRUE or FALSE>’;
      return;
endif;
if (command = ‘<other commands>’) then
      resultout:=‘ ‘;
      return;
endif;
exception
when others then
      WF_CORE.CONTEXT (‘<package name>’, ‘<procedure name>’, <itemtype>,
                       <itemkey>, to_char(<actid>), <command>);
      raise;
end <procedure name>;
```



Standard API for a Selector/Callback Function

- itemtype: The internal name for the item type. Defined in the Oracle Workflow Builder.
- itemkey: A string that represents a primary key generated by the workflow-enabled application for the item type. The string uniquely identifies the item within an item type.
- actid: The ID number of the activity from which this procedure is called. This parameter is always null if the procedure is called with the ‘RUN’ command to execute the selector function.



Standard API for a Selector/Callback Function

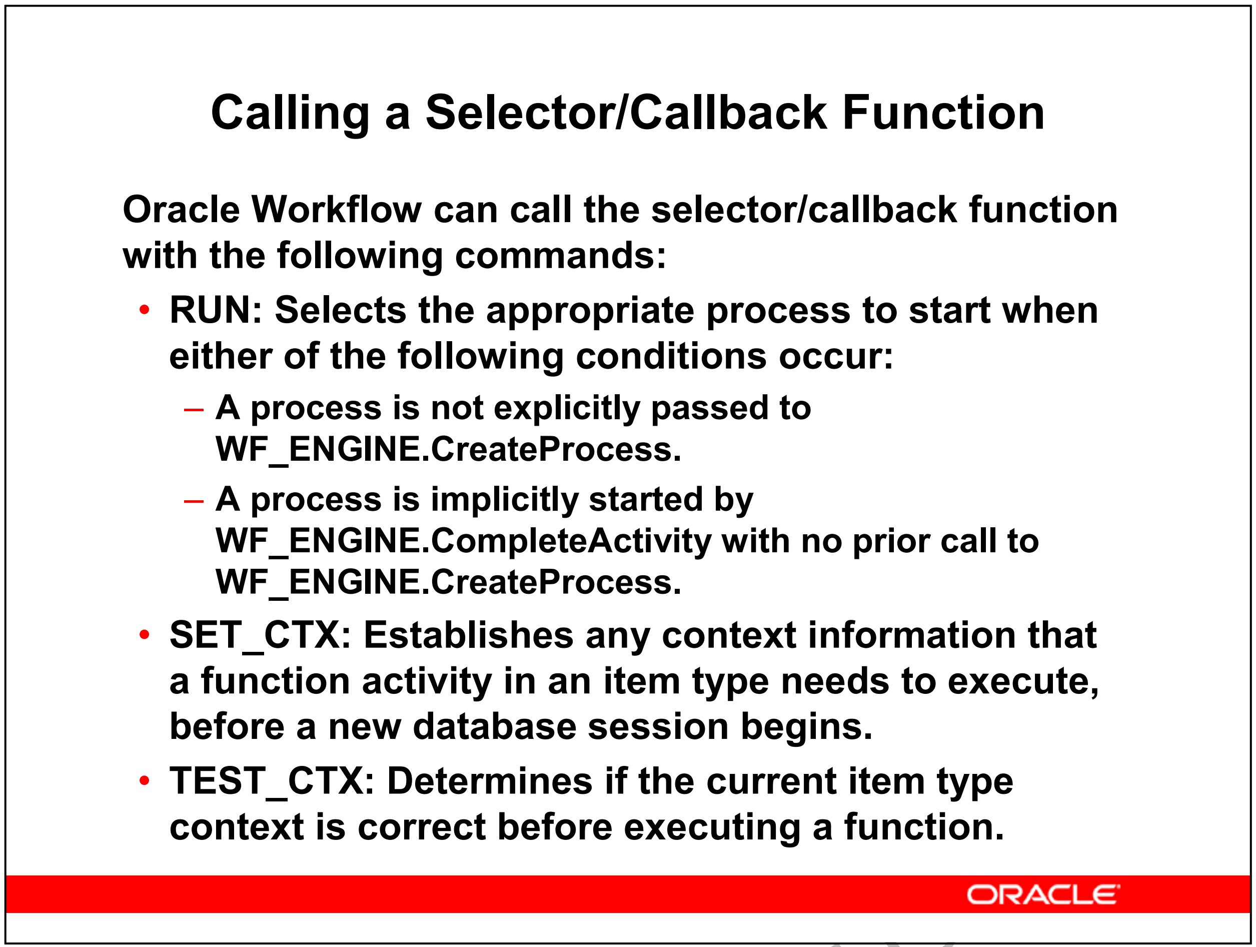
- command: Determines how to execute the selector/callback function.
- resultout: A result may be returned depending on the command used to call the selector/callback function.

Standard API for a Selector/Callback Function

The different commands for a selector/callback function require different return values.

- If the function is called with ‘RUN’, the name of the process to run must be returned through the resultout parameter.
- If the function is called with ‘SET_CTX’, then no return value is expected.
- If the function is called with ‘TEST_CTX’, then the code must return ‘TRUE’ if the context is correct or ‘FALSE’ if the context is incorrect.
- If any other value is returned, Oracle Workflow assumes the command is not implemented by the callback.

Note: Other commands may be added in the future.



Calling a Selector/Callback Function

Oracle Workflow can call the selector/callback function with the following commands:

- **RUN: Selects the appropriate process to start when either of the following conditions occur:**
  - **A process is not explicitly passed to WF_ENGINE.CreateProcess.**
  - **A process is implicitly started by WF_ENGINE.CompleteActivity with no prior call to WF_ENGINE.CreateProcess.**
- **SET_CTX: Establishes any context information that a function activity in an item type needs to execute, before a new database session begins.**
- **TEST_CTX: Determines if the current item type context is correct before executing a function.**

ORACLE

Calling a Selector/Callback Function

TEST_CTX also determines whether a form can be launched with the current item type context information just before the Notification Details web page launches a reference form.

Note: Launching a form from the Notification Details web page is only available in the version of Oracle Workflow embedded in Oracle E-Business Suite.

Practice - Defining a Selector Function

Overview

In this practice, you will define a selector function to determine which process to run in the Vacation Proposal item type you created in the Creating a Workflow Process practice.

- Define a new process within the Vacation Proposal item type.
- Define a selector function for the item type.
- Define the PL/SQL procedure for the selector function.

Note: Because many students access the system and create objects during this course, you need a way to distinguish between the objects created by you and by your classmates. Therefore, you will be assigned a terminal number by your instructor. Use this number as a prefix wherever you see *XX* included in the name of something you are defining. In this way, you can ensure that the definitions you create are unique.

Note: In order to use the sample solution scripts provided for these practices, you must enter the internal names for all objects you define exactly as shown in the instructions. Otherwise, you must modify the sample code to reference the object names you define.

Assumptions

- The instructor will provide you with the connect string for the class database and the username and password of the Oracle Workflow database account.
- The instructor will provide you with the username and password of a user with workflow administrator privileges. The workflow administrator is defined in the Global Workflow Preferences page.
- For standalone Oracle Workflow, the instructor will provide you with the URL for the Oracle Workflow home page. The URL is <*webagent*>/wfa_html.home, where <*webagent*> is the base URL of the web agent configured for Oracle Workflow in your Web server.
- For Oracle Workflow embedded in Oracle E-Business Suite, the instructor will provide you with the name of a Workflow administrator responsibility. The username you use to log in should have this responsibility assigned to it.
- The instructor will provide you with the names of users that you can assign as the requestor and approver in the Vacation Proposal process.

Tasks

1. Define a new process within the Vacation Proposal item type.

Solution:

1. Start the Oracle Workflow Builder.

2. From the File menu, choose Open to open the wfvac*XX*.wft data store you defined in the Creating a Workflow Process practice.

3. In the navigator tree, select and expand your WFVAC*XX* item type.

4. In the navigator tree, drag and drop the Vacation Proposal process onto the WFVAC*XX* item type to create a copy of the process. The property pages for the copied process open automatically so that you can enter new internal and display names for the new process.
   - Internal Name: WFVAC*XX*_ALTERNATE_PROCESS
   - Display Name: *XX* Alternate Vacation Proposal

5. Open the process diagram window for the Alternate Vacation Proposal process.

6. Delete the Loop Counter node. The transitions to and from the node are automatically deleted as well

7. Create a timeout transition from the Vacation Proposal notification back to itself. To do so, select the Vacation Proposal node and hold down the right mouse button. Drag the cursor away from the notification and then back to the notification, and then release the right mouse button. Select <Timeout> from the transition results menu.

2. Define a selector function for the item type.

Solution:

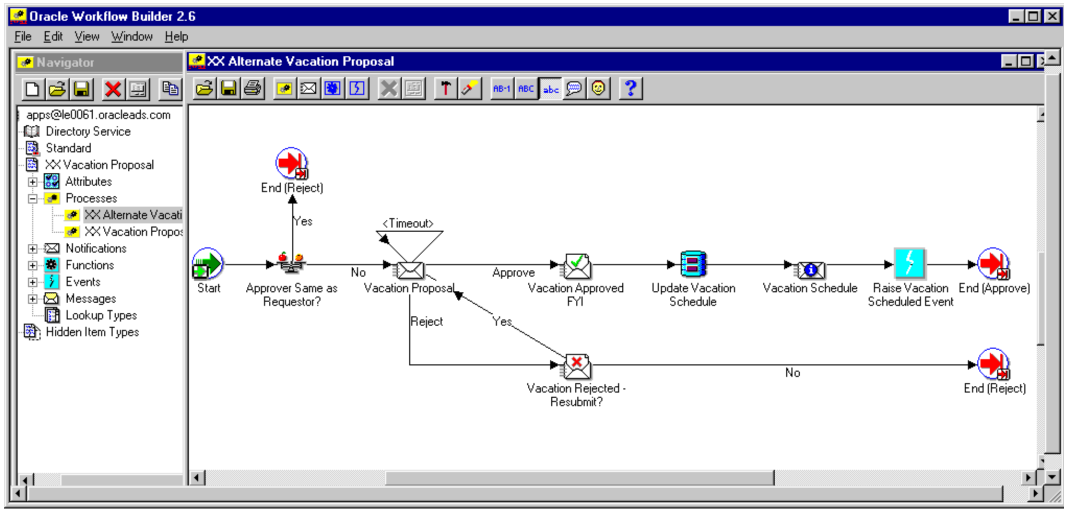
8. In the navigator tree, select your item type.

9. Open the property pages for the item type. In the Selector field, enter WFVAC*XX*.SELECTOR as the selector function for the item type.

10. In the Navigator window, click the Verify button to verify your workflow.

11. From the File menu, choose Save to save your work to your workflow definition file.

12. From the File menu, choose Save As and save your item type to the class database, using the database username, password, and connect string provided by the instructor.



3. Define the PL/SQL procedure for the selector function.

Solution:

13. Using Wordpad, open the SQL scripts you created for the specification and body of a PL/SQL package, named wfvac*xx*s.sql and wfvac*xx*b.sql, respectively. In the WFVAC*XX* package, create a new procedure named SELECTOR that determines which process to run in the Vacation Proposal item type.

14. Use the standard Selector Function API for the VACATION_SCHEDULED procedure.

```
procedure <procedure_name>
     (itemtype  in     varchar2,
      itemkey   in     varchar2,
      actid     in     number,
      command   in     varchar2,
      resultout in out varchar2)
```

15. In the SELECTOR procedure, include logic for the RUN command.
    – Evaluate the value passed in the item key.
    – If the uppercase value of the first four characters is CNTR, set resultout to 'WFVAC*XX*_PROCESS' (the internal name of the process with the Loop Counter timeout implementation).
    – If the uppercase value of the first four characters is SELF, set resultout to 'WFVAC*XX*_ALTERNATE_PROCESS' (the internal name of the process with the self-looping timeout implementation).
    – If the uppercase value of the first four characters is neither CNTR nor SELF, either raise an error or set resultout to the internal name for one of the processes as the default process.

    Note: This example is contrived for class purposes only and does not reflect a practical implementation of selector function logic. The selector function is generally expected to use the item key as the primary key to retrieve supporting application data. That application data would then be used to determine which process is appropriate to run.

It is not possible to use the value of an item attribute in the selector function, because at the time the selector function is called by the Workflow Engine, the item attributes for the work item have not yet been created.

16. You can also copy and edit the sample package specification and body scripts instead of coding your own scripts. The sample scripts are named wfvacxxs.sql and wfvacxxb.sql, respectively. They contain all the sample PL/SQL procedures for all practices in this course. The procedure specific to this practice is WFVACXX.SELECTOR. Open a copy of each sample file and replace all instances of XX with your own terminal number. Save the files and rename them by replacing *xx* with your terminal number.

    Note: If you have already copied, edited, and run the complete sample package specification and body scripts containing all the sample procedures for all practices, you do not need to repeat these steps.

17. Log in to SQL*Plus using the database username, password, and connect string provided by the instructor. Run the package specification and package body scripts in that order.

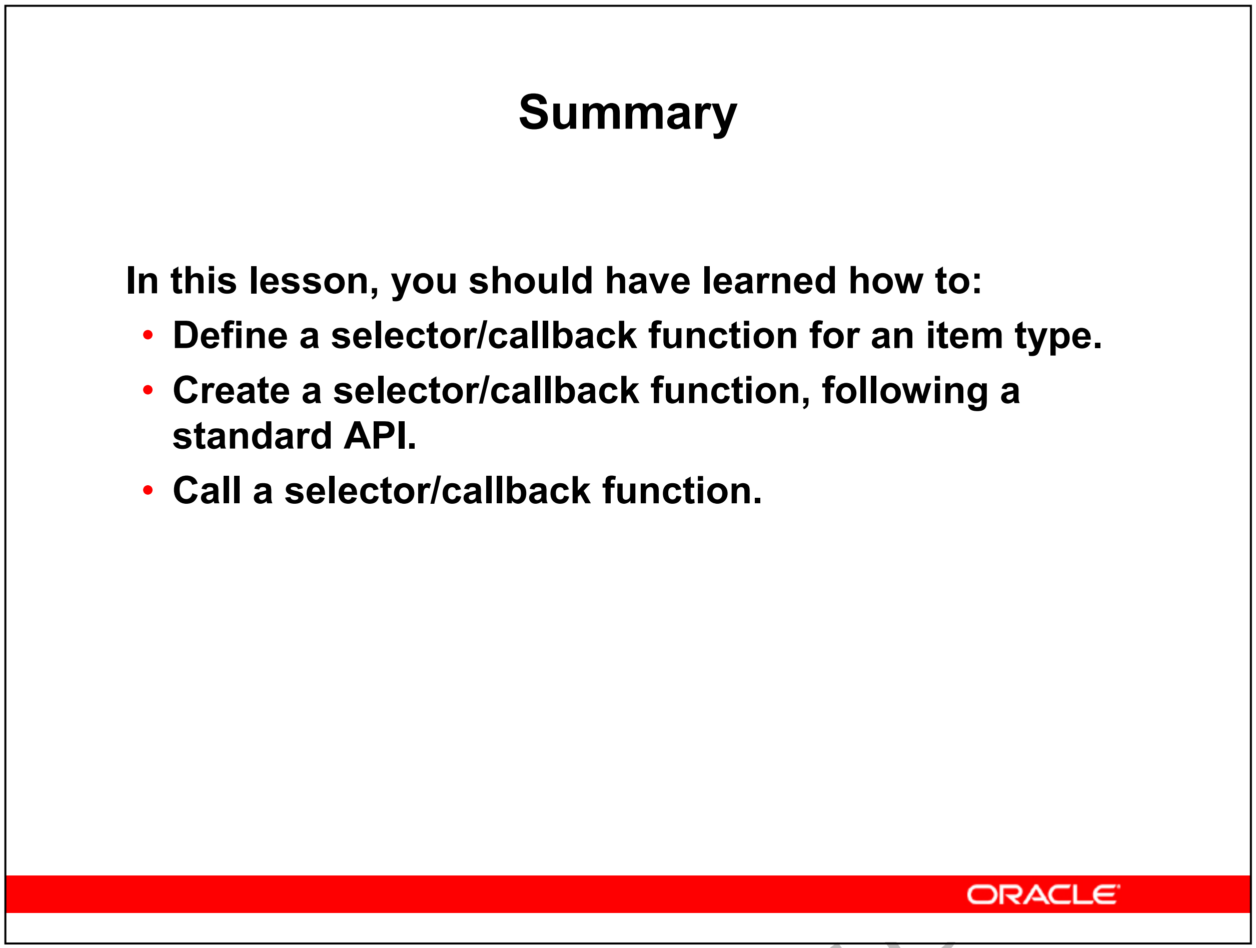
18. Use a web browser to connect to the Oracle Workflow home page with the URL provided by the instructor for standalone Oracle Workflow, or to a Workflow administrator responsibility provided by the instructor for Oracle Workflow embedded in Oracle E-Business Suite. Log in as a user with workflow administrator privileges

19. Use the Launch Processes page to launch your workflow process and test your work.
    - In the Launch Processes page, select your item type.
    - In the Initiate Workflow page, enter an item key that begins with CNTR, and select the blank value in the Process Name field to make the Workflow Engine run the selector function for the item type to determine which process to run. The Workflow Engine will run the selector function only if the process parameter is not passed in the call to create the new work item. Also enter a user key, requestor, approver, from date, to date, and event key, and click OK to launch the process.
    - Navigate back to the Launch Processes page, and select your item type.
    - In the Initiate Workflow page, enter an item key that begins with SELF, and select the blank value in the Process Name field. Also enter a user key, requestor, approver, from date, to date, and event key, and click OK to launch the process.
    - Navigate back to the Launch Processes page, and select your item type.
    - In the Initiate Workflow page, enter an item key that begins neither with CNTR nor with SELF, and select the blank value in the Process Name field. Also enter a user key, requestor, approver, from date, to date, and event key, and click OK to launch the process.

    Note: The selector function provided in the sample solution package raises an error if the item key begins neither with CNTR nor with SELF, so if you are using the sample solution, the expected behavior when you attempt to launch the process in this case is an error message.

20. You can use the Notifications Worklist to view the notifications sent by the processes, and use the Workflow Monitor to review the status of the processes. For each process you launched, verify that the expected process was run. The process that was run appears in the following Workflow Monitor pages:
    - Activities List, in the header and Activity and Parent Activity columns
    - Diagram, in the process title and the diagram itself
    - Process List, in the Process Name column

21. For extra practice, you can optionally create another process within your item type that implements timeout processing in which you transition to a reminder notification when the Vacation Proposal notification times out. Add a timeout loop for the reminder notification as well. Then modify your selector function to run this new process when the upper case value of the first four characters of the item key is RMND, and test your changes.



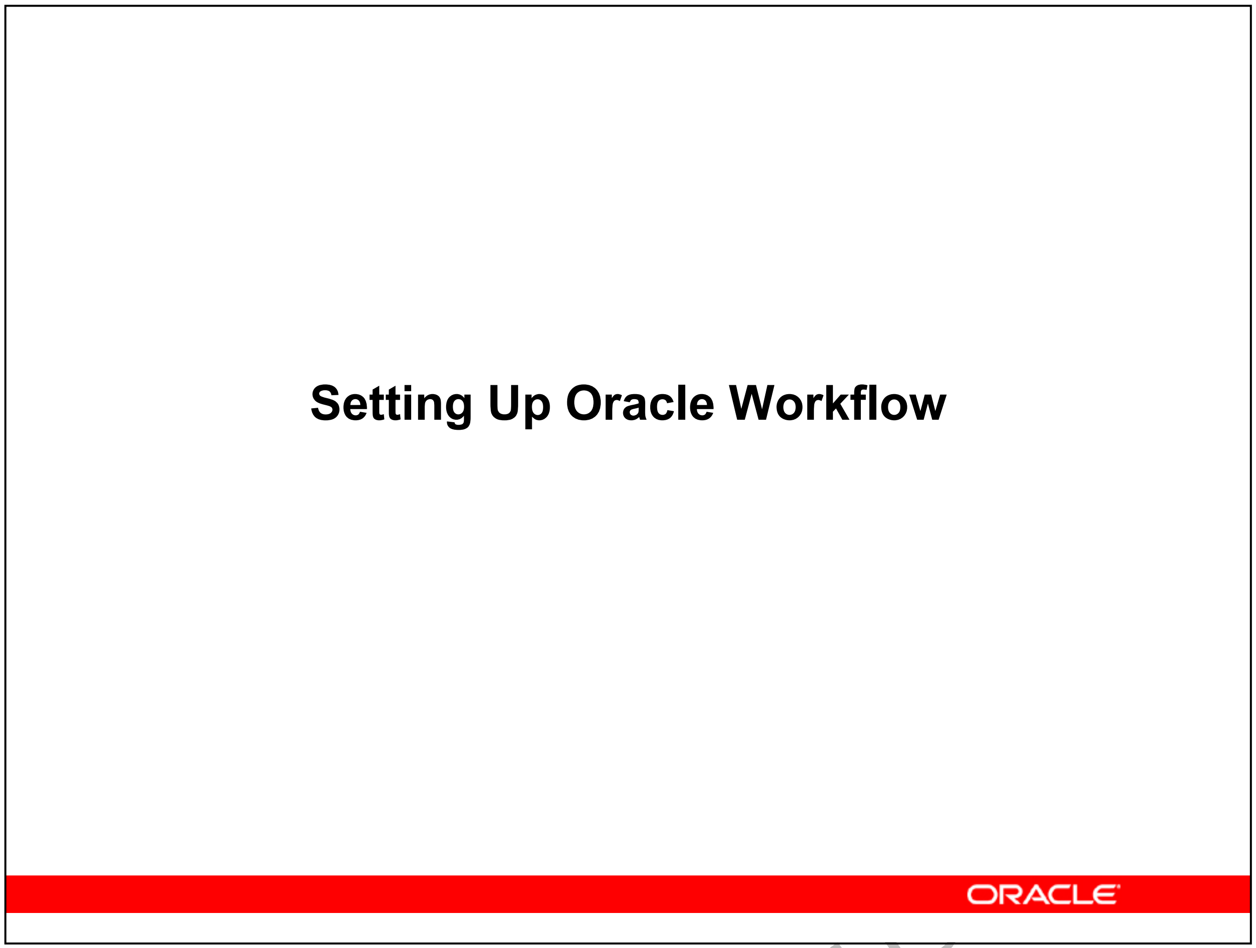
Summary

In this lesson, you should have learned how to:

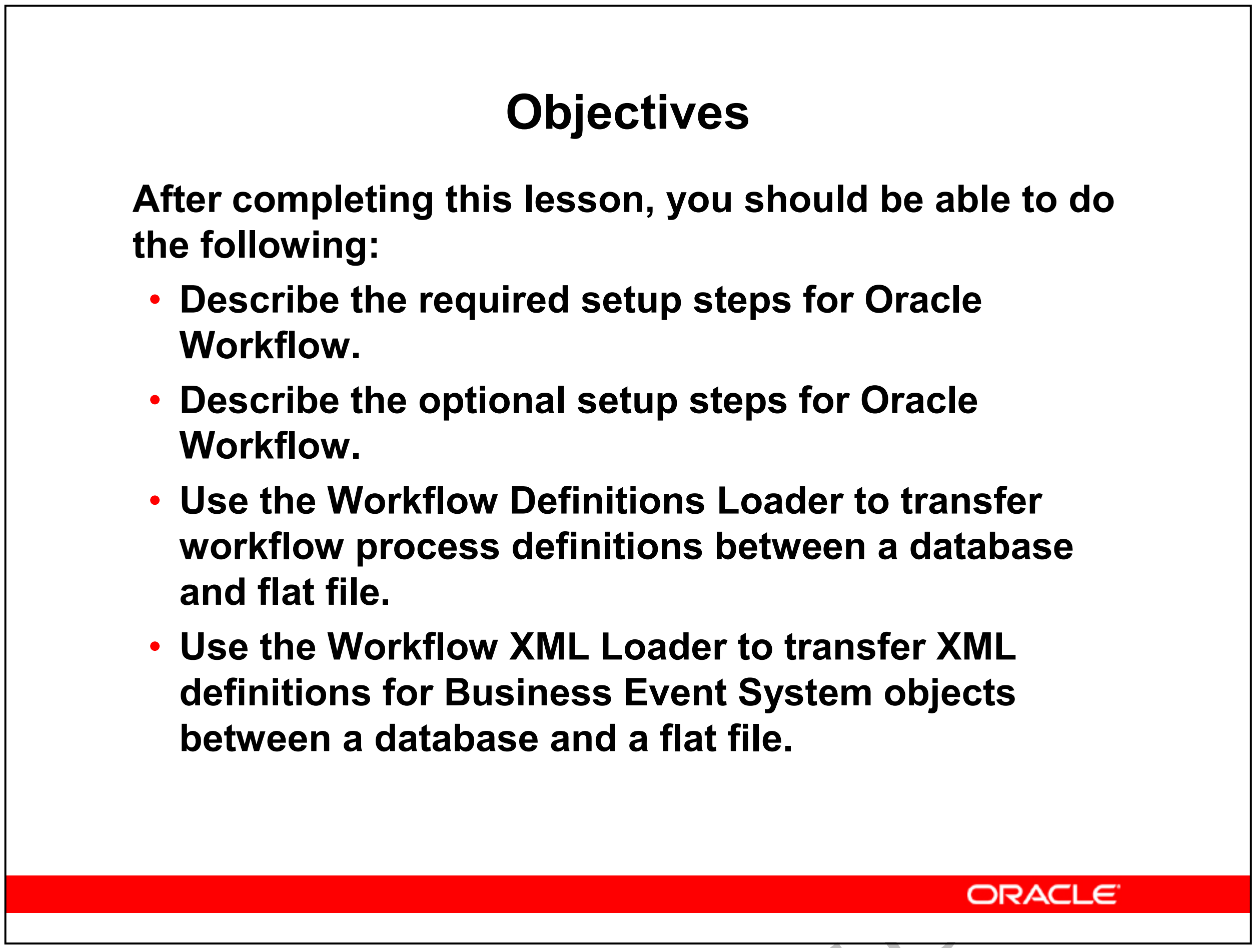
- **Define a selector/callback function for an item type.**
- **Create a selector/callback function, following a standard API.**
- **Call a selector/callback function.**

Setting Up Oracle Workflow

Chapter 24



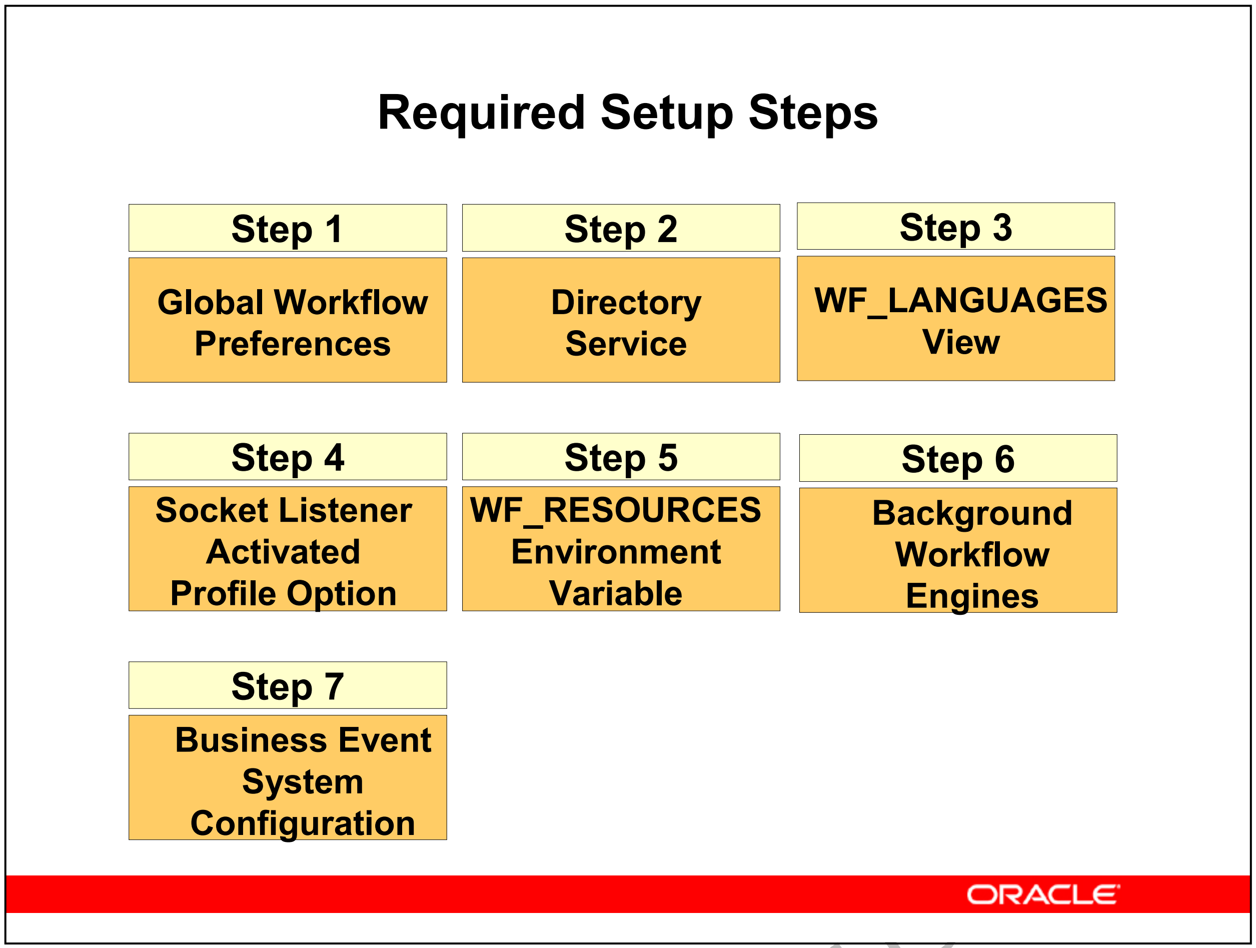
Setting Up Oracle Workflow



Objectives

After completing this lesson, you should be able to do the following:

- Describe the required setup steps for Oracle Workflow.
- Describe the optional setup steps for Oracle Workflow.
- Use the Workflow Definitions Loader to transfer workflow process definitions between a database and flat file.
- Use the Workflow XML Loader to transfer XML definitions for Business Event System objects between a database and a flat file.



Required Setup Steps

Step 1
Global Workflow Preferences

Step 2
Directory Service

Step 3
WF_LANGUAGES View

Step 4
Socket Listener Activated Profile Option

Step 5
WF_RESOURCES Environment Variable

Step 6
Background Workflow Engines

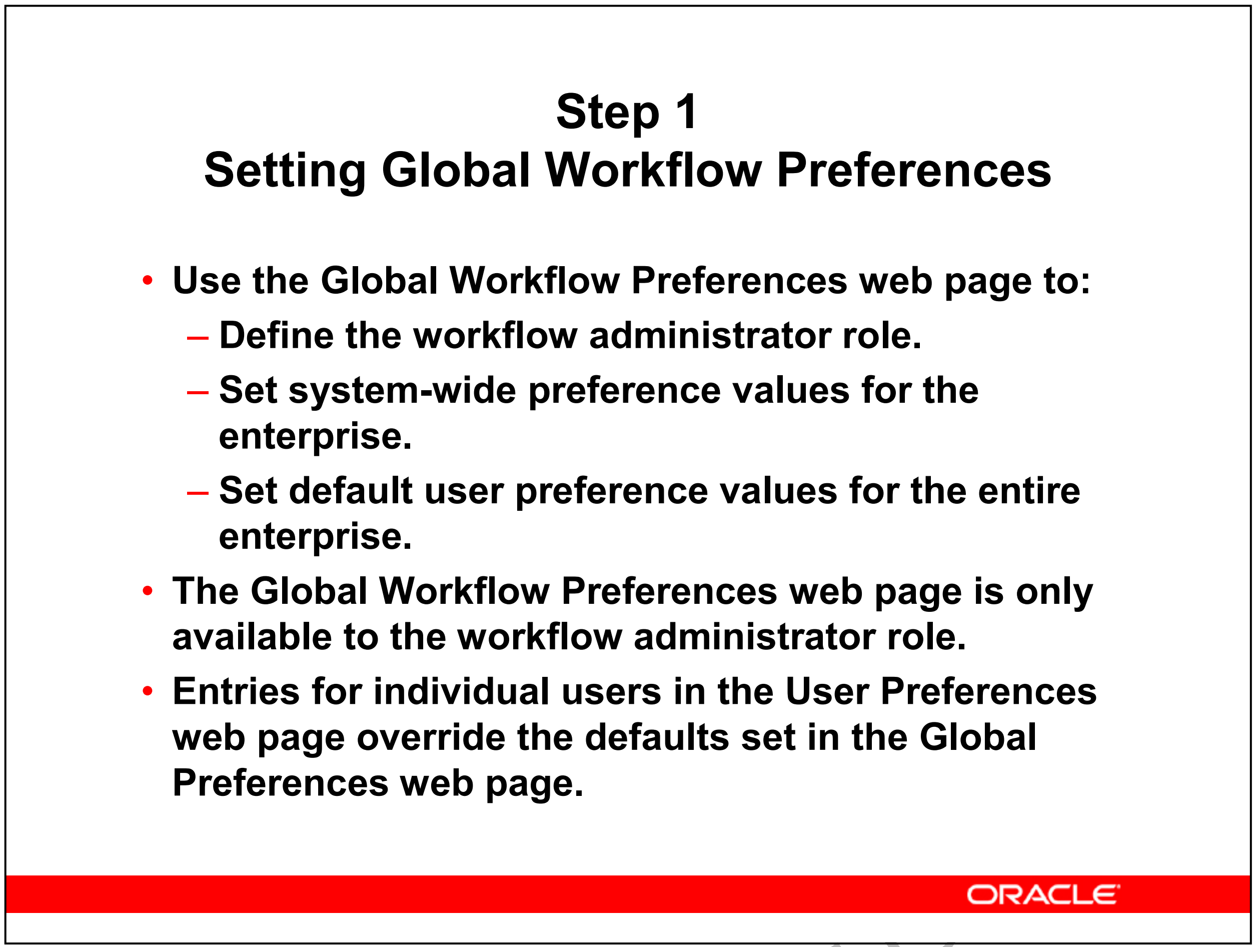
Step 7
Business Event System Configuration

ORACLE

Required Setup Steps for Oracle Workflow

1. Set system-wide preferences and default user preferences for your installation of Oracle Workflow using the Global Workflow Preferences page.
2. Map Oracle Workflow's directory service to the users and roles currently defined in your organization's directory repository by constructing views based on the directory repository tables.
3. Create a view called WF_LANGUAGES that identifies the languages defined in your Oracle database installation.
4. If you are using Oracle Workflow embedded in Oracle E-Business Suite, set the Socket Listener Activated profile option.
5. If you are using the standalone version of Oracle Workflow and your Workflow server is installed on a UNIX platform, define an environment variable called WF_RESOURCES.
6. Set up background Workflow engines to manage the load on the primary Workflow Engine by processing deferred and timed out activities and stuck processes.
7. Configure the Business Event System for event communication.

For more information, refer to the Setting Up Oracle Workflow chapter in the *Oracle Workflow Guide*.



Step 1
Setting Global Workflow Preferences

- Use the Global Workflow Preferences web page to:
  - Define the workflow administrator role.
  - Set system-wide preference values for the enterprise.
  - Set default user preference values for the entire enterprise.
- The Global Workflow Preferences web page is only available to the workflow administrator role.
- Entries for individual users in the User Preferences web page override the defaults set in the Global Preferences web page.

Setting Global Workflow Preferences

To access the Global Workflow Preferences web page for standalone Oracle Workflow, use a web browser to connect to the Oracle Workflow home page at the following URL:

<*webagent*>/wfa_html.home

Replace <*webagent*> with the base URL of the web agent configured for Oracle Workflow in your Web server.

For Oracle Workflow embedded in Oracle E-Business Suite, log on to a Workflow administrator responsibility.

Then choose the Global Workflow Preferences link for standalone Oracle Workflow or the Global Preferences link for Oracle Workflow embedded in Oracle E-Business Suite.

Note: You must have workflow administrator privileges to access the Global Workflow Preferences web page.

Workflow Administrator

Enter the role with workflow administrator privileges. If you want all users and roles to have workflow administrator privileges, such as in a development environment, enter an asterisk (*).

Workflow Web Agent

Enter the base URL of the Oracle web agent you defined for Oracle Workflow in Oracle HTTP Server. For Oracle Workflow embedded in Oracle E-Business Suite, the Workflow Web Agent value is retrieved from the system profile option Applications Web Agent.

Jinitiator Classid, Jinititator Download Location, Jinitiator Version

If you are using the version of Oracle Workflow embedded in Oracle E-Business Suite, enter the JInitiator plugin class ID, download location, and version number. This information is required for Oracle Workflow to launch Oracle E-Business Suite forms linked to notifications.

You can find the class ID and version number for the version of JInitiator you have installed in the jinit-version.txt file (C:\Program Files\Oracle\jinit<version>\doc\jinit-version.txt). The download location is the location where you have staged the JInitiator executable for download to users' client machines. For more information, refer to *Complete Guide to JInitiator for Oracle's E-Business Suite* (Note 162488.1) and *Upgrading Oracle JInitiator with Oracle Applications 11i (*Note 124606.1), available on MetaLink.

Local System

When you install Oracle Workflow, the database where the installation is located is automatically defined as a system in the Event Manger and set as the local system in the Global Workflow Preferences page.

System Status

- Enabled—Subscriptions are executed on all events.
- Local Only—Subscriptions are executed only on events raised on the local system.
- External Only—Subscriptions are executed only on events received from external systems.
- Disabled—No subscriptions are executed on any events.

Oracle Workflow sets the system status to Enabled by default. After you finish setting up the Business Event System, change the setting to the status you want for event processing.

Language, Territory

If you are using standalone Oracle Workflow, select the NLS_LANGUAGE and NLS_TERRITORY combination that defines the default language–dependent behavior and territory–dependent formatting of your users' notification sessions.

Note: For Oracle Workflow embedded in Oracle E-Business Suite, the Global Workflow Preferences page does not include the Language and Territory fields.

Date Format

Specify an Oracle9*i*-compliant date format that defines the default date format for the workflow database sessions of all users.

Document Home Node

This functionality is reserved for future use.

Send me electronic mail notifications

Select the default notification preference for all users:

- HTML mail
- Plain text mail with HTML attachments
- Plain text mail
- Plain text summary mail
- Do not send me mail

Note: The language, territory, and notification preference settings in the Global Preferences and User Preferences web pages are valid only if your directory service views map the Language, Territory, and Notification_Preference columns to the Oracle Workflow preferences table.

Step 2
Setting Up an Oracle Workflow Directory Service

Oracle Workflow:

- **Offers flexibility in defining who the Workflow users and roles are.**
- **References any directory repository for user and role information by creating views based on the database tables making up that repository.**
- **Provides local tables that contain columns similar to those defined in the views. These tables are used to store users and roles not included in the existing directory repository.**
- **Provides predefined Directory Service view definition and verification scripts.**

Oracle Workflow Directory Service

Oracle Workflow Directory Service Views

- WF_USERS
- WF_ROLES
- WF_USER_ROLES

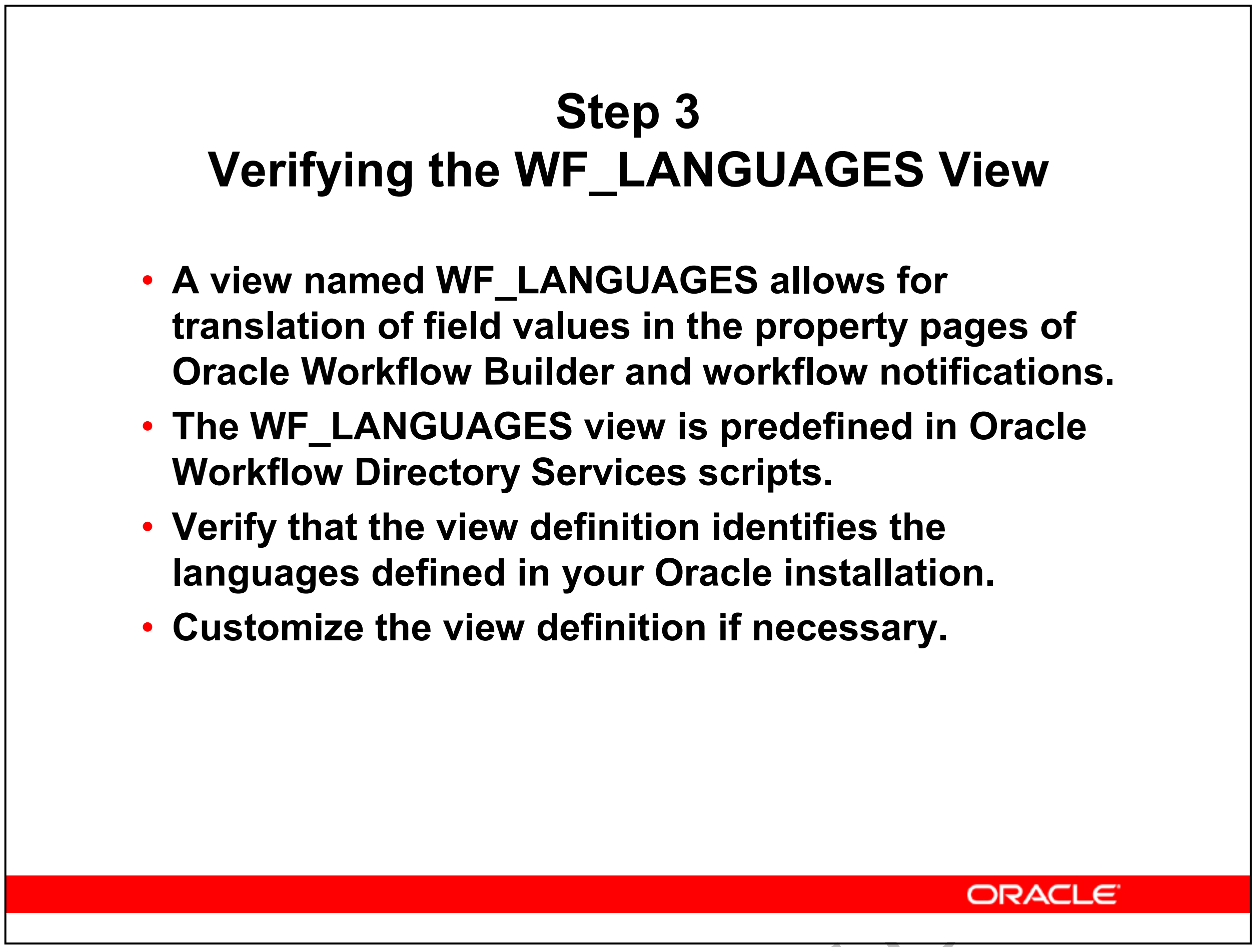
Oracle Workflow Directory Service Local Tables

- WF_LOCAL_USERS
- WF_LOCAL_ROLES
- WF_LOCAL_USER_ROLES

Oracle Workflow Directory Service Scripts

- wfdirhrv.sql: Maps users and roles over a unified Oracle E-Business Suite environment
- wfdirouv.sql: Maps users and roles onto the native users and roles defined in the Oracle database server
- wfdircsv.sql: Maps users and roles to the Oracle Workflow Local_* tables only

If you create your own views or modify any of the predefined directory service view definitions, run the script wfdirchk.sql to verify your directory service model.



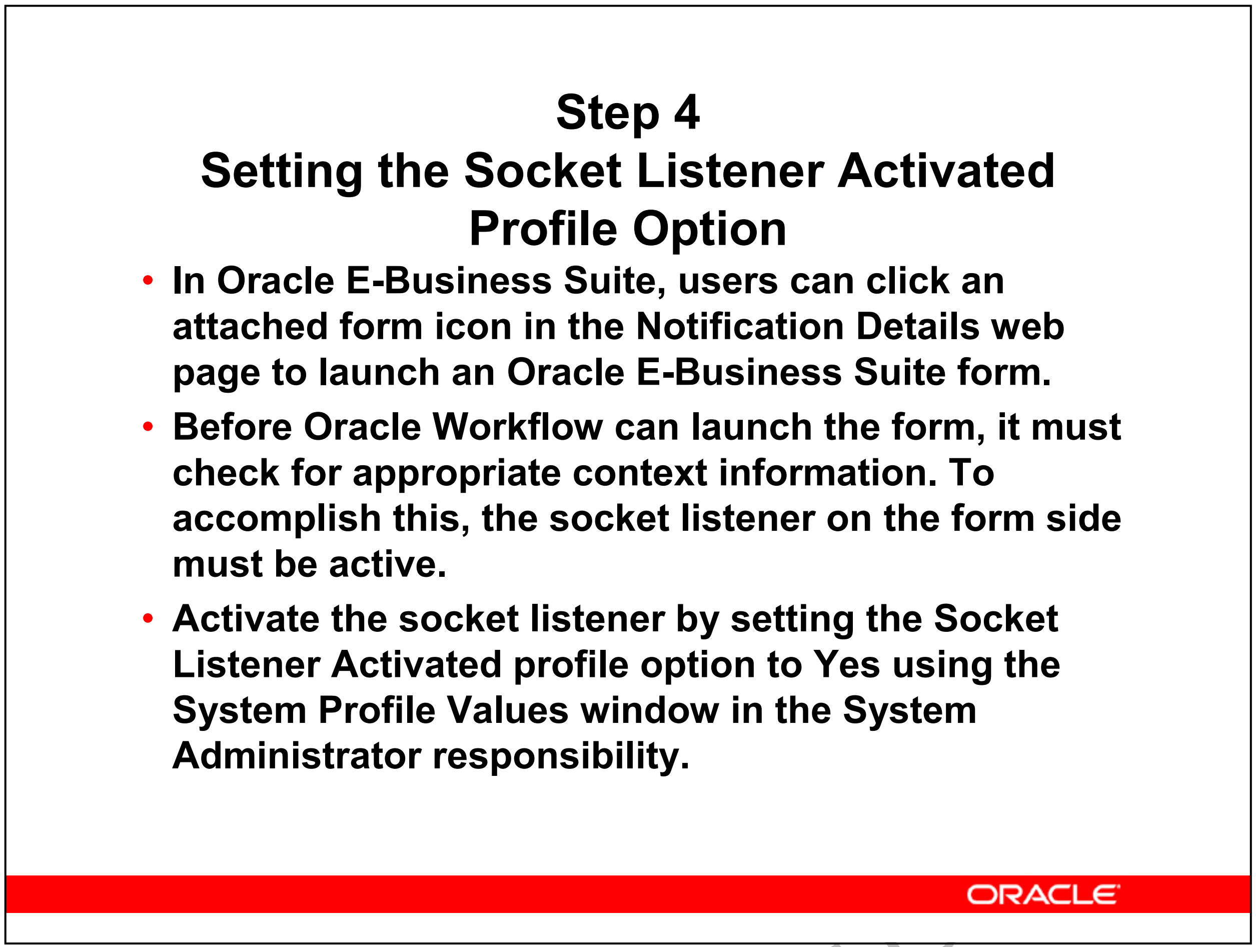
Step 3
Verifying the WF_LANGUAGES View

- A view named WF_LANGUAGES allows for translation of field values in the property pages of Oracle Workflow Builder and workflow notifications.
- The WF_LANGUAGES view is predefined in Oracle Workflow Directory Services scripts.
- Verify that the view definition identifies the languages defined in your Oracle installation.
- Customize the view definition if necessary.

Mandatory Columns of the WF_LANGUAGES View

- Code - The language code
- Display_Name - The display name of the language
- NLS_Language - The value of the Oracle NLS_LANGUAGE initialization parameter that specifies the default language-dependent behavior of a session
- NLS_Territory - The value of the Oracle NLS_TERRITORY initialization parameter that specifies the default territory-dependent date and numeric formatting of a session
- NLS_Codeset - The character set for the language
- Installed_Flag - A flag to indicate whether the language is installed and available for use

Note: A sample WF_LANGUAGES view is included in the script of each of the predefined directory services that Oracle Workflow provides.

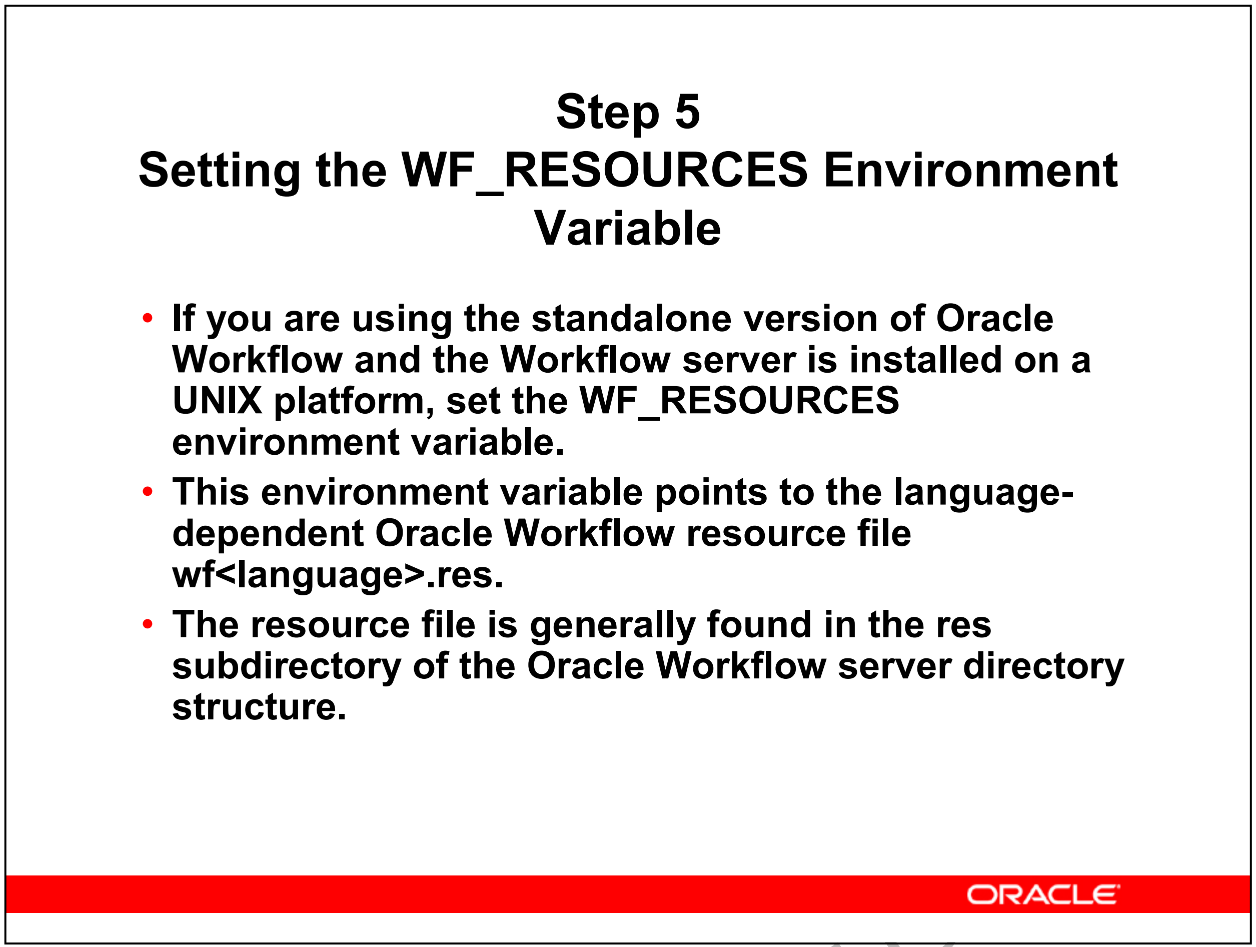


Step 4 Setting the Socket Listener Activated Profile Option

- In Oracle E-Business Suite, users can click an attached form icon in the Notification Details web page to launch an Oracle E-Business Suite form.
- Before Oracle Workflow can launch the form, it must check for appropriate context information. To accomplish this, the socket listener on the form side must be active.
- Activate the socket listener by setting the Socket Listener Activated profile option to Yes using the System Profile Values window in the System Administrator responsibility.

Setting the Socket Listener Activated Profile Option

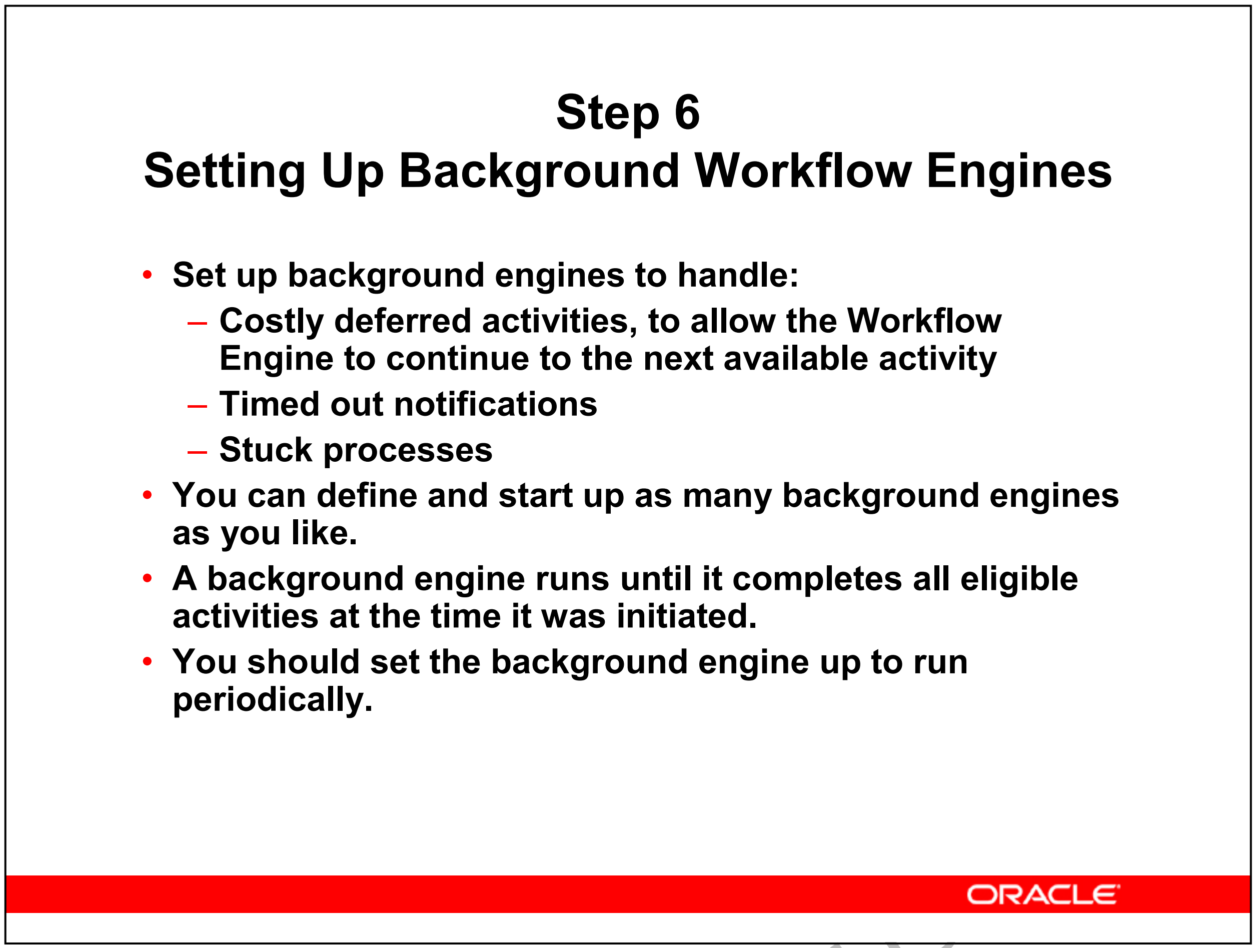
For more information about setting profile options, refer to the Overview of Setting User Profiles section in the *Oracle Applications System Administrator's Guide*.



Step 5
Setting the WF_RESOURCES Environment Variable

- If you are using the standalone version of Oracle Workflow and the Workflow server is installed on a UNIX platform, set the WF_RESOURCES environment variable.
- This environment variable points to the language-dependent Oracle Workflow resource file wf<language>.res.
- The resource file is generally found in the res subdirectory of the Oracle Workflow server directory structure.

Step 6 Setting Up Background Workflow Engines



Step 6
Setting Up Background Workflow Engines

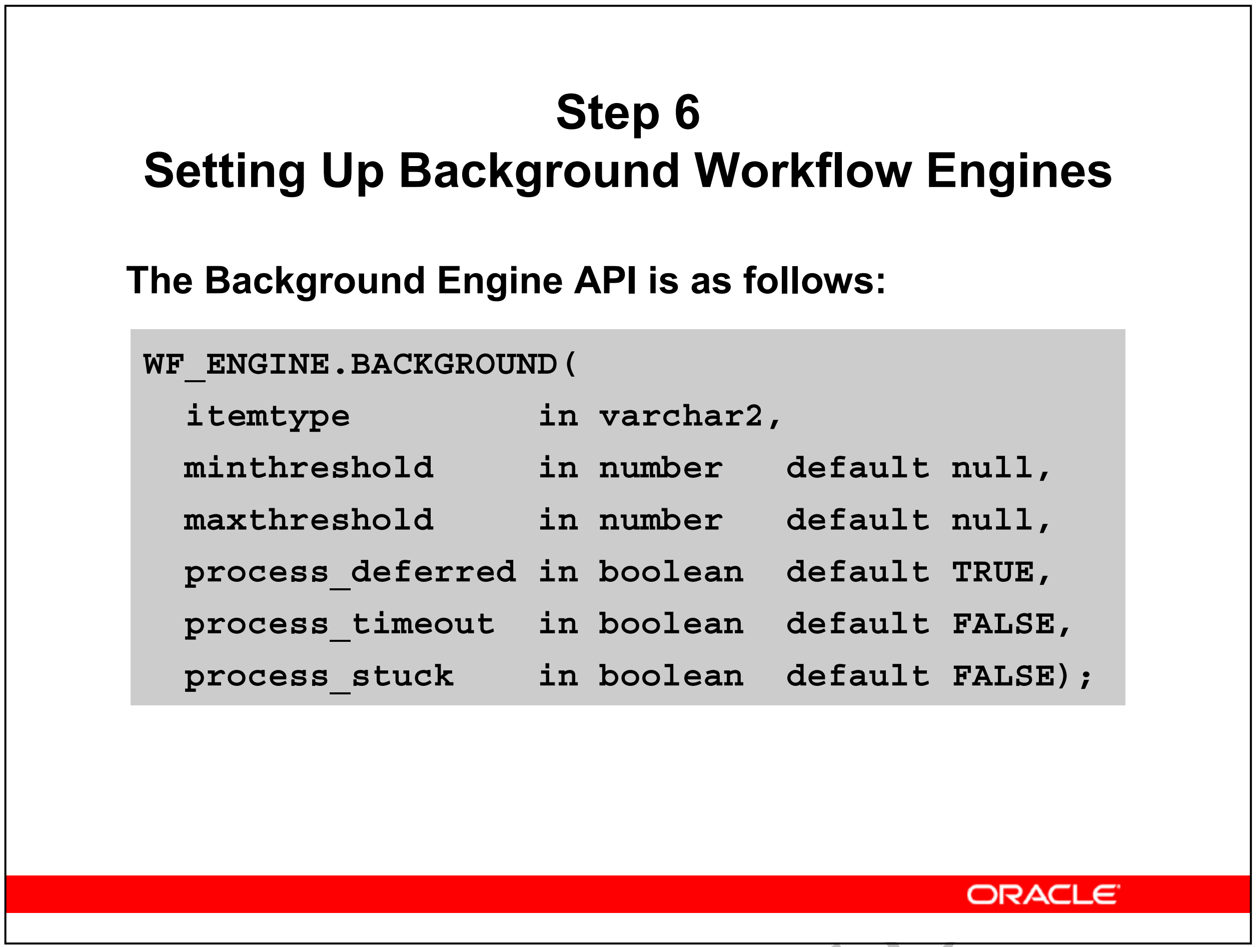
- Set up background engines to handle:
  - Costly deferred activities, to allow the Workflow Engine to continue to the next available activity
  - Timed out notifications
  - Stuck processes
- You can define and start up as many background engines as you like.
- A background engine runs until it completes all eligible activities at the time it was initiated.
- You should set the background engine up to run periodically.

To Start a Background Engine

- If you are using the standalone version of Oracle Workflow, then use the WF_ENGINE.BACKGROUND() API to start up a background engine. Sample scripts that repeatedly run the background engine are provided with the standalone version of Oracle Workflow.
- If you are using the version of Oracle Workflow embedded in Oracle E-Business Suite, start a background engine by submitting the Background Process concurrent program using the Submit Requests form.

Note: Ensure that you have a least one background engine that can check for timed out activities, one that can process deferred activities, and one that can handle stuck processes. At a minimum, you need to set up one background engine that can handle both timed out and deferred activities as well as stuck processes. Generally, you should run a separate background engine to check for stuck processes at less frequent intervals than the background engine that you run for deferred or timed out activities.

Do not run more background engines concurrently than your server has CPU processors.



Step 6
Setting Up Background Workflow Engines

The Background Engine API is as follows:

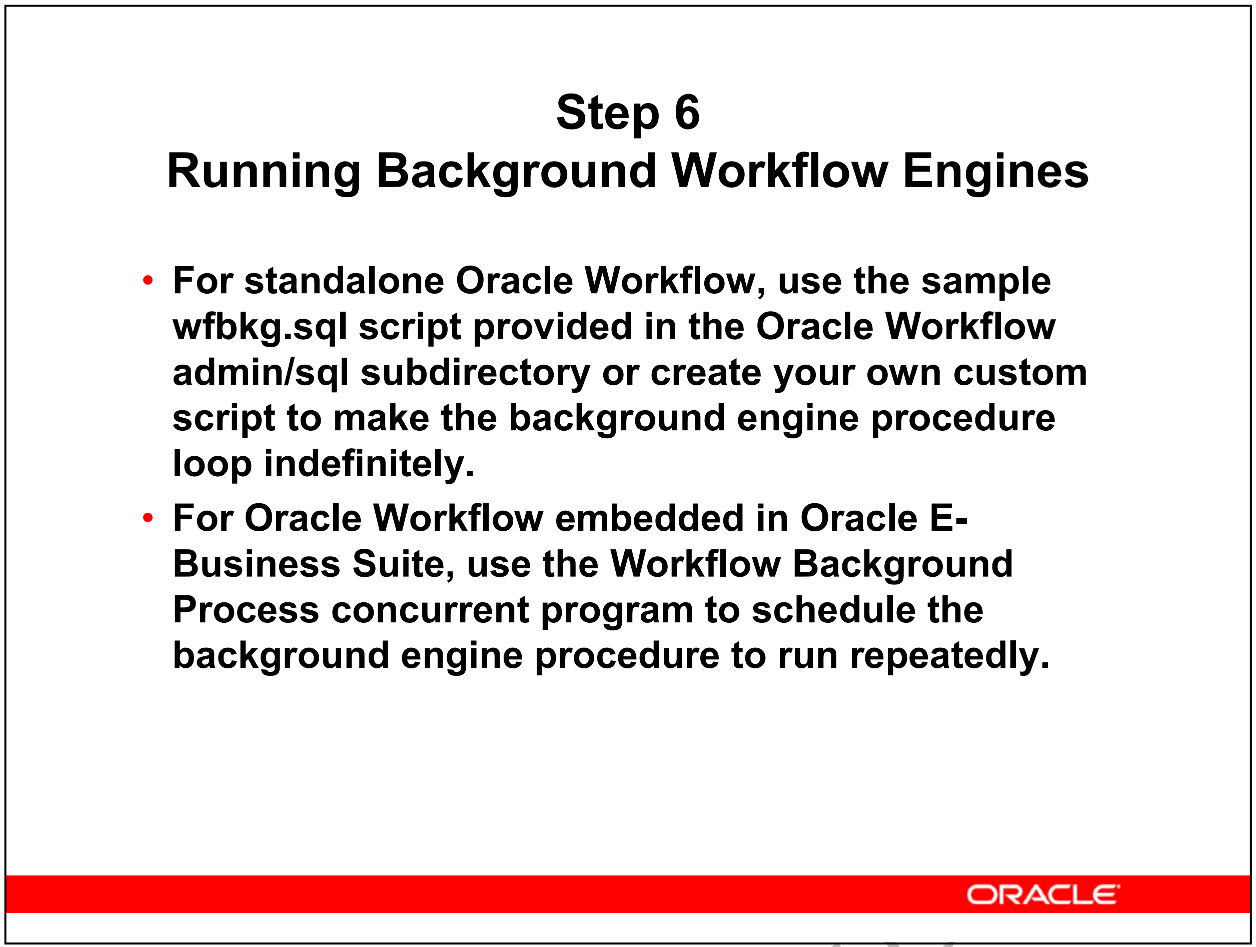
```
WF_ENGINE.BACKGROUND(
  itemtype          in varchar2,
  minthreshold      in number   default null,
  maxthreshold      in number   default null,
  process_deferred in boolean  default TRUE,
  process_timeout  in boolean  default FALSE,
  process_stuck    in boolean  default FALSE);
```

ORACLE

Background Engine

Background Engine Parameters

- itemtype: Optional item type to restrict this engine to activities associated with that item type.
- minthreshold: Optional minimum cost that an activity must have for this background engine to execute it, in hundredths of a second.
- maxthreshold: Optional maximum cost an activity can have for this background engine to execute it, in hundredths of a second.
- process_deferred: Specify TRUE or FALSE to indicate whether the engine should check for deferred activities.
- process_timeout: Specify TRUE or FALSE to indicate whether the engine should check for timed out activities.
- process_stuck: Specify TRUE or FALSE to indicate whether the engine should check for stuck processes.



Step 6 Running Background Workflow Engines

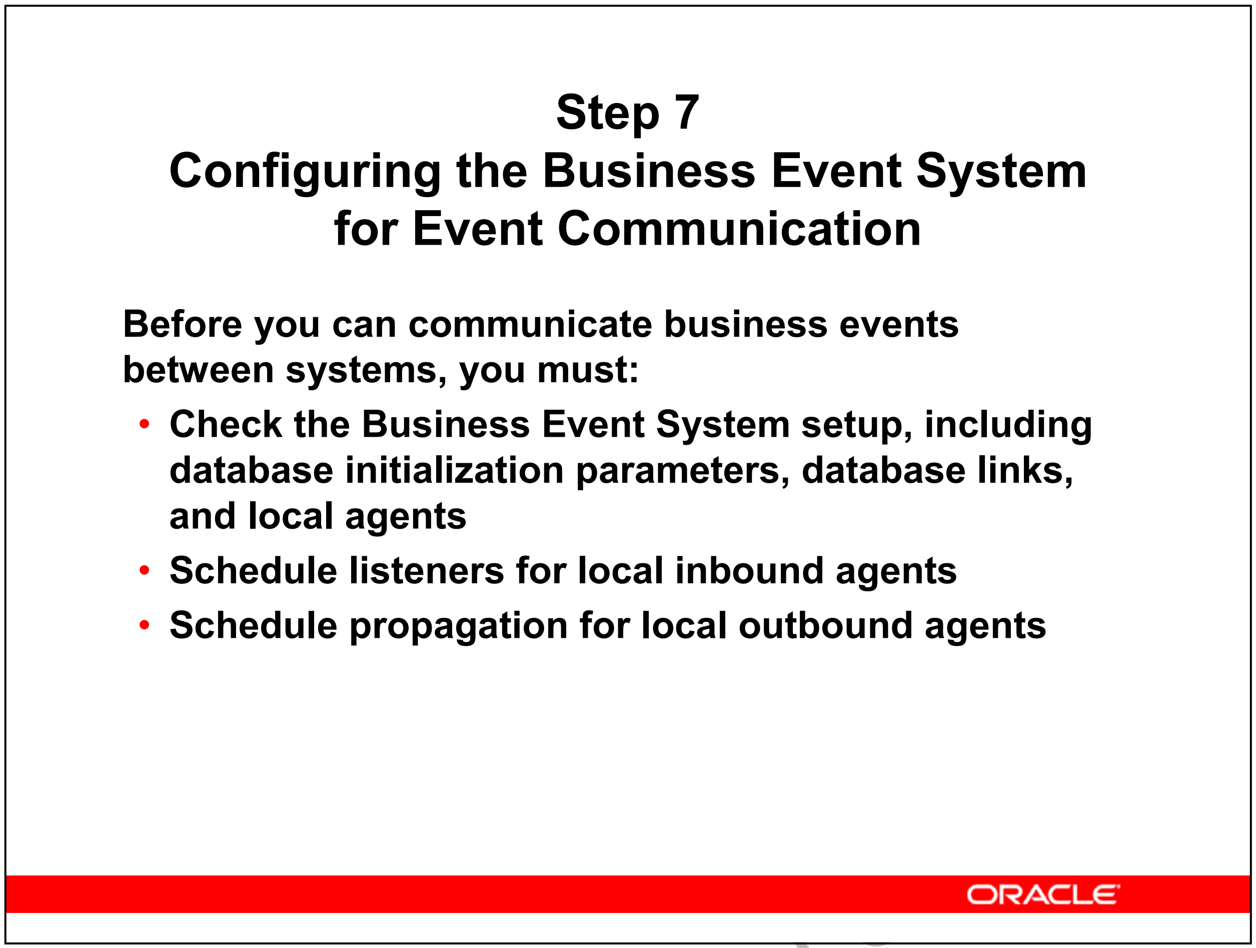
- **For standalone Oracle Workflow, use the sample wfbkg.sql script provided in the Oracle Workflow admin/sql subdirectory or create your own custom script to make the background engine procedure loop indefinitely.**
- **For Oracle Workflow embedded in Oracle E-Business Suite, use the Workflow Background Process concurrent program to schedule the background engine procedure to run repeatedly.**

Running the Background Engine

You can set up as many background engines as needed, but if you set up only one, then that background engine must handle both deferred and timed out activities as well as stuck processes. That is, the process_deferred, process_timeout, and process_stuck parameters must all be TRUE.

Generally, you should run a separate background engine to check for stuck processes at less frequent intervals than the background engine that you run for deferred or timed out activities, normally not more often than once a day. Run the background engine to check for stuck processes when the load on the system is low.

Step 7 Configuring the Business Event System for Event Communication



Step 7
Configuring the Business Event System for Event Communication

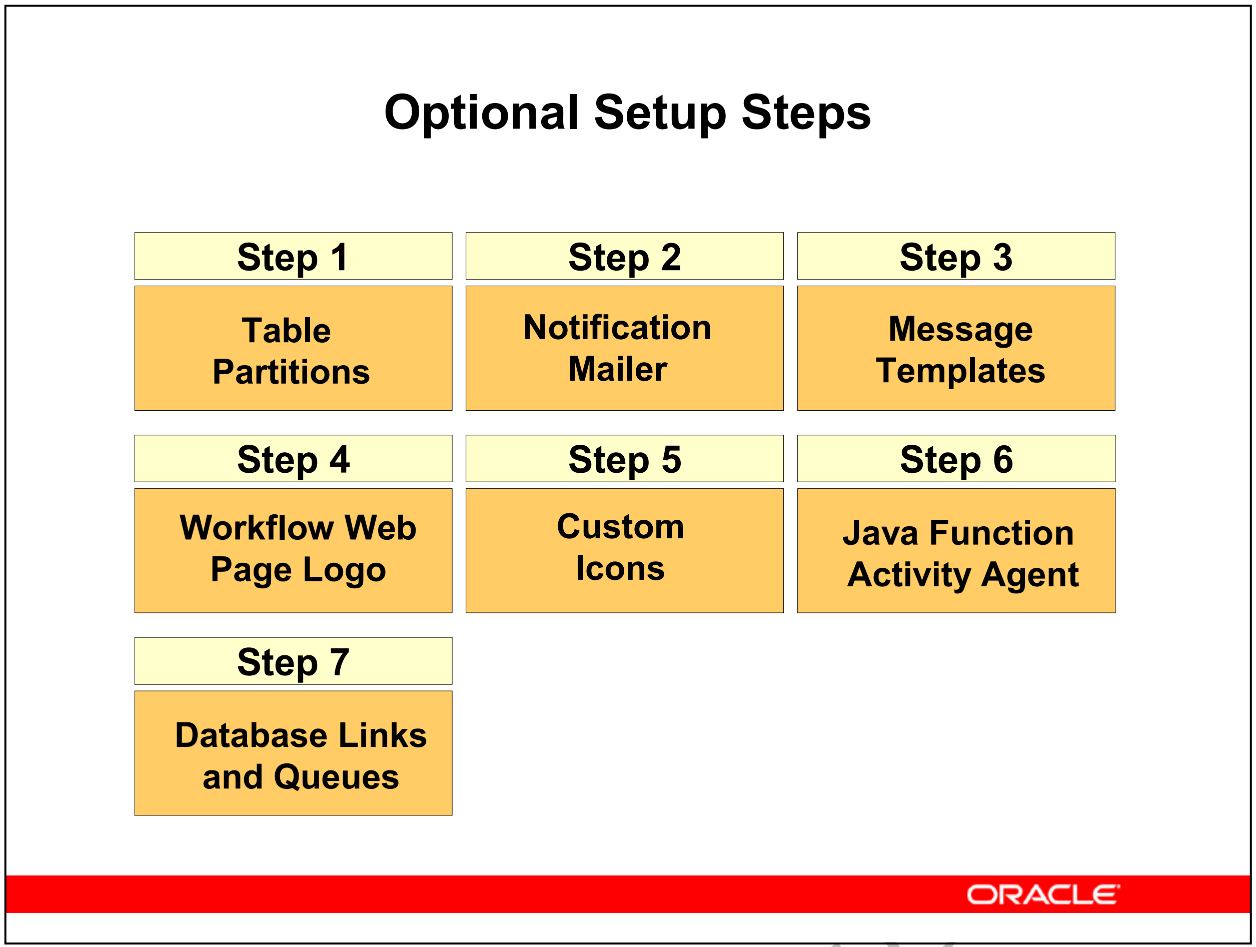
Before you can communicate business events between systems, you must:

- **Check the Business Event System setup, including database initialization parameters, database links, and local agents**
- **Schedule listeners for local inbound agents**
- **Schedule propagation for local outbound agents**

ORACLE

Configuring the Business Event System for Event Communication

For detailed instructions, see the *Configuring Oracle Workflow for Event Communication* chapter.



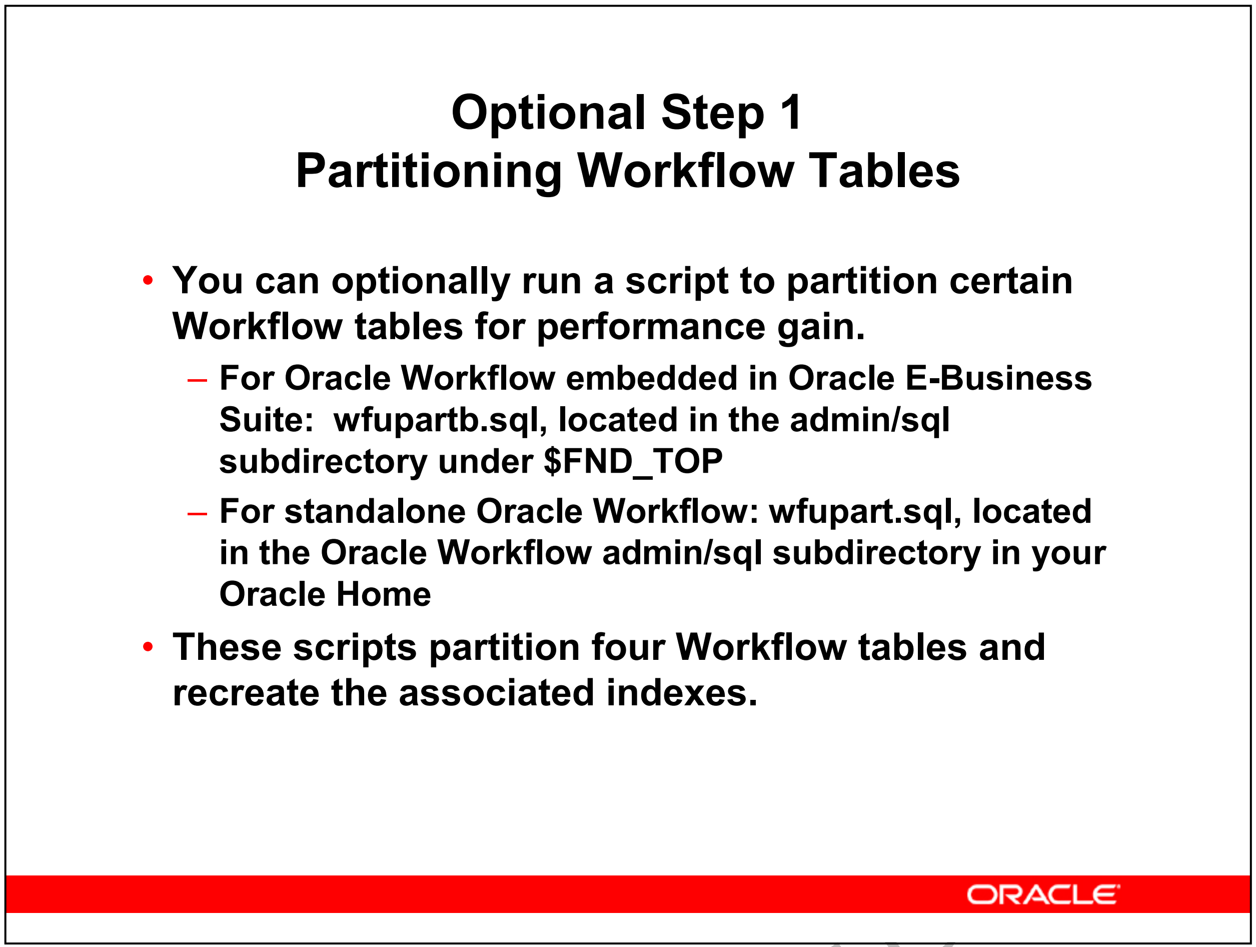
Optional Setup Steps

Step 1	Step 2	Step 3
Table Partitions	Notification Mailer	Message Templates
Step 4	Step 5	Step 6
Workflow Web Page Logo	Custom Icons	Java Function Activity Agent
Step 7		
Database Links and Queues		

Optional Setup Steps for Oracle Workflow

1. Partition certain Workflow tables for performance gain.
2. Run the Notification Mailer program if you want to allow your users to receive notifications by e-mail.
3. Modify the templates for your electronic mail notifications.
4. Customize the company logo that appears in Oracle Workflow's web pages.
5. Include additional icons in your Oracle Workflow icons subdirectory to customize the diagrammatic representation of your workflow processes.
6. Start the Java Function Activity Agent if you are using the standalone version of Oracle Workflow and you want to run external Java function activities.
7. Set up database links and queues for the Business Event System.

For more information, refer to the Setting Up Oracle Workflow chapter in the *Oracle Workflow Guide*.



Optional Step 1
Partitioning Workflow Tables

- **You can optionally run a script to partition certain Workflow tables for performance gain.**
  - **For Oracle Workflow embedded in Oracle E-Business Suite: wfupartb.sql, located in the admin/sql subdirectory under $FND_TOP**
  - **For standalone Oracle Workflow: wfupart.sql, located in the Oracle Workflow admin/sql subdirectory in your Oracle Home**
- **These scripts partition four Workflow tables and recreate the associated indexes.**

ORACLE

Partitioning Workflow Tables

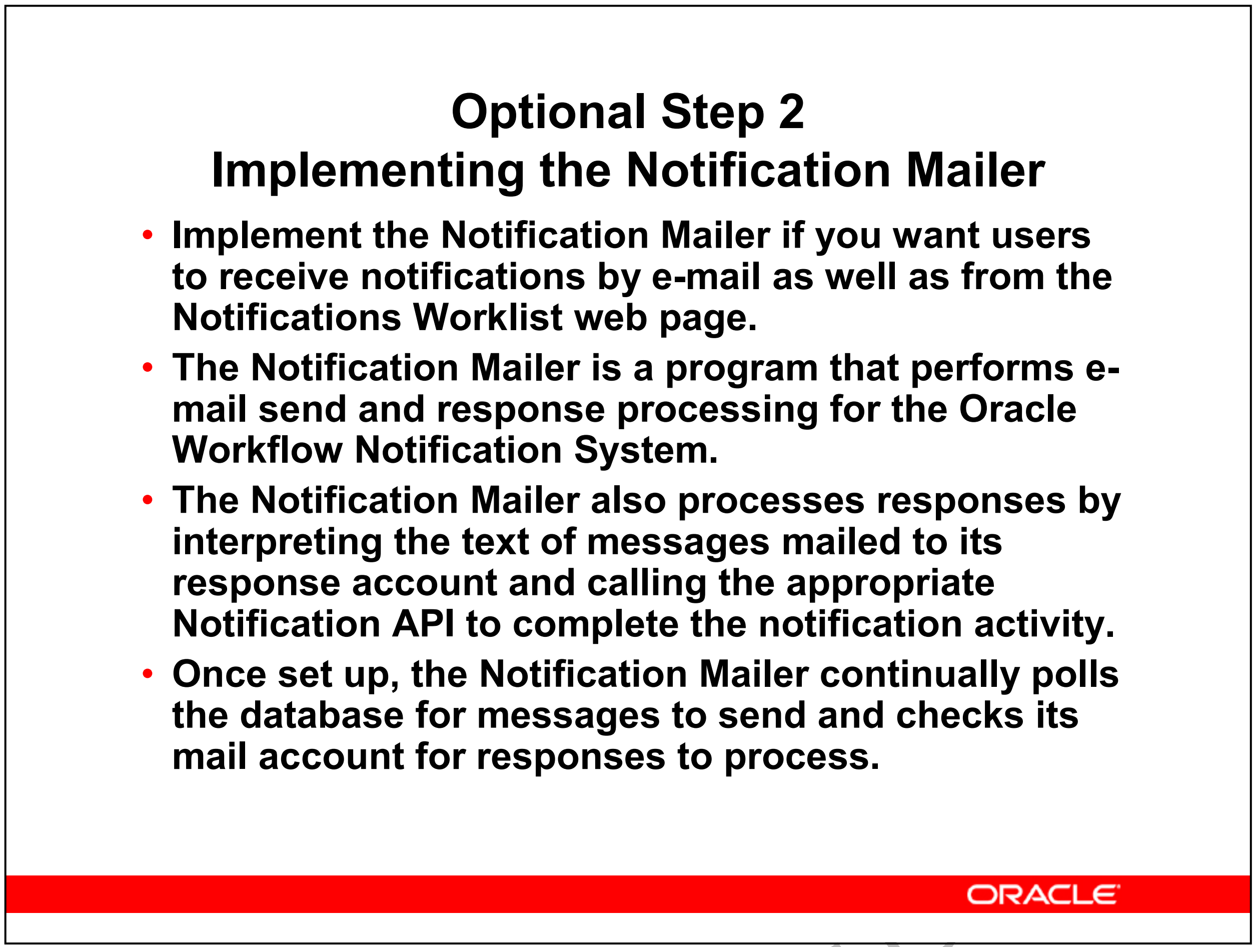
The wfupartb.sql and wfupart.sql scripts partition the following tables:

- WF_ITEM_ACTIVITY_STATUSES
- WF_ITEM_ACTIVITY_STATUSES_H
- WF_ITEM_ATTRIBUTE_VALUES
- WF_ITEMS

The script recreates the following indexes:

- WF_ITEM_ACTIVITY_STATUSES_PK
- WF_ITEM_ACTIVITY_STATUSES_N1
- WF_ITEM_ACTIVITY_STATUSES_N2
- WF_ITEM_ACTIVITY_STATUSES_H_N1
- WF_ITEM_ACTIVITY_STATUSES_H_N2
- WF_ITEM_ATTRIBUTE_VALUES_PK
- WF_ITEMS_PK
- WF_ITEMS_N1

- WF_ITEMS_N2
- WF_ITEMS_N3



Optional Step 2 Implementing the Notification Mailer

- Implement the Notification Mailer if you want users to receive notifications by e-mail as well as from the Notifications Worklist web page.
- The Notification Mailer is a program that performs e-mail send and response processing for the Oracle Workflow Notification System.
- The Notification Mailer also processes responses by interpreting the text of messages mailed to its response account and calling the appropriate Notification API to complete the notification activity.
- Once set up, the Notification Mailer continually polls the database for messages to send and checks its mail account for responses to process.

Notification Mailer

Full MIME Support

Oracle Workflow fully supports Multi-purpose Internet Mail Extensions (MIME) encoded messages. This means that the Notification Mailer can exchange messages with workflow users containing languages with different character sets and multimedia encoded content.

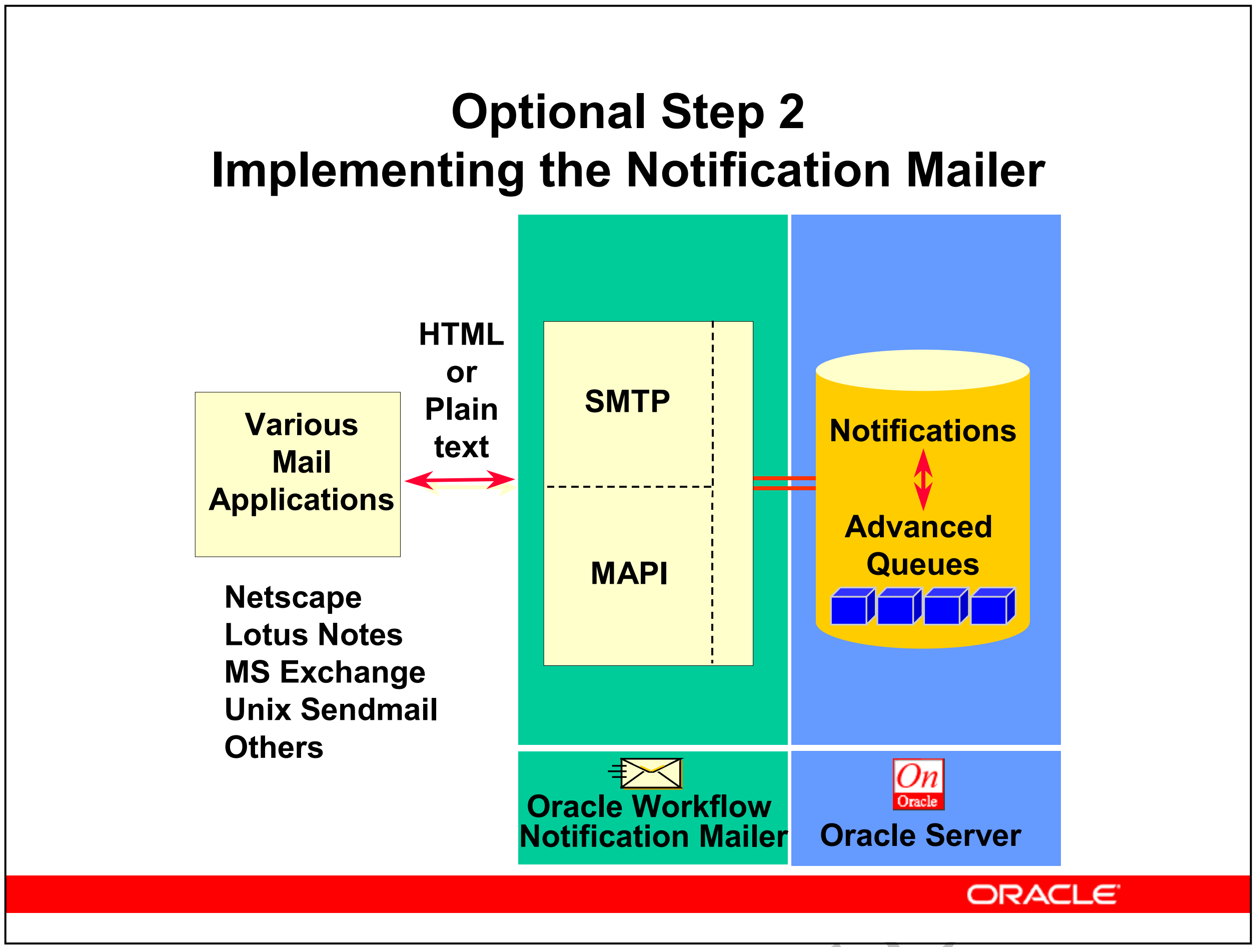
Notification Preferences

Oracle Workflow allows you to determine how you view notifications by setting a notification preference in the User Preferences web page.

There are five types of notification preferences:

- MAILTEXT: Plain text mail. The notification message appears as plain text with no attachments.
- MAILHTML: HTML mail. The notification message appears as HTML-formatted text with at least one other attachment that is a link to the notification in the Notifications web page.

- MAILATTH: Plain text mail with HTML attachments. The notification message appears as plain text with at least two other attachments, one being an HTML-formatted version of the message, and the other being a link to the notification in the Notifications web page.
- SUMMARY: Plain text summary mail. The message is a plain text summary of all open notifications.
- QUERY: Do not send me mail. The Notification Mailer does not send you email notifications. You must query your notifications in the Notifications web page.



Notification Mailer

The Notification Mailer sends e-mail and processes responses for the Oracle Workflow Notifications System. The Notification Mailer:

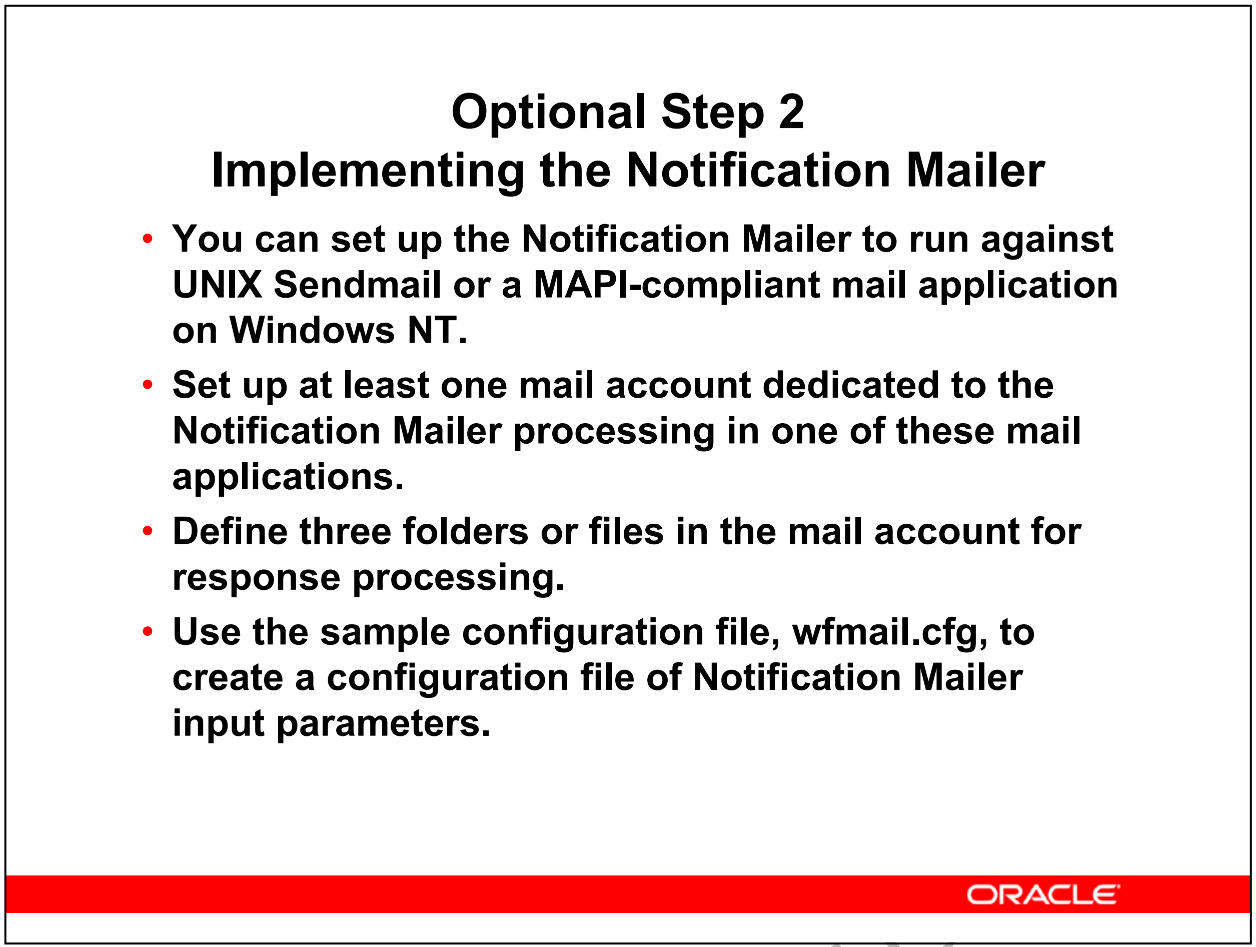
- Polls the database for messages to send.
- Dequeues these messages from the SMTP (Simple Mail Transfer Protocol) advanced queue.
- Resolves the e-mail address of the recipient role, which itself can be a mail distribution list.
- Switches its database session to use the preferred language of the role and territory setting.
- Selects information from the database as defined by the attributes of the message.
- Generates the message using a message template.
- Sends the message via UNIX Sendmail or a MAPI-compliant mail application (Messaging Application Programming Interface).
- Processes responses returned and calls the appropriate Notification API to complete the notification activity.
- Remains running unless a database failure occurs or the PL/SQL package state changes for the session.

For the standalone version of Oracle Workflow, create and run a perpetual shell script that restarts the Notification Mailer, should it shut down. An example UNIX shell script named wfmail.csh is located in the Oracle Workflow server bin subdirectory.

For Oracle Workflow embedded in Oracle E-Business Suite, you can use the concurrent manager to restart the Notification Mailer program manually or schedule it to restart periodically.

Attention: Check whether your platform and mail application are certified for use with Oracle Workflow before implementing the the MAPI-compliant version of the Notification Mailer. The Microsoft Outlook E-mail Security Update that was released on June 7, 2000 desupports the MAPI Common Messaging Calls (CMC) interface used by the Oracle Workflow MAPI Mailer. (See: *OL2000: Developer Information About the Outlook E-mail Security Update*, http://support.microsoft.com/support/kb/articles/Q262/7/01.ASP.) As a result, the Oracle Workflow MAPI Mailer is not certified on any Microsoft Windows platforms where this Microsoft Outlook E-mail Security Update or above has been applied. The Oracle Workflow MAPI Mailer is not certified on Windows XP.

Workflow customers running on NT/2000 are certified to install the UNIX version of the Oracle Workflow Notification Mailer (on UNIX) and connect to a Workflow Server database running on NT/2000.



Optional Step 2
Implementing the Notification Mailer

- You can set up the Notification Mailer to run against UNIX Sendmail or a MAPI-compliant mail application on Windows NT.
- Set up at least one mail account dedicated to the Notification Mailer processing in one of these mail applications.
- Define three folders or files in the mail account for response processing.
- Use the sample configuration file, wfmail.cfg, to create a configuration file of Notification Mailer input parameters.

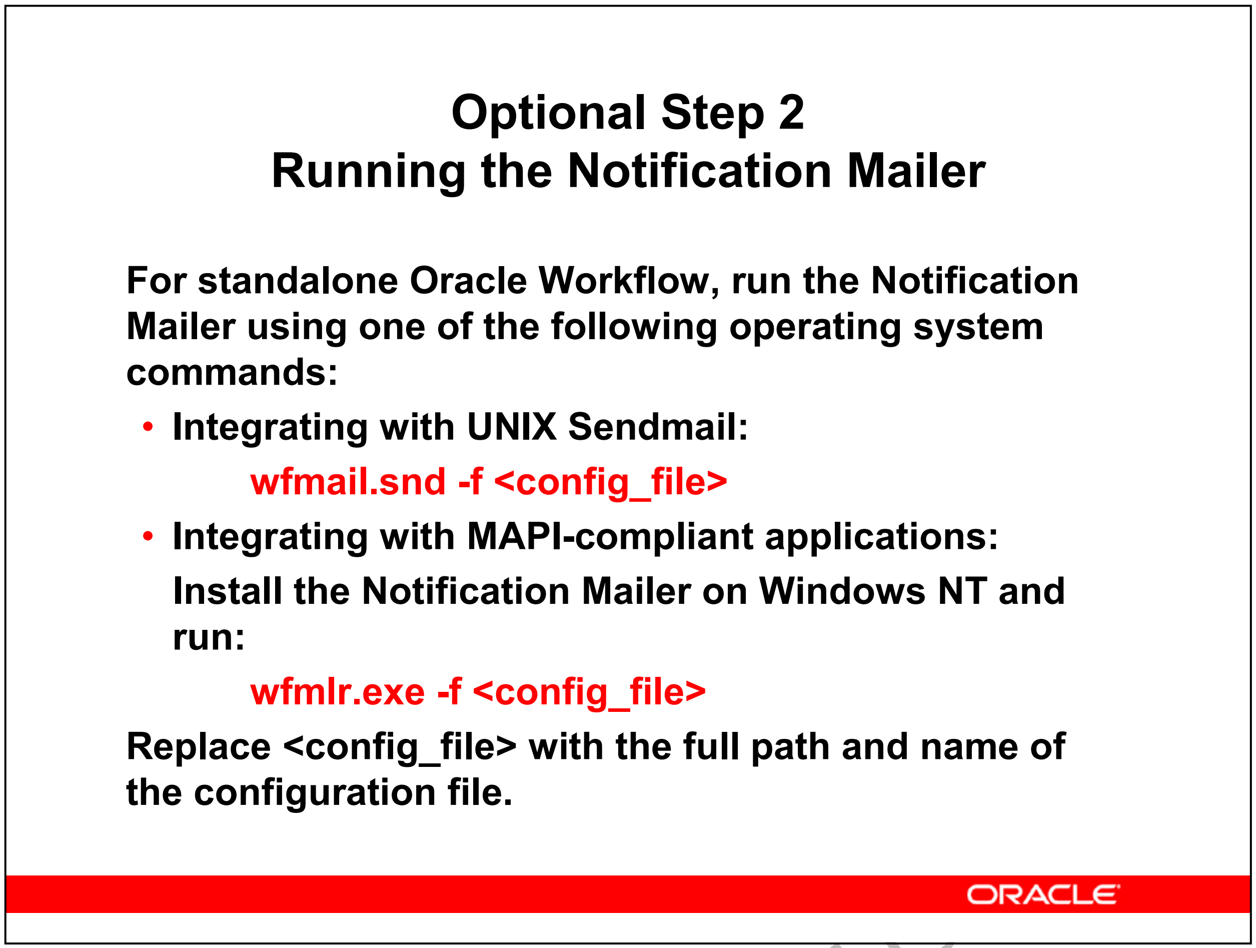
ORACLE

Implementing the Notification Mailer

- The e-mail account for the Notification Mailer must not be used for any other purposes.
- Oracle Workflow provides an example configuration file called wfmail.cfg. If you are using the standalone version of Oracle Workflow, the file resides in your Oracle Workflow server directory structure under the subdirectory *res*. For the version of Oracle Workflow embedded in Oracle E-Business Suite, the file resides in the *resource* subdirectory under $FND_TOP on your server. The file also resides on your PC in the *<drive>:\<ORACLE_HOME>*\wf\data subdirectory.
- The configuration file lists the parameters you want to run with the program and must have the following format:

```
#Description
PARAMETER1=value1
#Description
PARAMETER2=value2
. . .
```

Note: If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to configure and run the Notification Mailer. For more information, please refer to the Oracle Applications Manager online help.



Optional Step 2
Running the Notification Mailer

For standalone Oracle Workflow, run the Notification Mailer using one of the following operating system commands:

- **Integrating with UNIX Sendmail:**

  ```
  wfmail.snd -f <config_file>
  ```

- **Integrating with MAPI-compliant applications:**

  Install the Notification Mailer on Windows NT and run:

  ```
  wfmlr.exe -f <config_file>
  ```

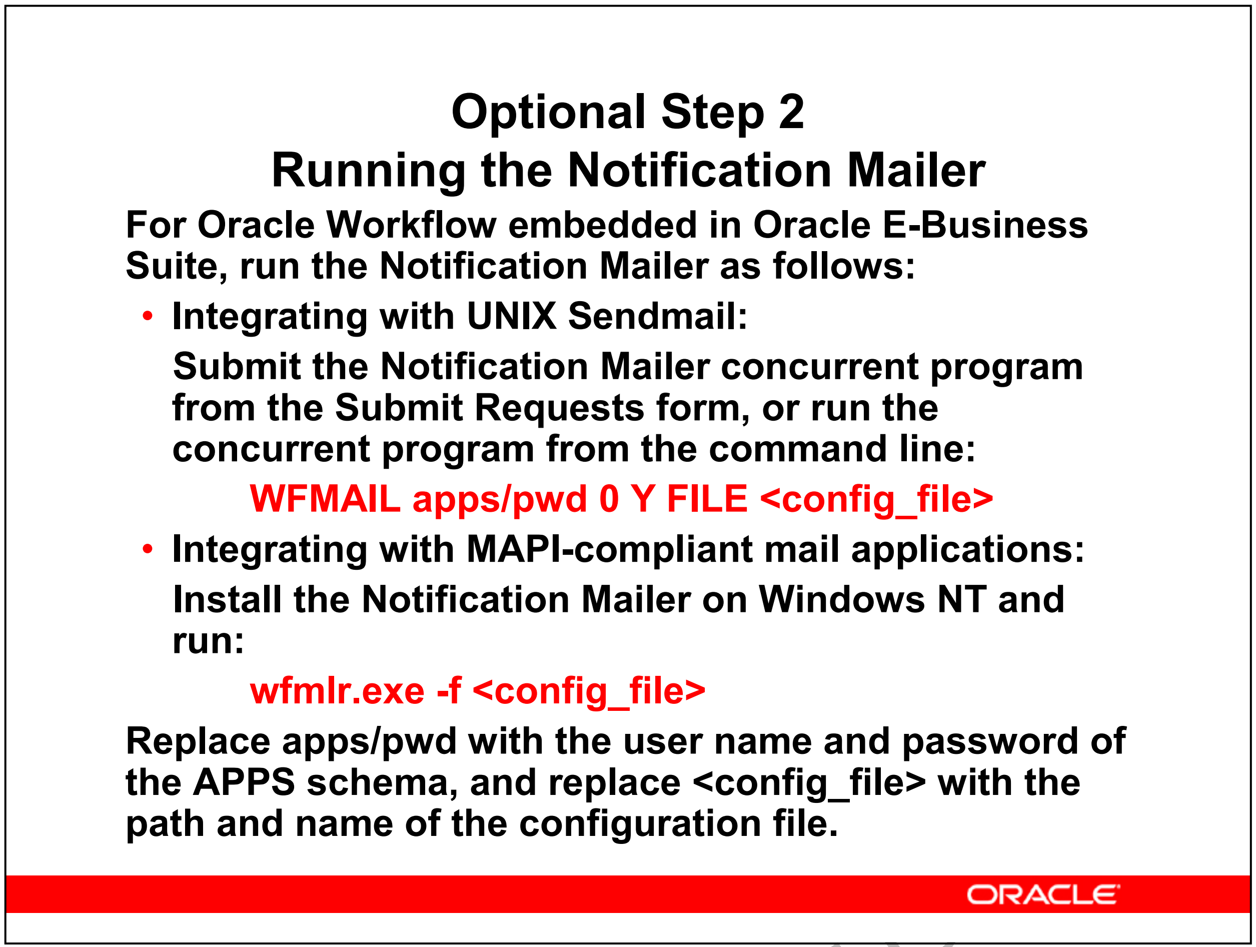
Replace <config_file> with the full path and name of the configuration file.

ORACLE

Location of the Notification Mailer

The UNIX Sendmail Notification Mailer executable resides on your server in the $ORACLE_HOME/bin subdirectory.

The MAPI-compliant Notification Mailer executable resides on your Windows NT PC in the <drive>:\<ORACLE_HOME>\bin subdirectory.



Optional Step 2
Running the Notification Mailer

For Oracle Workflow embedded in Oracle E-Business Suite, run the Notification Mailer as follows:

- **Integrating with UNIX Sendmail:**
  Submit the Notification Mailer concurrent program from the Submit Requests form, or run the concurrent program from the command line:
  ```
  WFMAIL apps/pwd 0 Y FILE <config_file>
  ```
- **Integrating with MAPI-compliant mail applications:**
  Install the Notification Mailer on Windows NT and run:
  ```
  wfmlr.exe -f <config_file>
  ```

Replace apps/pwd with the user name and password of the APPS schema, and replace <config_file> with the path and name of the configuration file.

Running the Notification Mailer

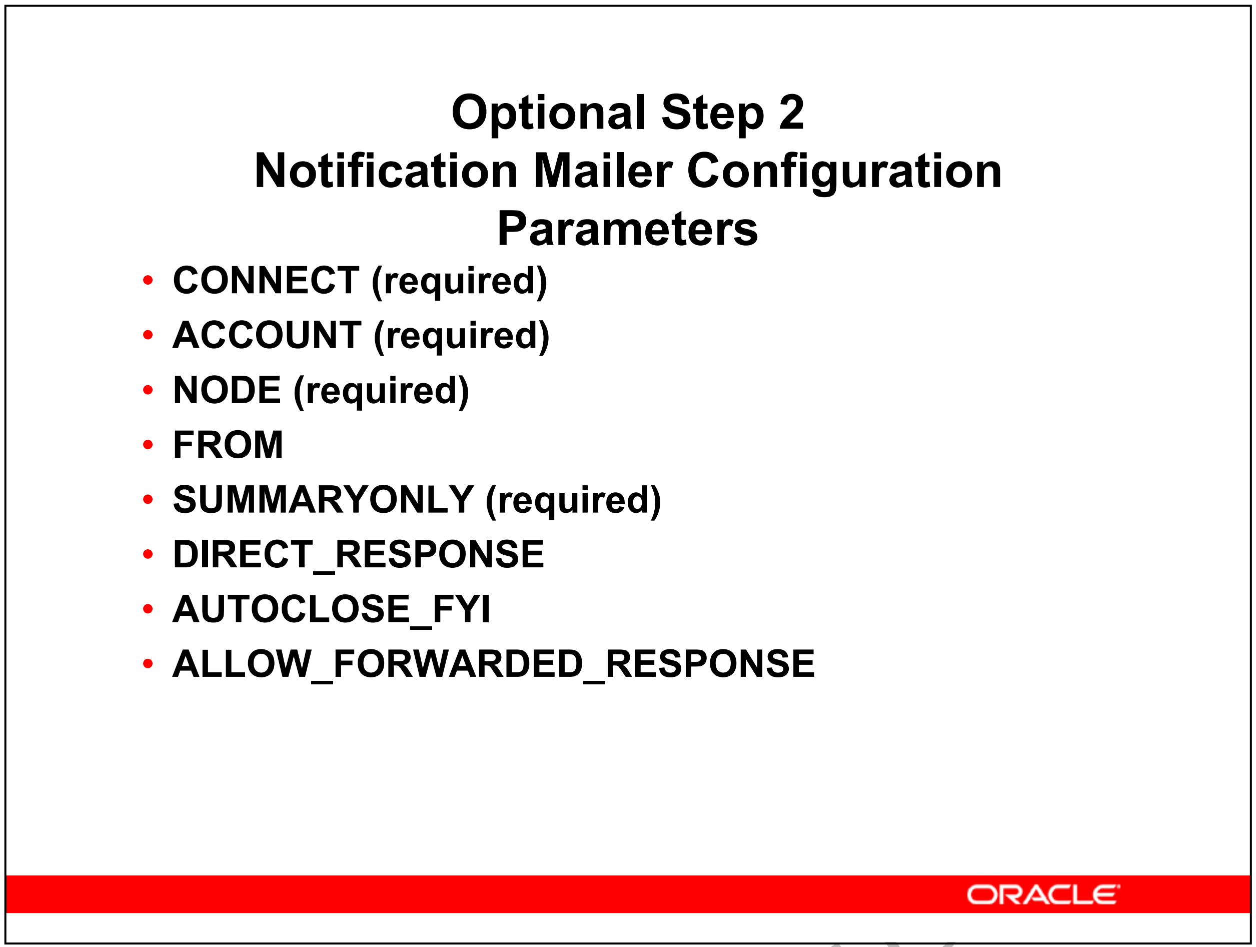
Notification Mailer Concurrent Program

- The concurrent program links to the Sendmail version of the mailer program.
- Your system administrator needs to add the Notification Mailer concurrent program to a request security group for the responsibility from which you want to run the program.
- In the Parameters window of the Submit Requests, enter the path and filename of the configuration file.

MAPI-Compliant Notification Mailer

The MAPI-compliant Notification Mailer executable resides on your Windows NT PC in the <drive>:\<ORACLE_HOME>\bin subdirectory.

Note: If you are using the version of Oracle Workflow embedded in Oracle Applications and you have implemented Oracle Applications Manager, you can use Oracle Workflow Manager to configure and run the Notification Mailer. For more information, please refer to the Oracle Applications Manager online help.



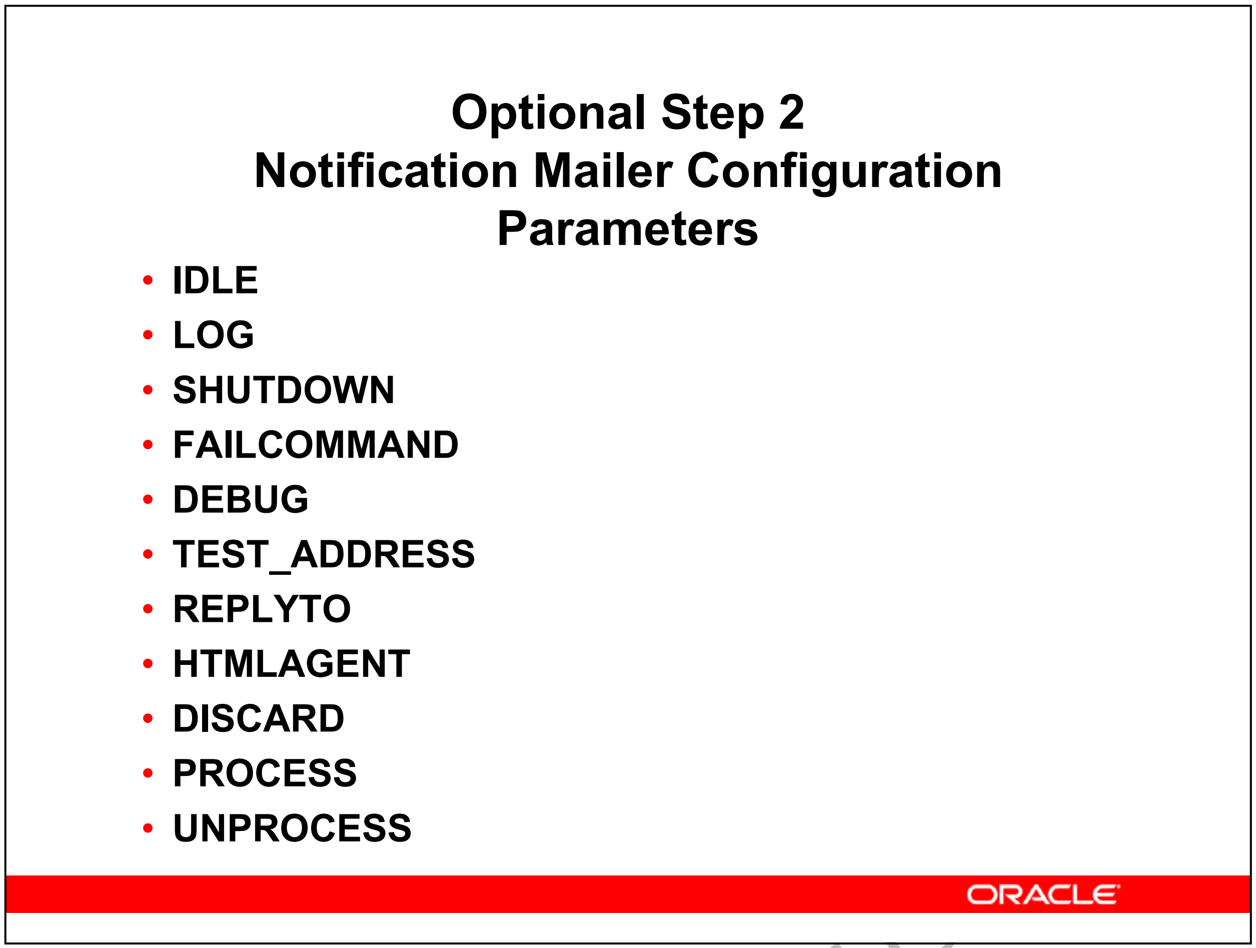
Optional Step 2 Notification Mailer Configuration Parameters

- CONNECT (required)
- ACCOUNT (required)
- NODE (required)
- FROM
- SUMMARYONLY (required)
- DIRECT_RESPONSE
- AUTOCLOSE_FYI
- ALLOW_FORWARDED_RESPONSE

Notification Mailer Configuration Parameters

- CONNECT: The <username/password>@<connect_string> to connect to the Workflow database account.
- ACCOUNT: The information to connect to the mail account. For MAPI-compliant mail programs, the account information is the mail account profile name and mail account password. For Sendmail, the account information would be the full file path of the mail spool file where incoming messages are stored.
- NODE: The identifier for this Notification Mailer. For example, you might use the server name as the node identifier. The node identifier for each Notification Mailer must be unique.
- FROM: The value that appears in the From: field of notifications.
- SUMMARYONLY: Y or N to send only notification summaries with this Notification Mailer.
- DIRECT_RESPONSE: Y to implement direct response processing for notifications sent to roles with MAILTEXT or MAILATTH only; N to implement to templated response processing.

- AUTOCLOSE_FYI: Y or N to automatically close notifications that do not require a response after sending the notification by e-mail.
- ALLOW_FORWARDED_RESPONSE: Y or N to allow a user to respond to an e-mail notification that has been forwarded (using e-mail) from another role.



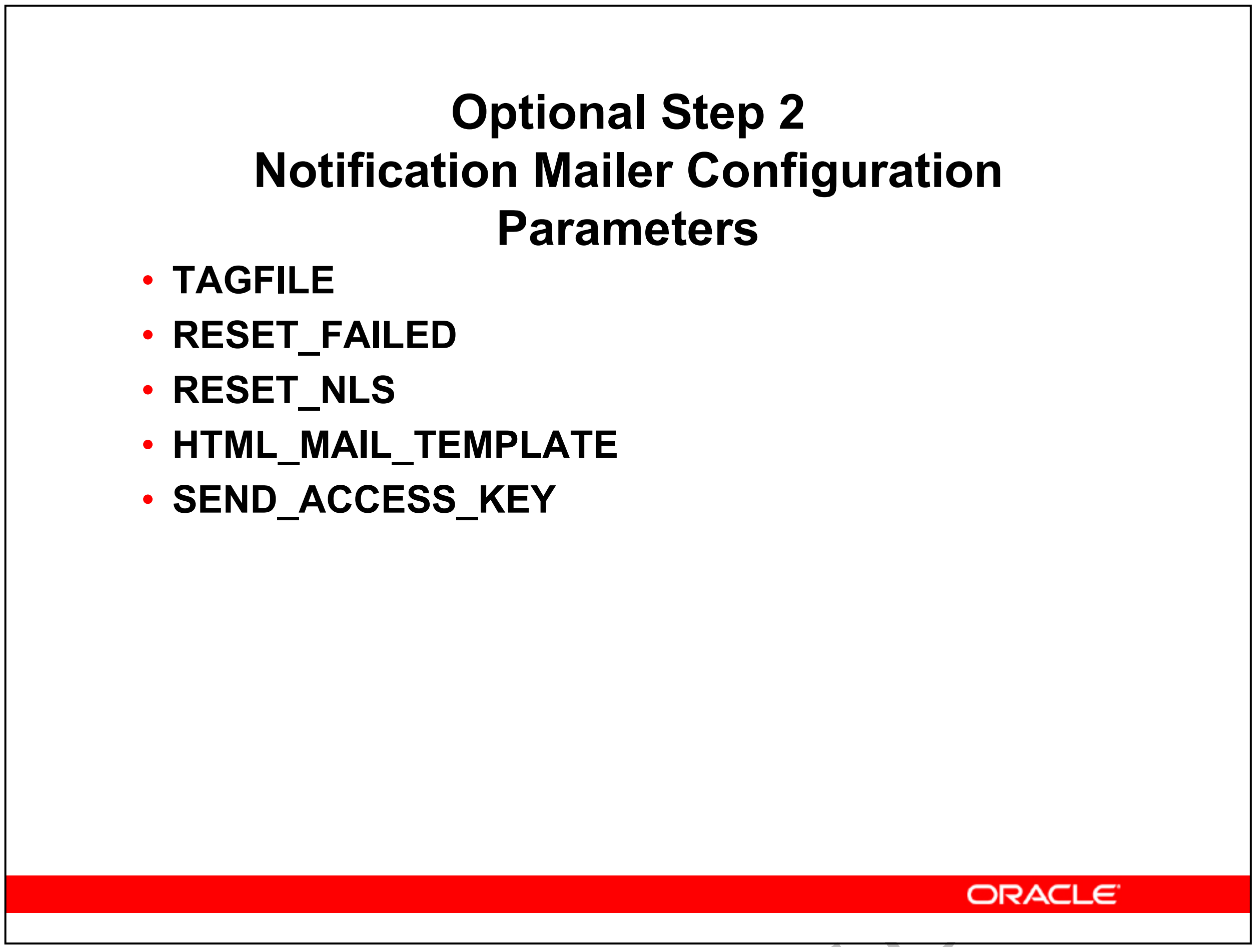
Optional Step 2 Notification Mailer Configuration Parameters

- IDLE
- LOG
- SHUTDOWN
- FAILCOMMAND
- DEBUG
- TEST_ADDRESS
- REPLYTO
- HTMLAGENT
- DISCARD
- PROCESS
- UNPROCESS

Notification Mailer Configuration Parameters

- IDLE: The number of seconds to wait before checking for messages to send.
- LOG: The name of a log file to record activity.
- SHUTDOWN: The name of a file that cues the Notification Mailer to shut down.
- FAILCOMMAND: The command to run if the Notification Mailer encounters a fatal error.
- DEBUG: Y or N to print debugging information in the log.
- TEST_ADDRESS: A test e-mail address where you want the Notification Mailer to direct all outgoing e-mail notifications. The test address overrides each recipient's e-mail address so that you can test a workflow process without having to change each recipient's e-mail address to access the test notifications.
- REPLYTO: A default e-mail address to reply to, if the e-mail account that processes responses is different from the e-mail account that sends outgoing notifications.
- HTMLAGENT: The base URL that identifies the HTML Web Agent that handles HTML notification responses.
- DISCARD: The name of mail folder or file for discarded incoming mail.

- PROCESS: The name of mail folder or file for processed responses.
- UNPROCESS: The name of mail folder or file for responses awaiting processing.



Optional Step 2 Notification Mailer Configuration Parameters

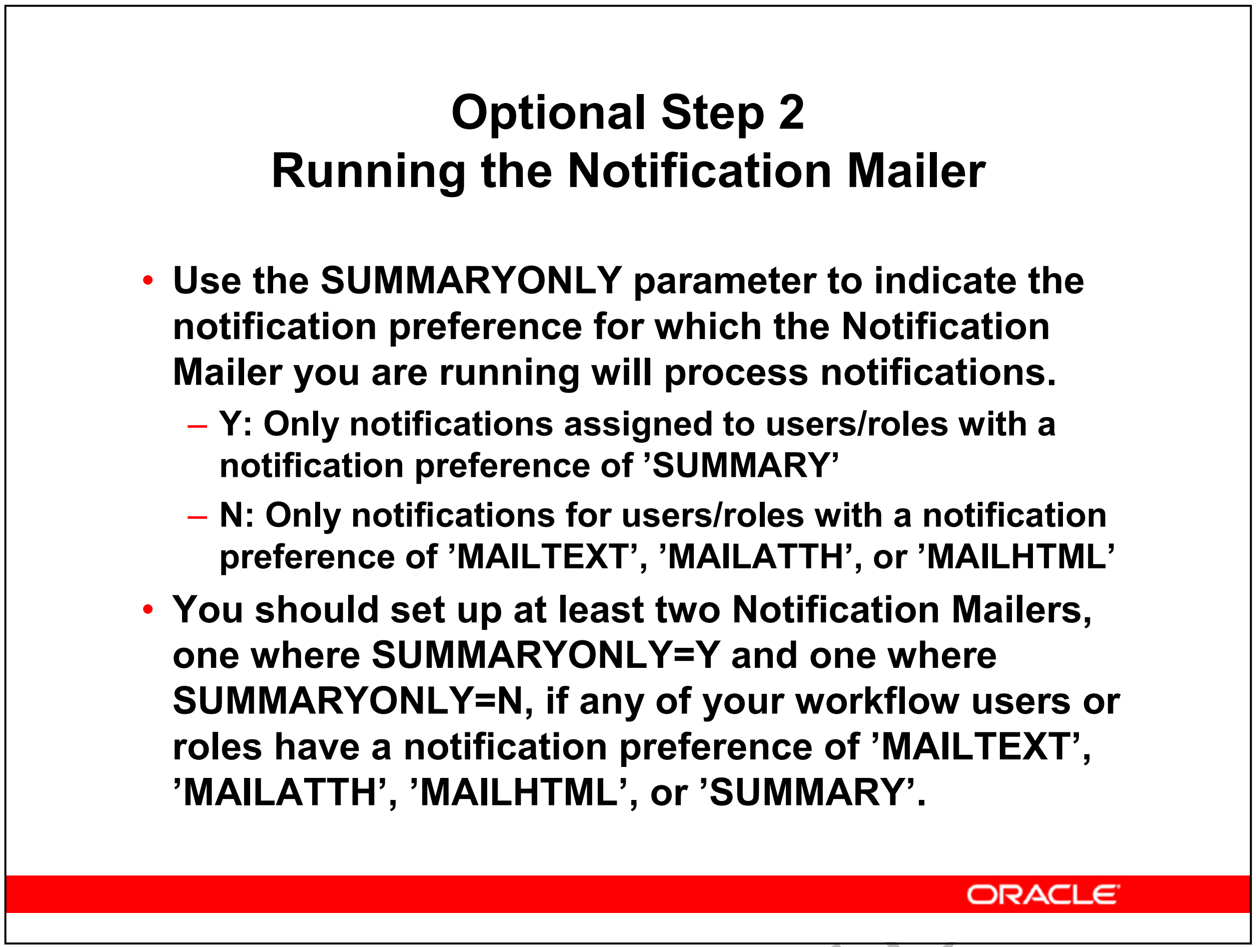
- TAGFILE
- RESET_FAILED
- RESET_NLS
- HTML_MAIL_TEMPLATE
- SEND_ACCESS_KEY

Notification Mailer Configuration Parameters

- TAGFILE: A file that lists strings found in unusual messages and the statuses to assign to those messages. For example, tag file entries might include:

```
- ERROR   "-- Unsent message follows --"
- UNAVAIL "Returned mail:"
- IGNORE  "POSTMASTER"
```

- RESET_FAILED: Y or N to reset all notifications with a mail status of FAILED to a mail status of MAIL when the Notification Mailer is started.
- RESET_NLS: Y or N to convert the NLS codeset for a notification message according to the notification recipient's preferences before composing the message.
- HTML_MAIL_TEMPLATE: Specify OPEN_MAIL_OUTLOOK to use the Workflow Open Mail for Outlook Express message as the template for e-mail notifications that require a response if you use Microsoft Outlook Express as your e-mail client.
- SEND_ACCESS_KEY: Y or N to include an access key in the Notification Detail Link attachment that is sent with HTML e-mail notifications and with plain text e-mail notifications with HTML attachments.



Optional Step 2 Running the Notification Mailer

- Use the SUMMARYONLY parameter to indicate the notification preference for which the Notification Mailer you are running will process notifications.
  - Y: Only notifications assigned to users/roles with a notification preference of ’SUMMARY’
  - N: Only notifications for users/roles with a notification preference of ’MAILTEXT’, ’MAILATTH’, or ’MAILHTML’
- You should set up at least two Notification Mailers, one where SUMMARYONLY=Y and one where SUMMARYONLY=N, if any of your workflow users or roles have a notification preference of ’MAILTEXT’, ’MAILATTH’, ’MAILHTML’, or ’SUMMARY’.

Notification Mailer

If you set SUMMARYONLY=Y, then the Notification Mailer will shut itself down after it polls the database and delivers any appropriate notification summaries. You must therefore schedule the Notification Mailer to run at the frequency you want notification summaries to be delivered.

It is recommended that you run the summary Notification Mailer once a day, since the summary includes all open notifications. For Oracle Workflow running in the standalone environment, this would involve creating an operating system script, such as a cron job in UNIX, to schedule the Notification Mailer. For the version of Oracle Workflow embedded in Oracle E-Business Suite, this simply involves scheduling the Notification Mailer concurrent program in the Submit Request form.

Optional Step 3 Modifying Message Templates

Optional Step 3
Modifying Message Templates

- Messages sent by the Notification Mailer are composed using the templates defined in the System: Mailer item type.
- The System: Mailer item type is defined in the wfmail.wft workflow definition file.
- You can modify the templates to configure the e-mail messages sent to users.

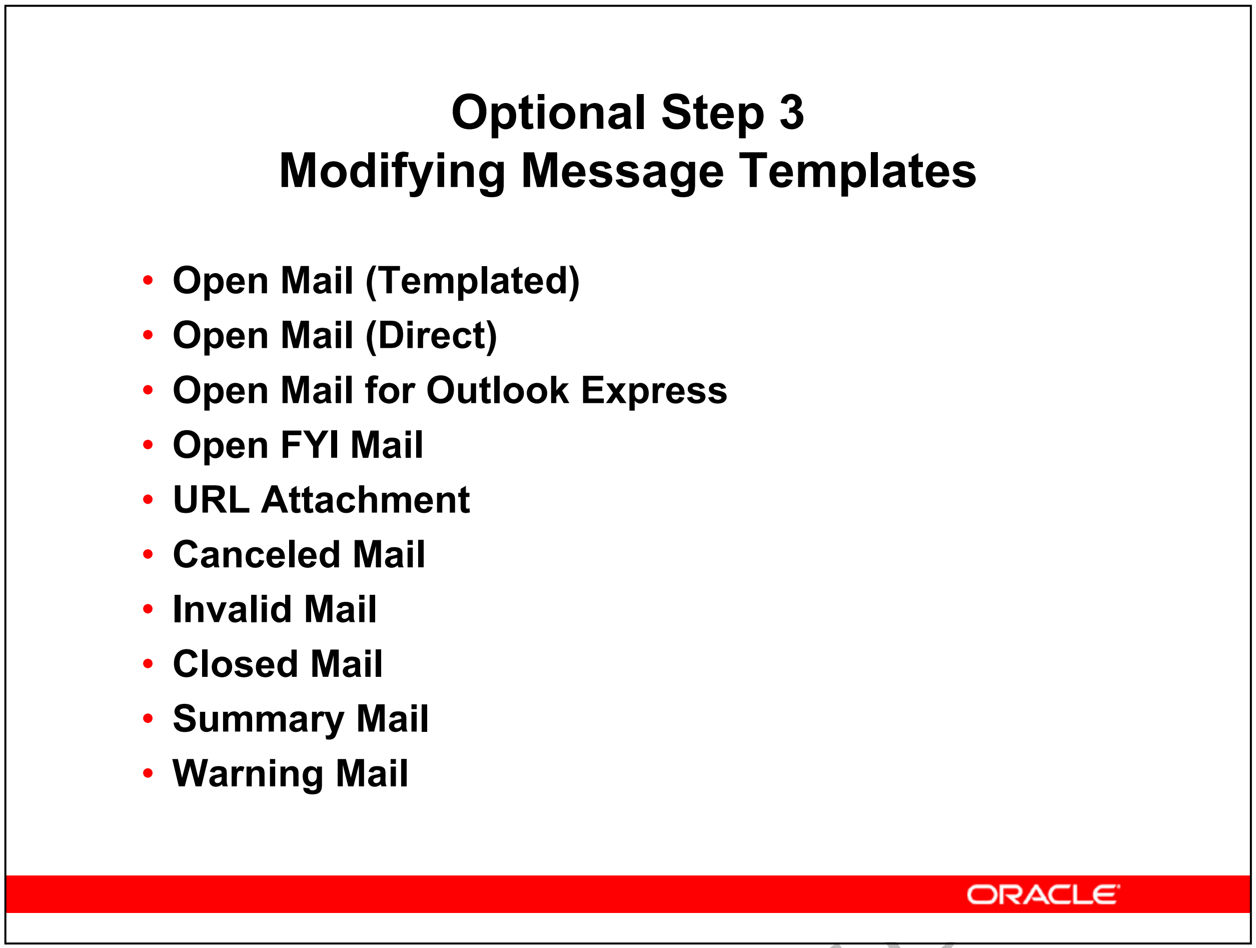
ORACLE

Modifying Message Templates

Note: Always use flat file data stores (.wft files) for edits rather than editing directly against a database data store. Always source control your workflow definitions.

1. Copy wfmail.wft and store the new file in your source-controlled file system.
2. Open your new copy of the System: Mailer item type in Oracle Workflow Builder.
3. Select one of the message templates.
4. Display the property sheet for the message.
5. Edit the subject or body of the message.
6. Do not add, delete, or modify any attributes in the System: Mailer item type.
7. Verify your edits using File > Verify, and save your work.
8. Source control your edited version of the System: Mailer item type.
9. Load your modified System: Mailer item type to your database using the Workflow Loader program or the Save As menu option in Oracle Workflow Builder.
10. Test your modified templates using the test account capability of the Notification Mailer.

Optional Step 3 Modifying Message Templates



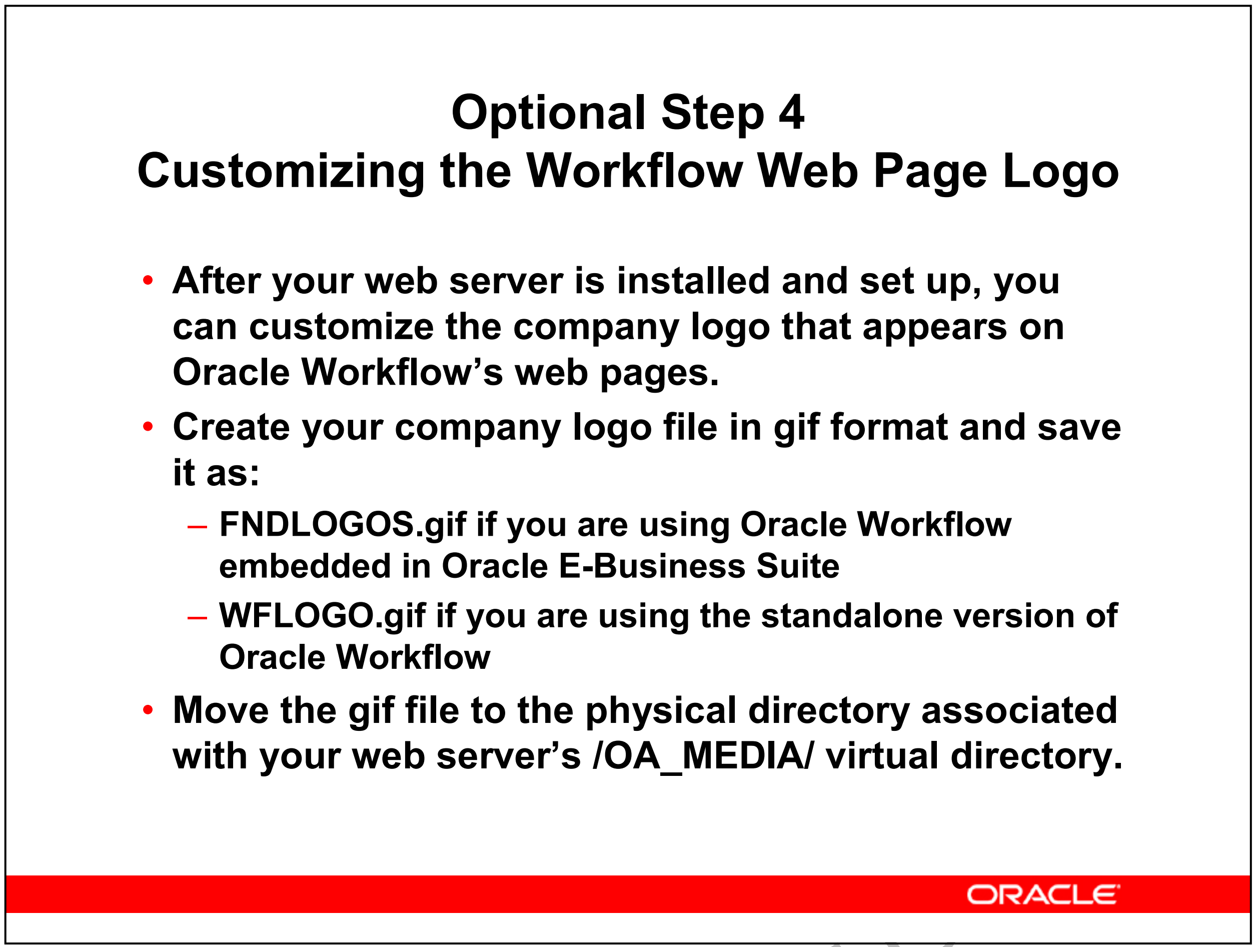
Optional Step 3 Modifying Message Templates

- Open Mail (Templated)
- Open Mail (Direct)
- Open Mail for Outlook Express
- Open FYI Mail
- URL Attachment
- Canceled Mail
- Invalid Mail
- Closed Mail
- Summary Mail
- Warning Mail

ORACLE

Mail Templates

- Open Mail (Templated): For notifications that require a response when you are using the templated response method
- Open Mail (Direct): For notifications that require a response when you are using the direct response method
- Open Mail for Outlook Express: For notifications that require a response, if you use an e-mail application such as Microsoft Outlook Express as your e-mail client
- Open FYI Mail: For notifications that do not require a response
- URL Attachment: Creates the Notification References attachment for HTML-formatted notification messages that include URL attributes with Attach Content checked
- Canceled Mail: Informs recipient that a notification is canceled
- Invalid Mail: Informs recipient that the response to the notification is invalid
- Closed Mail: Informs recipient that a previously sent notification is now closed
- Summary Mail: For notification summaries
- Warning Mail: Informs recipient of unsolicited mail that he or she sent

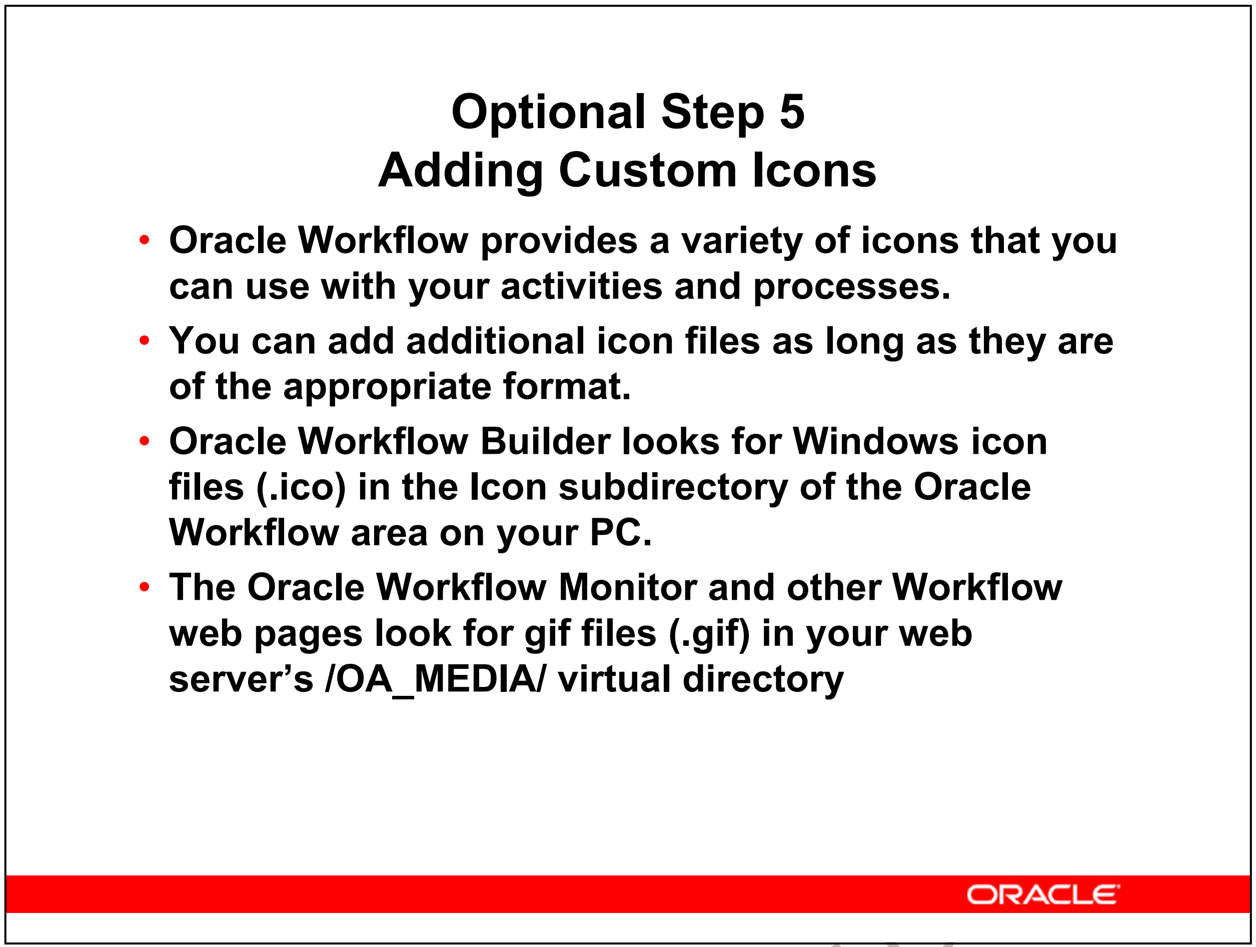


Optional Step 4
Customizing the Workflow Web Page Logo

- **After your web server is installed and set up, you can customize the company logo that appears on Oracle Workflow's web pages.**
- **Create your company logo file in gif format and save it as:**
  - **FNDLOGOS.gif if you are using Oracle Workflow embedded in Oracle E-Business Suite**
  - **WFLOGO.gif if you are using the standalone version of Oracle Workflow**
- **Move the gif file to the physical directory associated with your web server's /OA_MEDIA/ virtual directory.**

Customizing the Workflow Web Page Logo

Note: /OA_MEDIA/ is a virtual directory mapping defined in your Web server when Oracle Workflow is installed.



Optional Step 5
Adding Custom Icons

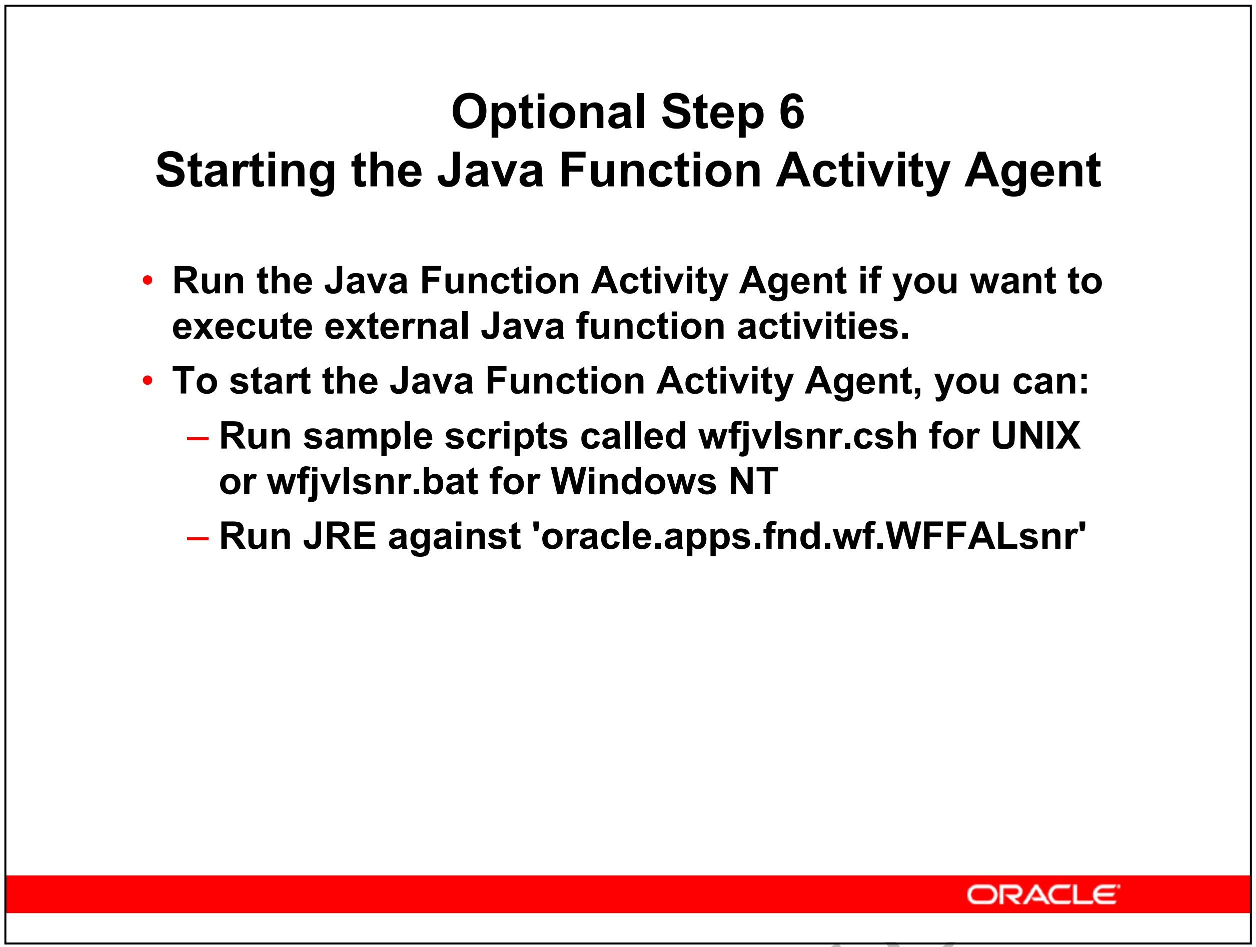
- Oracle Workflow provides a variety of icons that you can use with your activities and processes.
- You can add additional icon files as long as they are of the appropriate format.
- Oracle Workflow Builder looks for Windows icon files (.ico) in the Icon subdirectory of the Oracle Workflow area on your PC.
- The Oracle Workflow Monitor and other Workflow web pages look for gif files (.gif) in your web server's /OA_MEDIA/ virtual directory

Adding Custom Icons to Oracle Workflow

If you create custom icons to include in your Oracle Workflow Builder process definition, and you want the custom icons to appear in the Workflow Monitor when you view the process, you must do the following:

1. Convert the custom icon files (.ico) to gif format (.gif).
2. Copy the .gif files to the physical directory associated with your web server's /OA_MEDIA/ virtual directory, so that the Workflow Monitor can access them.

Note: /OA_MEDIA/ is a virtual directory mapping defined in your Web server when Oracle Workflow is installed.



Optional Step 6
Starting the Java Function Activity Agent

- Run the Java Function Activity Agent if you want to execute external Java function activities.
- To start the Java Function Activity Agent, you can:
  - Run sample scripts called wfjvlsnr.csh for UNIX or wfjvlsnr.bat for Windows NT
  - Run JRE against 'oracle.apps.fnd.wf.WFFALsnr'

ORACLE

Starting the Java Function Activity Agent

To run the Java Function Activity Agent, you must have Java Runtime Environment (JRE) Version 1.1.8, or a higher 1.1.x version, installed.

Note: The Java Function Activity Agent is currently only available for the standalone version of Oracle Workflow. This functionality is not currently available for the version of Oracle Workflow embedded in Oracle E-Business Suite.

Starting the Java Function Activity Agent From a Script

If you are using the standalone version of Oracle Workflow, you can run scripts provided by Oracle Workflow to start the Java Function Activity agent. The scripts are called wfjvlsnr.csh, for UNIX, and wfjvlsnr.bat, for Windows NT, and are located on your server in the Oracle Workflow *admin* subdirectory. If you define your own external Java function activities, you must include the path to the JAR files containing your custom Java classes in the scripts. Then you can run the scripts with the user name and password of your Oracle Workflow database account to submit the Java Function Activity Agent.

Starting the Java Function Activity Agent Manually

To start the Java Function Activity Agent manually, run JRE against 'oracle.apps.fnd.wf.WFFALsnr', specifying your CLASSPATH and the user name and password of your Oracle Workflow database account. The CLASSPATH must point to the Java Runtime Environment, the top-level Workflow Java directory, the Oracle XML parser, the Oracle JDBC implementation, and the following Workflow JAR files:

- wfjava.jar—The Java Function Activity Agent
- wfapi.jar—Workflow Java APIs
- The Share JAR file—Utilities referenced by the Workflow Java APIs
- The Ewt JAR file—Utilities referenced by the Workflow Java APIs
- The Swing JAR file—Optional additional utilities

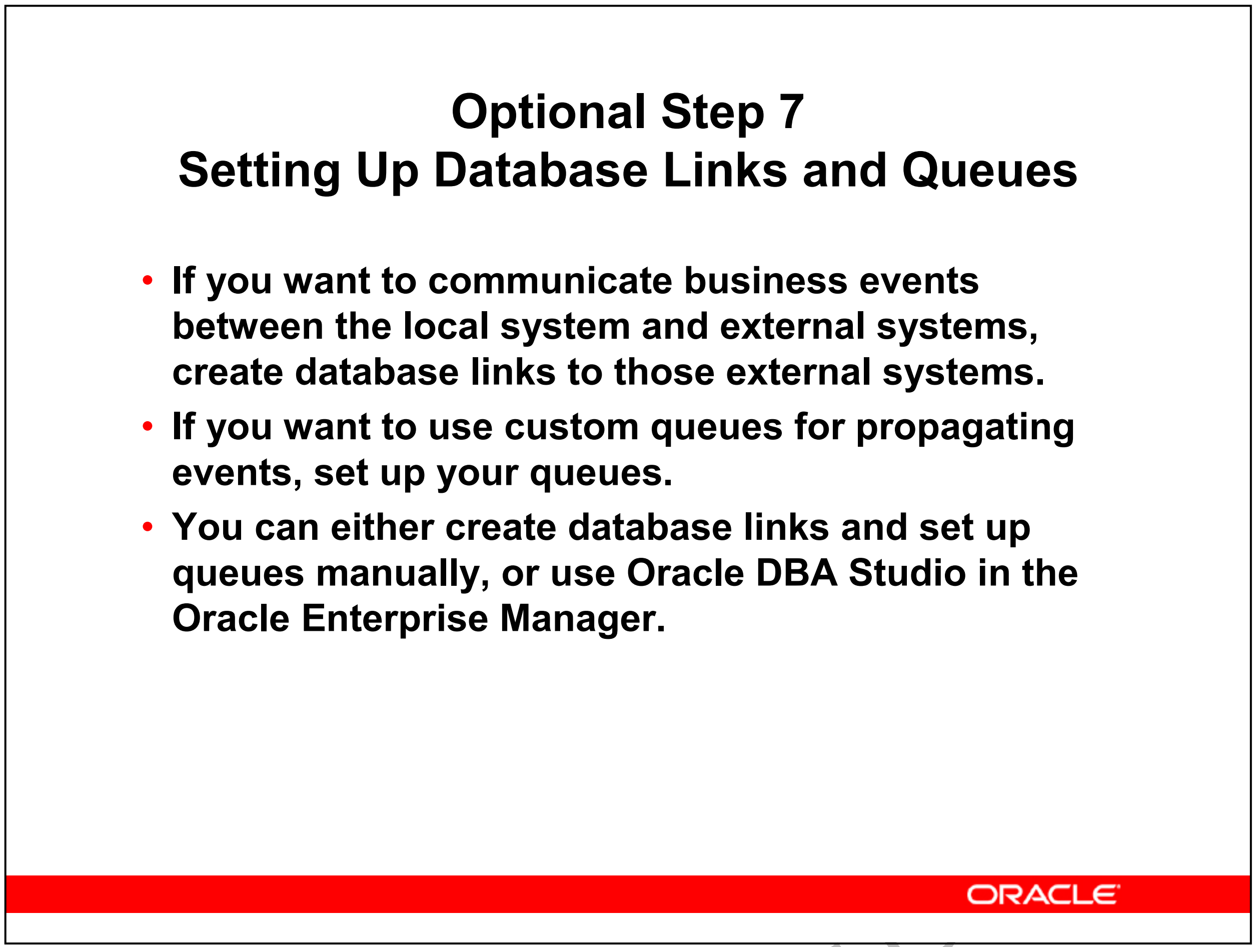
If you define your own external Java function activities, you must also include the JAR files containing your custom Java classes in the CLASSPATH.

Stopping the Java Function Activity Agent

Normally, the Java Function Activity Agent runs as a perpetual job. However, you can stop the agent by running a script called wfjvstop.sql, located in the *admin/sql* subdirectory on your Oracle Workflow server.

For more information, refer to the Setting Up Oracle Workflow chapter in the *Oracle Workflow Guide*.

Optional Step 7 Setting Up Database Links and Queues



Optional Step 7
Setting Up Database Links and Queues

- If you want to communicate business events between the local system and external systems, create database links to those external systems.
- If you want to use custom queues for propagating events, set up your queues.
- You can either create database links and set up queues manually, or use Oracle DBA Studio in the Oracle Enterprise Manager.

Setting Up Database Links and Queues

Database Links

When you set up database links for use by the Business Event System, you should fully qualify each database link name with the domain name.

Queues

Oracle Workflow provides four standard queues that you can use for event processing:

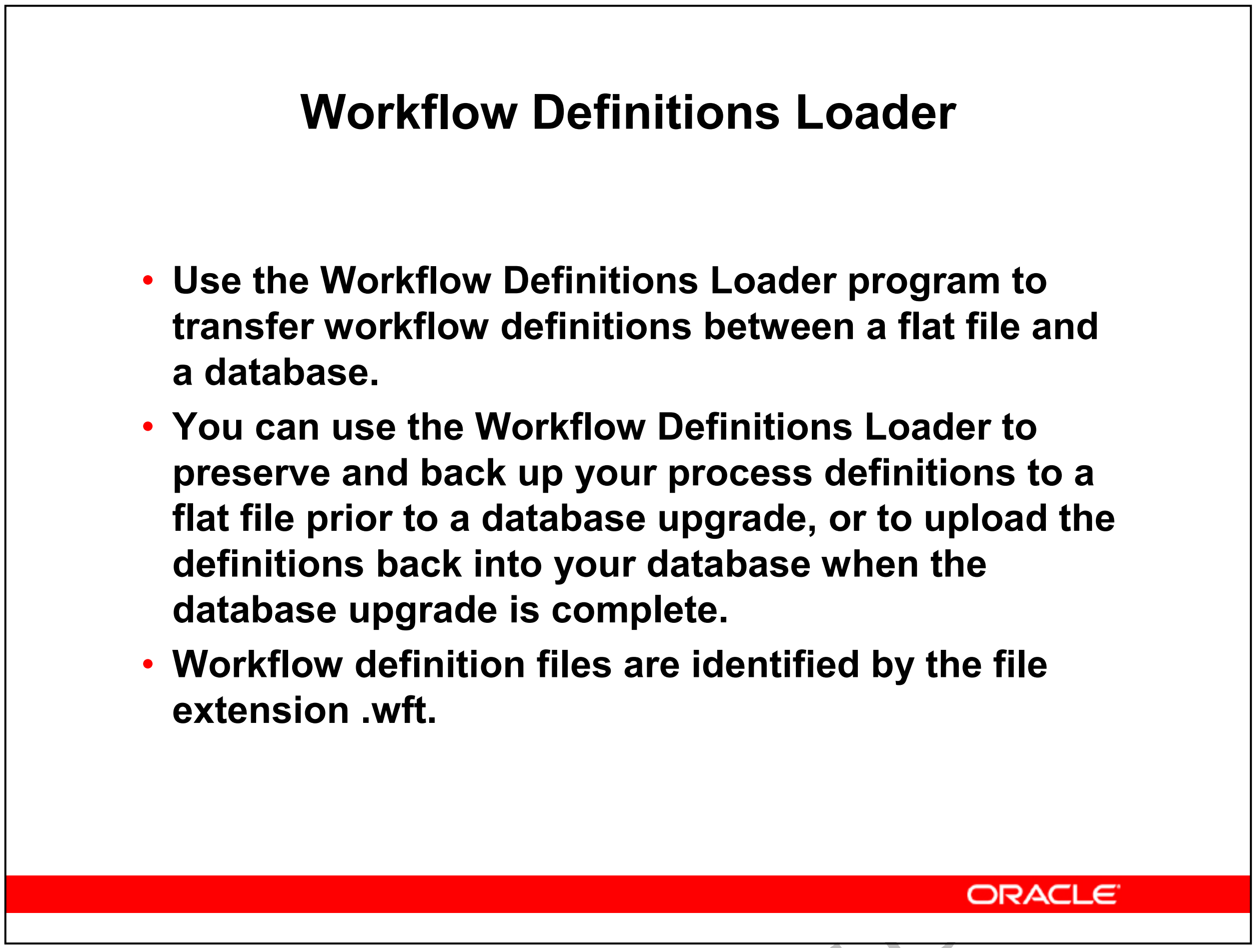
- WF_IN
- WF_OUT
- WF_DEFERRED
- WF_ERROR

You must not change the setup of these queues. However, you must schedule listeners for WF_DEFERRED and WF_ERROR to enable deferred subscription processing and error handling for the Business Event System, respectively. Also, if you want to use WF_IN and WF_OUT for event message propagation, schedule a listener for WF_IN and a propagation for WF_OUT as well.

You can also set up your own queues for event message propagation. To set up a queue, you must create the queue table, create the queue, and start the queue.

Oracle Workflow provides a sample script called wfevquec.sql which you can modify to set up your queues, as well as a sample script called wfevqued.sql which you can modify to drop queues. These scripts are located on your server in the Oracle Workflow *sql* subdirectory for the standalone version of Oracle Workflow, or in the *sql* subdirectory under $FND_TOP for the version of Oracle Workflow embedded in Oracle E-Business Suite.

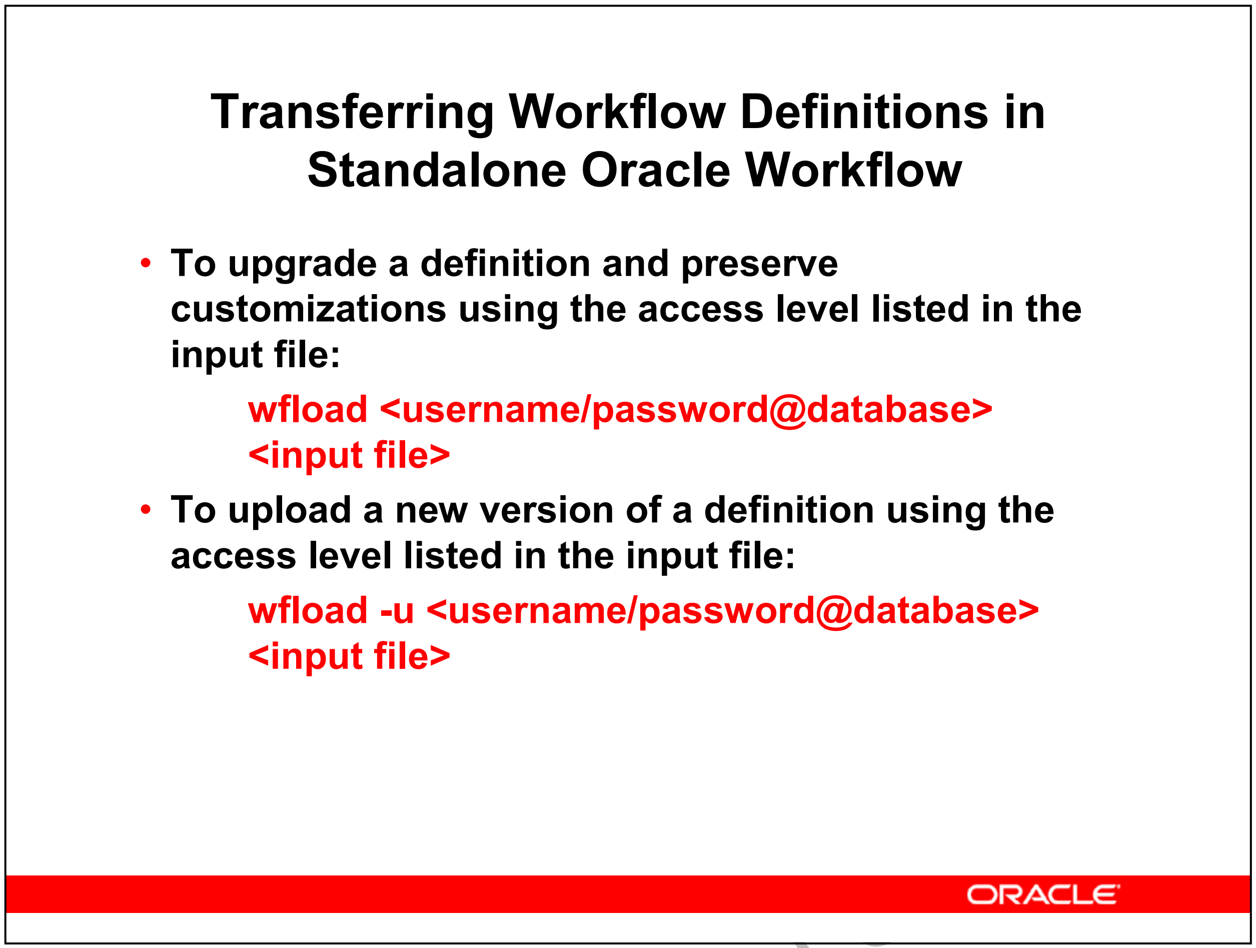
If you define a queue with a payload type other than the standard WF_EVENT_T Workflow format, you must create a queue handler to translate between WF_EVENT_T and the format required by the queue.



Workflow Definitions Loader

- Use the Workflow Definitions Loader program to transfer workflow definitions between a flat file and a database.
- You can use the Workflow Definitions Loader to preserve and back up your process definitions to a flat file prior to a database upgrade, or to upload the definitions back into your database when the database upgrade is complete.
- Workflow definition files are identified by the file extension .wft.

ORACLE



Transferring Workflow Definitions in Standalone Oracle Workflow

- To upgrade a definition and preserve customizations using the access level listed in the input file:

  ```
  wfload <username/password@database>
  <input file>
  ```

- To upload a new version of a definition using the access level listed in the input file:

  ```
  wfload -u <username/password@database>
  <input file>
  ```

ORACLE

Transferring Workflow Definitions

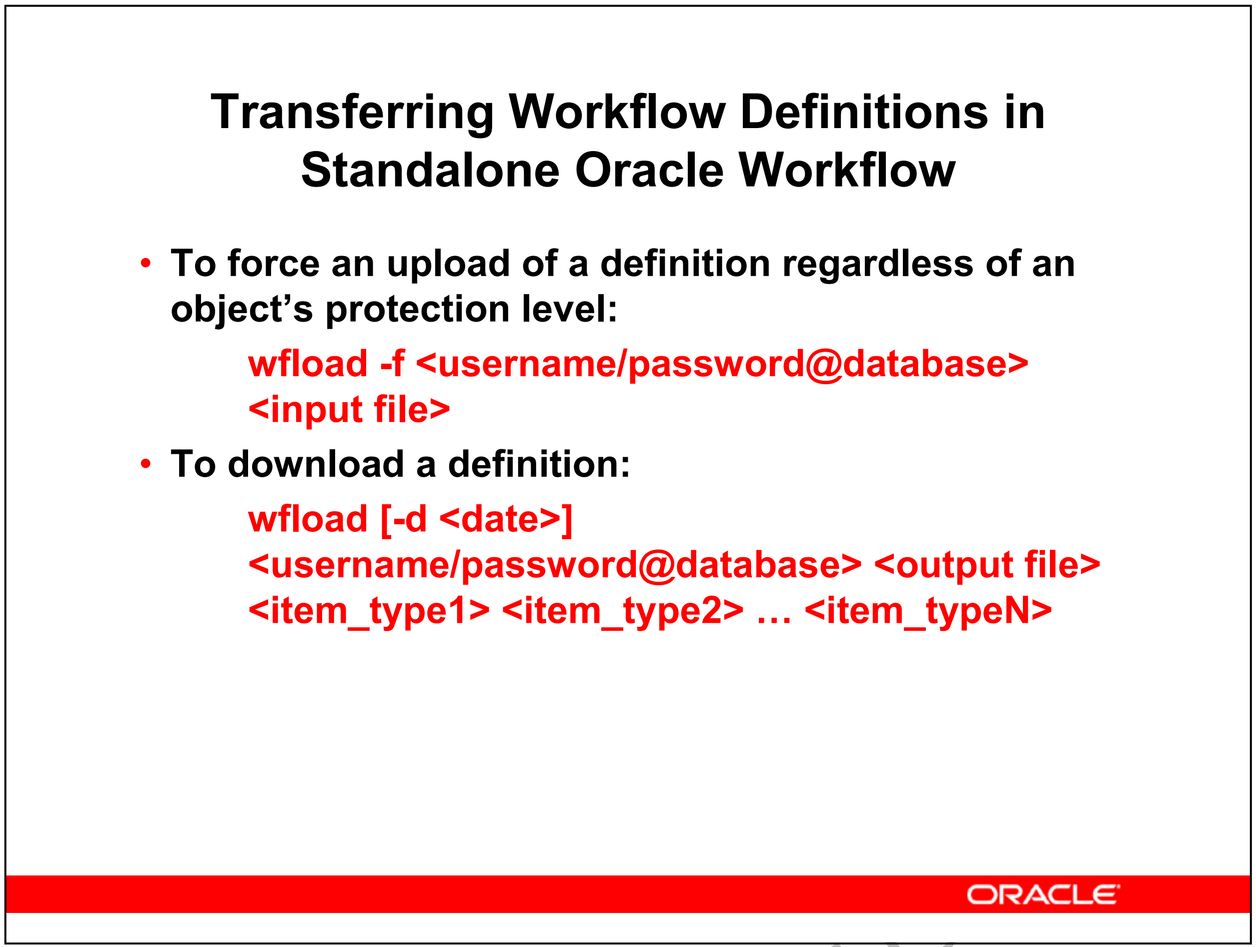
The Workflow Definitions Loader program is located on your server in the bin subdirectory of the Oracle Home directory structure.

Upgrade Behavior

The Workflow Definitions Loader assumes the access level of the .wft file and overwrites any objects protected at a level equal to or above the upgrade file's access level (possibly lowering the protection level). During an upgrade, the Loader program preserves any customizations made to customizable seed data in the database.

Upload Behavior

The upload mode is useful to someone who is developing a workflow process. It allows the developer to save definitions to the database without concern that accidental customizations to existing objects might prevent the upload of some process definition elements. (Upload mode essentially ignores the Preserve Customizations setting on an object.) The Workflow Definitions Loader uses the access level specified in the input file.



Transferring Workflow Definitions in Standalone Oracle Workflow

- To force an upload of a definition regardless of an object's protection level:

```
wfload -f <username/password@database>
<input file>
```

- To download a definition:

```
wfload [-d <date>]
<username/password@database> <output file>
<item_type1> <item_type2> … <item_typeN>
```

ORACLE

Transferring Workflow Definitions

Force Behavior

When using the force option, you should be certain that the process definition in the file is correct, because it overwrites the entire process stored in the database. The force option is useful for fixing data integrity problems in a database with a known, reliable file backup.

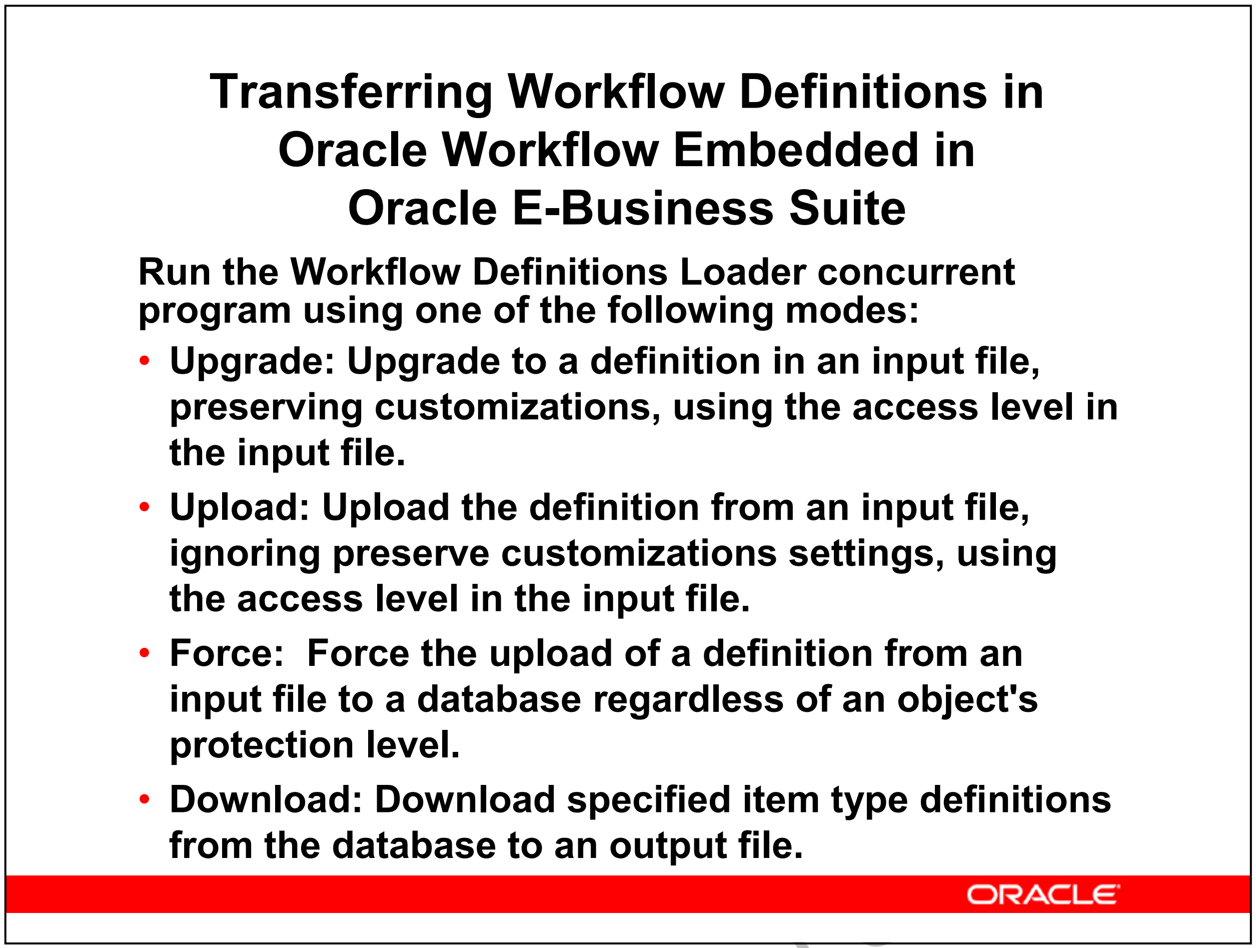
Download Behavior

Download one or more item type definitions from a database to a flat file.

- Replace <item_typeN> with an asterisk (*) to download all item types.
- Specify the -d option with a date to download definitions that were effective at a specific date. Use the format YYYY/MM/DD HH24:MI:SS to specify the effective date.

The access level of the resulting output file is set to the value stored in the WF_ACCESS_LEVEL environment variable.

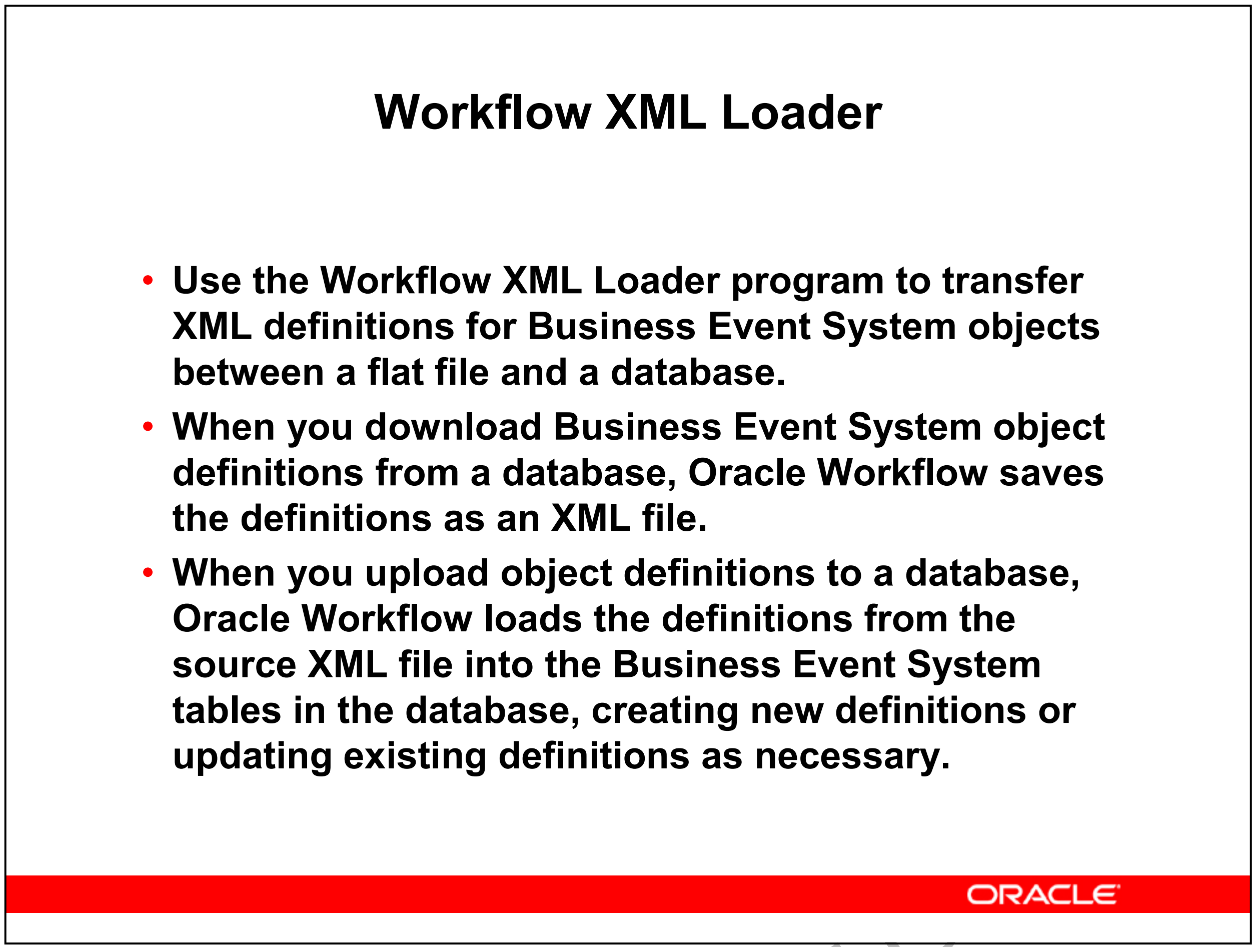
Transferring Workflow Definitions in Oracle Workflow Embedded in Oracle E-Business Suite



Transferring Workflow Definitions in Oracle Workflow Embedded in Oracle E-Business Suite

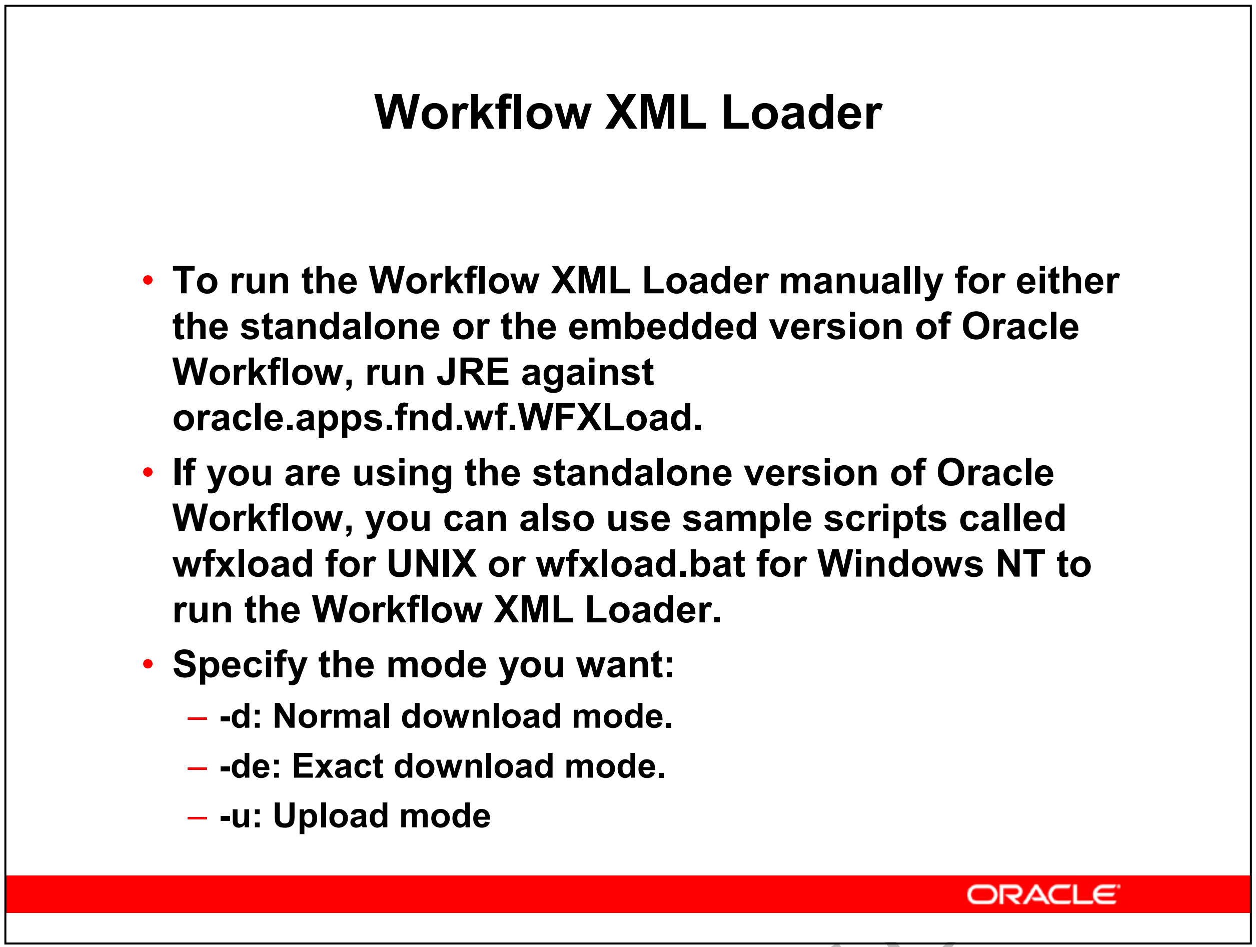
Run the Workflow Definitions Loader concurrent program using one of the following modes:

- **Upgrade: Upgrade to a definition in an input file, preserving customizations, using the access level in the input file.**
- **Upload: Upload the definition from an input file, ignoring preserve customizations settings, using the access level in the input file.**
- **Force: Force the upload of a definition from an input file to a database regardless of an object's protection level.**
- **Download: Download specified item type definitions from the database to an output file.**



Workflow XML Loader

- Use the Workflow XML Loader program to transfer XML definitions for Business Event System objects between a flat file and a database.
- When you download Business Event System object definitions from a database, Oracle Workflow saves the definitions as an XML file.
- When you upload object definitions to a database, Oracle Workflow loads the definitions from the source XML file into the Business Event System tables in the database, creating new definitions or updating existing definitions as necessary.



Workflow XML Loader

- To run the Workflow XML Loader manually for either the standalone or the embedded version of Oracle Workflow, run JRE against oracle.apps.fnd.wf.WFXLoad.
- If you are using the standalone version of Oracle Workflow, you can also use sample scripts called wfxload for UNIX or wfxload.bat for Windows NT to run the Workflow XML Loader.
- Specify the mode you want:
  - -d: Normal download mode.
  - -de: Exact download mode.
  - -u: Upload mode

ORACLE

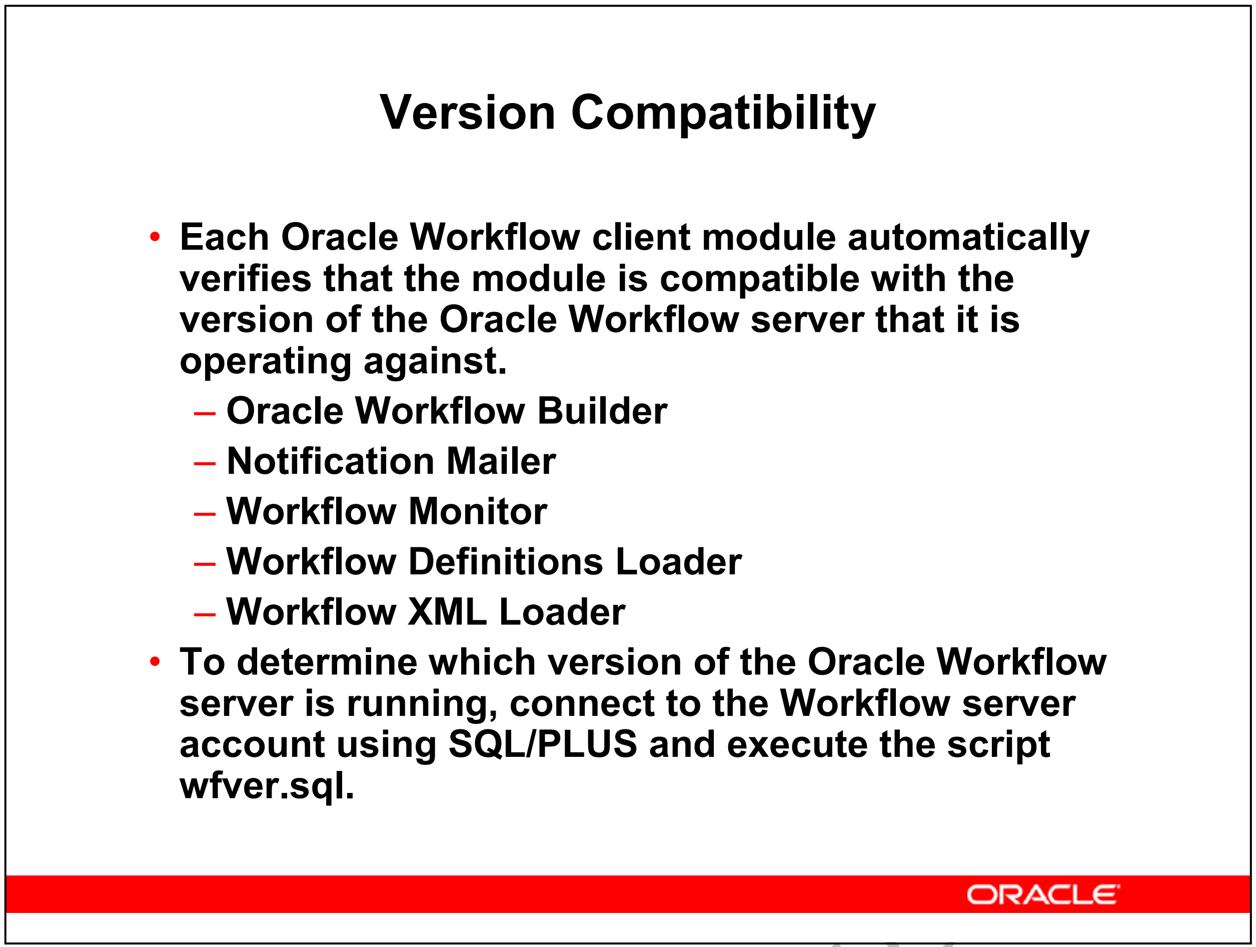
Workflow XML Loader

To run the Workflow XML Loader, you must have Java Runtime Environment (JRE) Version 1.1.8, or a higher 1.1.x version, installed.

You can download Business Event System object definitions in either normal download mode or exact download mode.

- Normal download mode: Lets you save a generic copy of object definitions from one system that you can use to create similar definitions in other systems. In this mode, the Workflow XML Loader replaces certain system-specific data within the object definitions with tokens. Choose normal download mode, for example, when you want to save Business Event System object definitions from a development system as seed data that can be uploaded to a production system.
- Exact download mode: Lets you save object definitions exactly as they are specified in the database. In this mode, the Workflow XML Loader does not convert any data to tokens; instead, all values, including system-specific values, are copied to the XML file. Choose exact download mode, for example, when you want to save Business Event System object definitions from one production system so that you can replicate them to another production system that communicates with the first.

For detailed information on the commands to run the Workflow XML Loader, refer to the Setting Up Oracle Workflow chapter in the *Oracle Workflow Guide*.



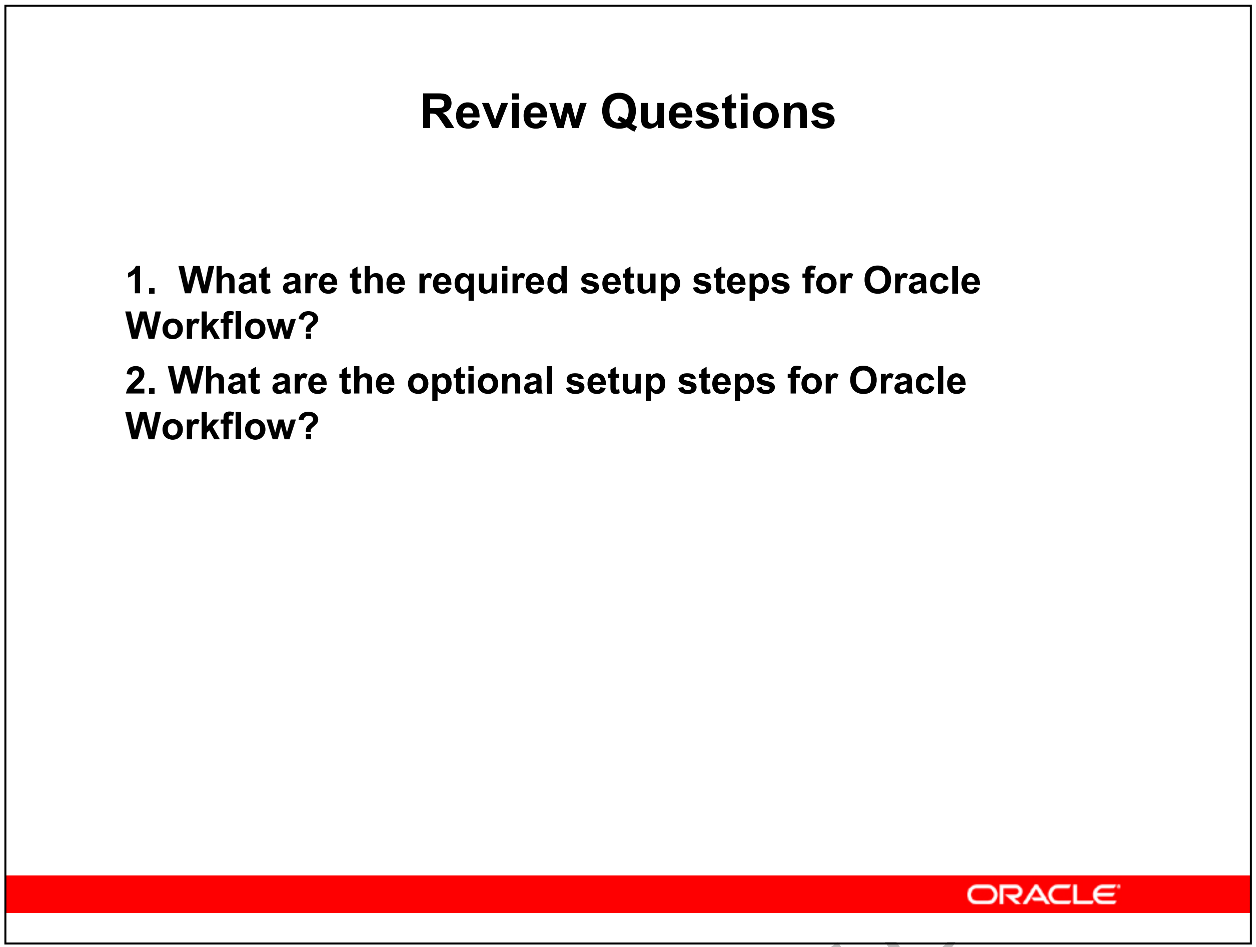
Version Compatibility

- Each Oracle Workflow client module automatically verifies that the module is compatible with the version of the Oracle Workflow server that it is operating against.
  - Oracle Workflow Builder
  - Notification Mailer
  - Workflow Monitor
  - Workflow Definitions Loader
  - Workflow XML Loader
- To determine which version of the Oracle Workflow server is running, connect to the Workflow server account using SQL/PLUS and execute the script wfver.sql.

ORACLE

Review Questions

1. **What are the required setup steps for Oracle Workflow?**
2. **What are the optional setup steps for Oracle Workflow?**



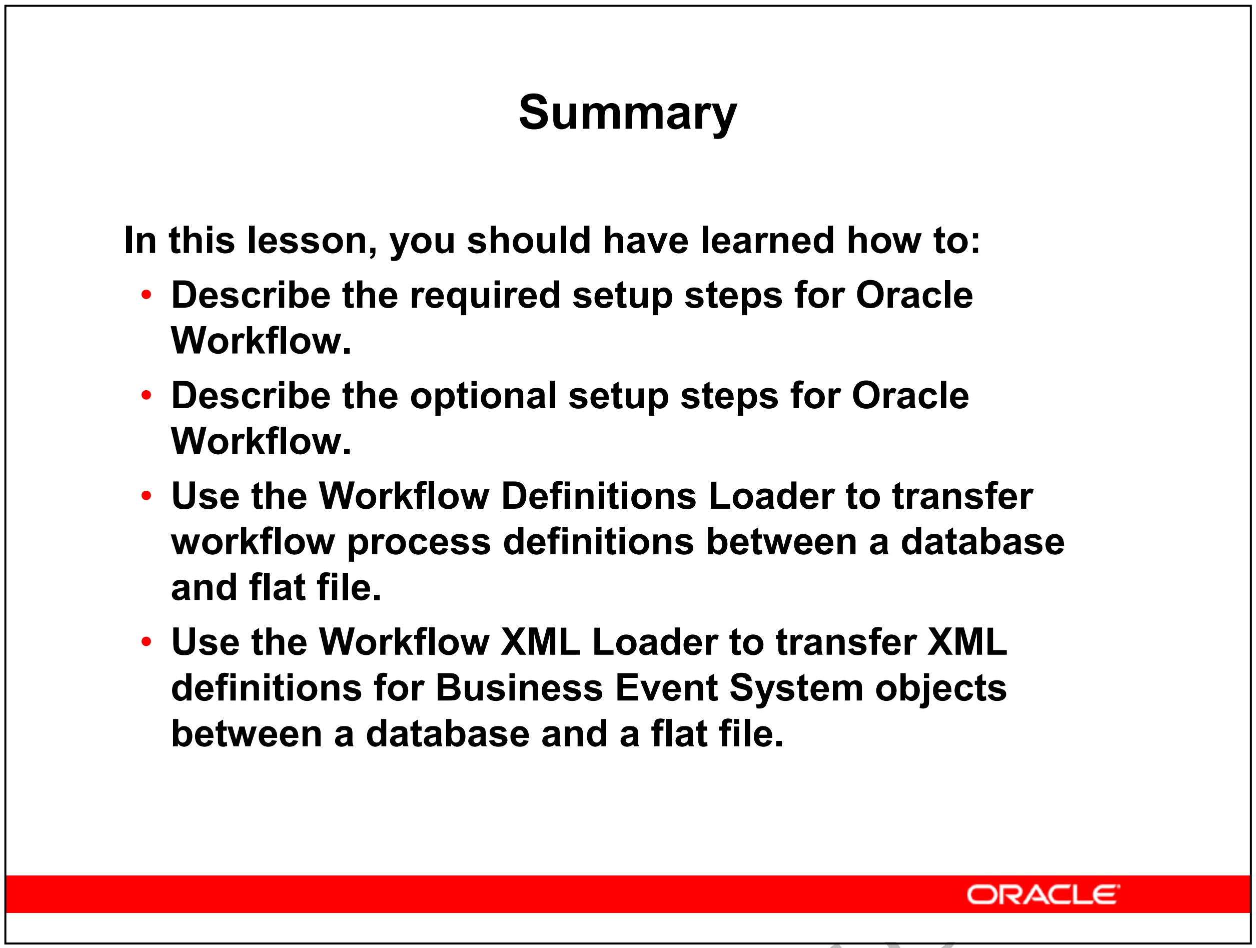
Review Questions

1. What are the required setup steps for Oracle Workflow?
2. What are the optional setup steps for Oracle Workflow?

Review Questions and Solutions

1. What are the required setup steps for Oracle Workflow?
   - **Set your global workflow preferences.**
   - **Map Oracle Workflow's directory service to your organization's directory repository.**
   - **Create a view called WF_LANGUAGES that identifies the languages defined in your Oracle database installation.**
   - **If you are using Oracle Workflow embedded in Oracle E-Business Suite, set the Socket Listener Activated profile option.**
   - **If you are using the standalone version of Oracle Workflow and your Workflow server is installed on a UNIX platform, define an environment variable called WF_RESOURCES.**
   - **Set up background Workflow engines.**
2. What are the optional setup steps for Oracle Workflow?
   - **Partition Workflow tables for performance gain.**
   - **Run the Notification Mailer program.**

- **Modify the templates for your electronic mail notifications.**
- **Customize the company logo that appears in Oracle Workflow’s web pages.**
- **Include custom icons in your Oracle Workflow icons subdirectory.**
- **Start the Java Function Activity Agent.**
- **Set up database links and queues for the Business Event System.**



Summary

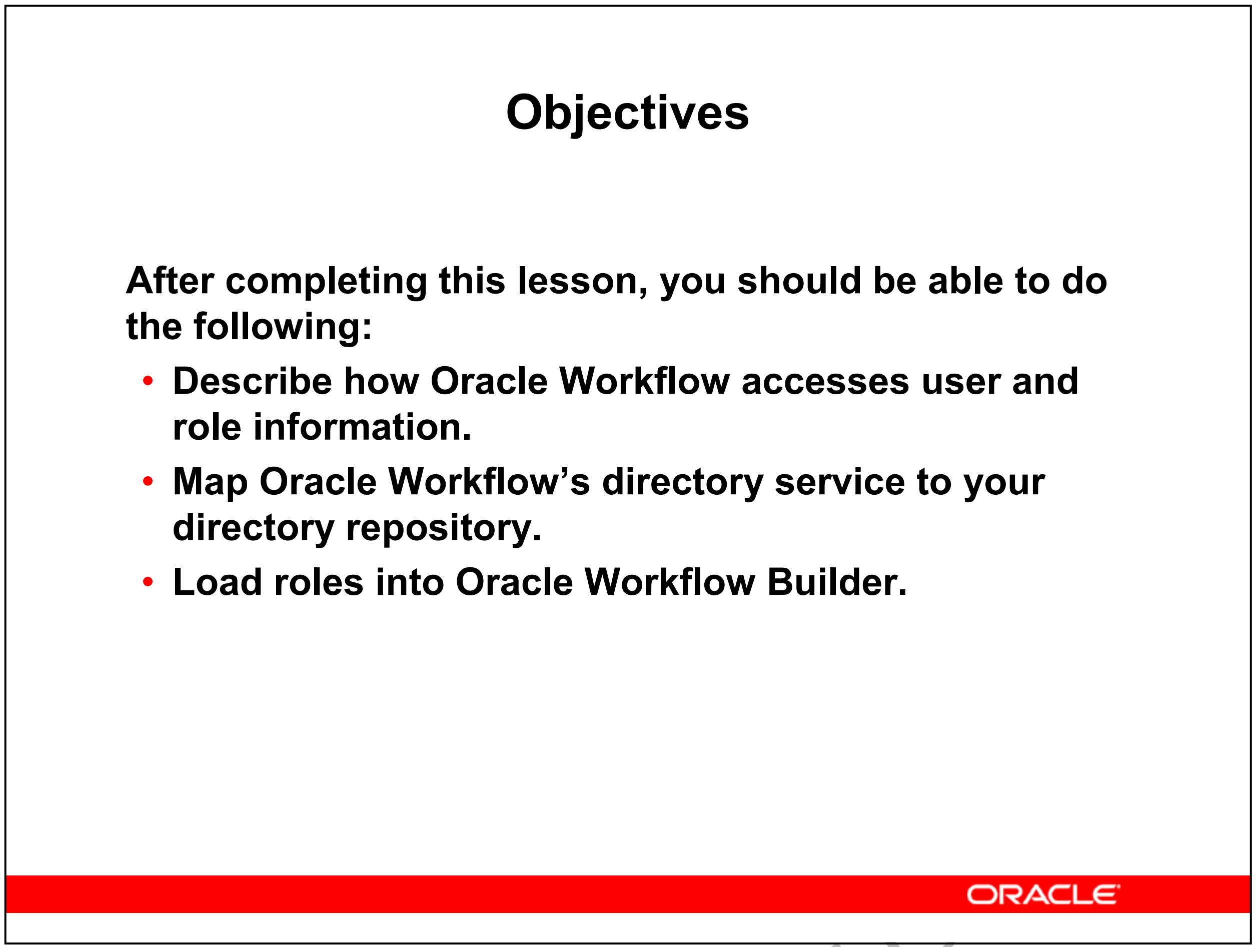
In this lesson, you should have learned how to:

- Describe the required setup steps for Oracle Workflow.
- Describe the optional setup steps for Oracle Workflow.
- Use the Workflow Definitions Loader to transfer workflow process definitions between a database and flat file.
- Use the Workflow XML Loader to transfer XML definitions for Business Event System objects between a database and a flat file.

Oracle Workflow Directory Service

Chapter 25

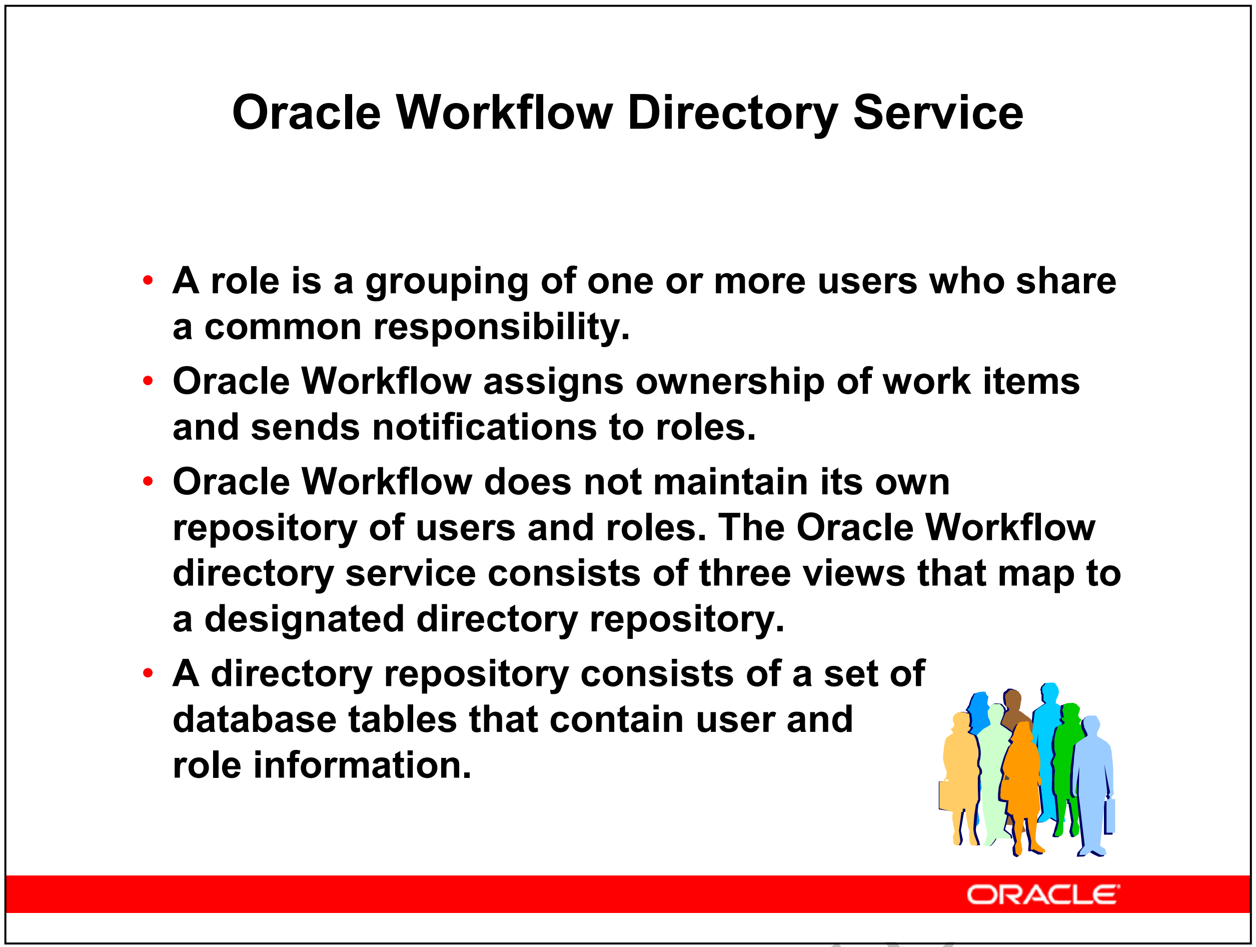
Oracle Workflow Directory Service



Objectives

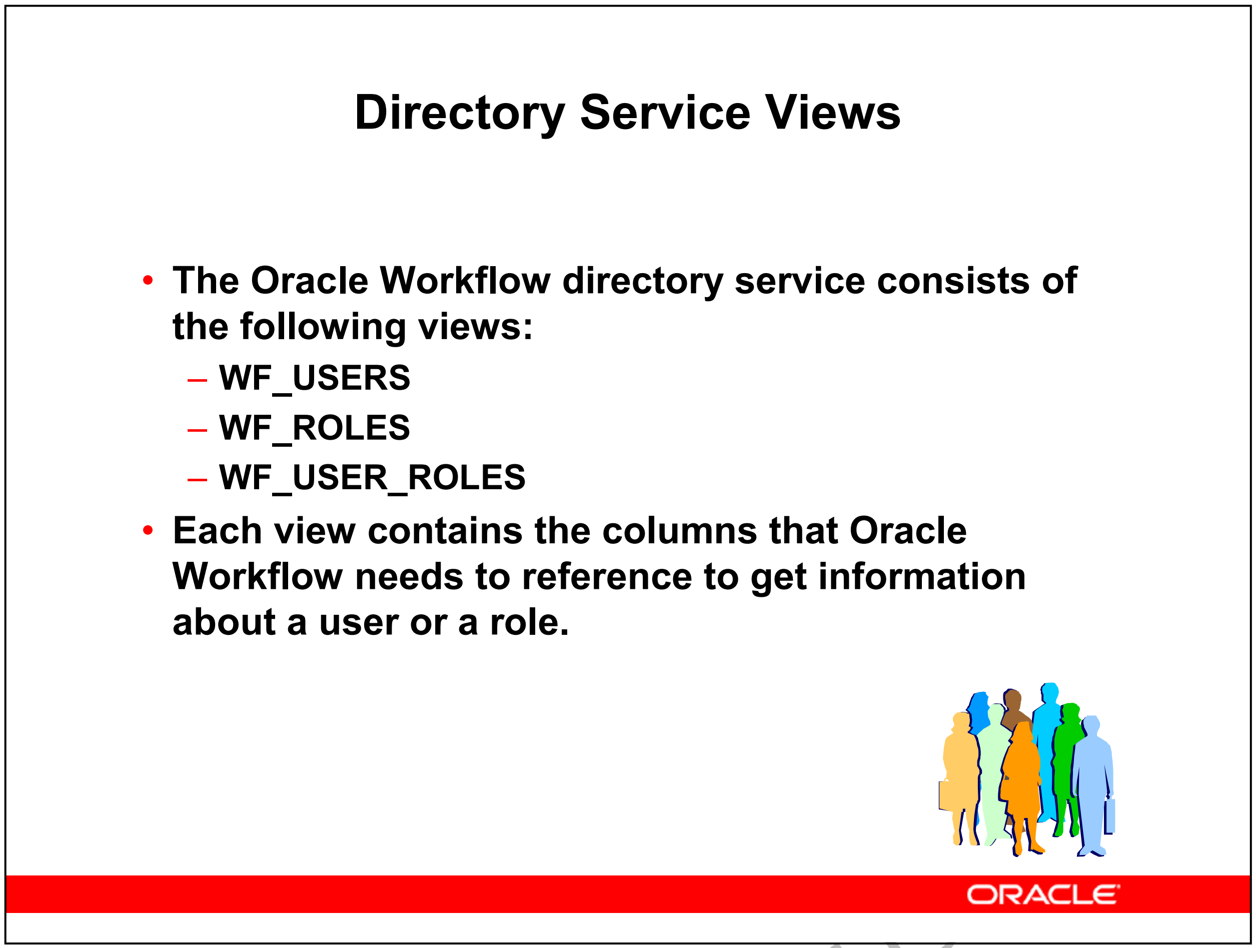
After completing this lesson, you should be able to do the following:

- Describe how Oracle Workflow accesses user and role information.
- Map Oracle Workflow's directory service to your directory repository.
- Load roles into Oracle Workflow Builder.



Oracle Workflow Directory Service

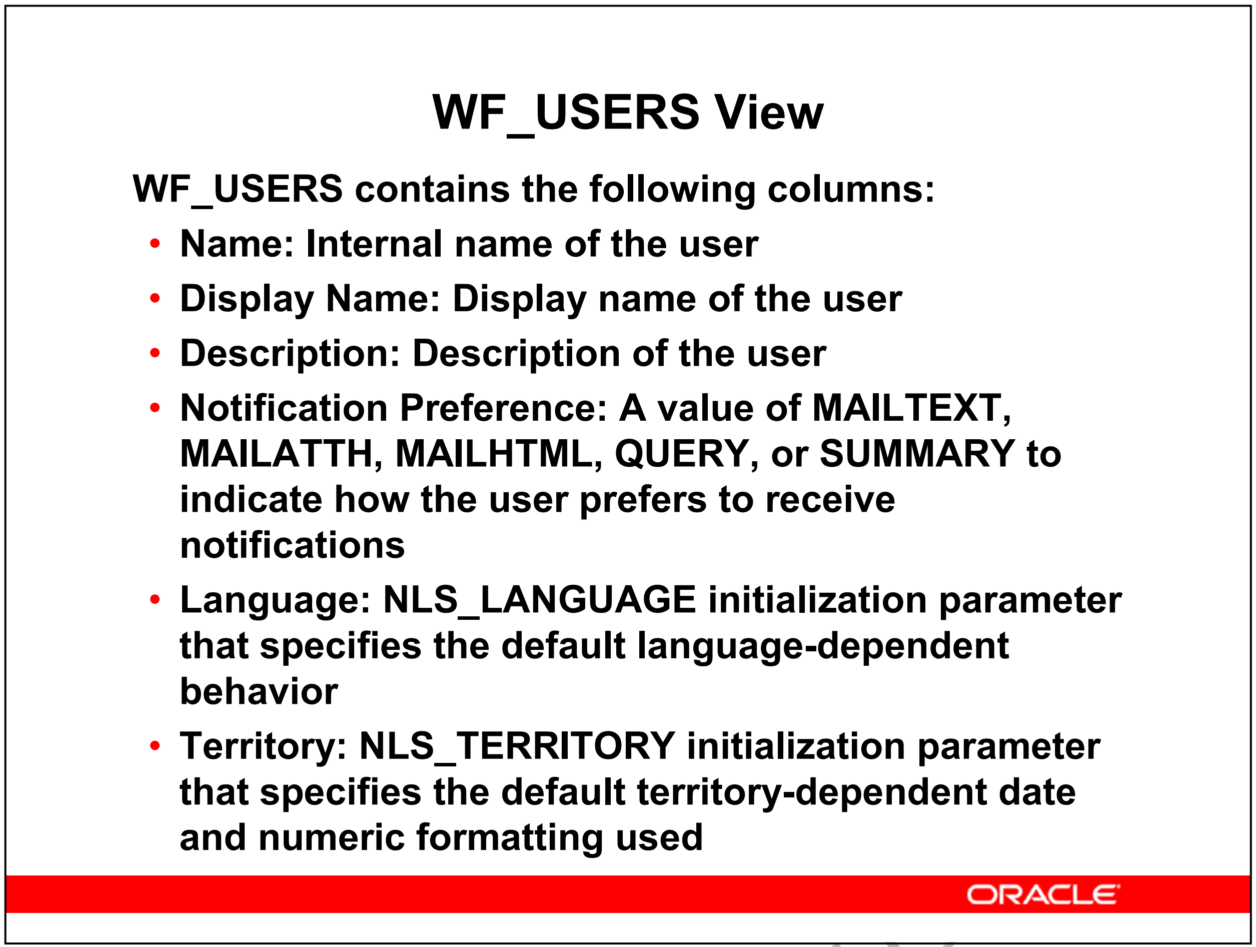
- A role is a grouping of one or more users who share a common responsibility.
- Oracle Workflow assigns ownership of work items and sends notifications to roles.
- Oracle Workflow does not maintain its own repository of users and roles. The Oracle Workflow directory service consists of three views that map to a designated directory repository.
- A directory repository consists of a set of database tables that contain user and role information.



Directory Service Views

- The Oracle Workflow directory service consists of the following views:
  - WF_USERS
  - WF_ROLES
  - WF_USER_ROLES
- Each view contains the columns that Oracle Workflow needs to reference to get information about a user or a role.

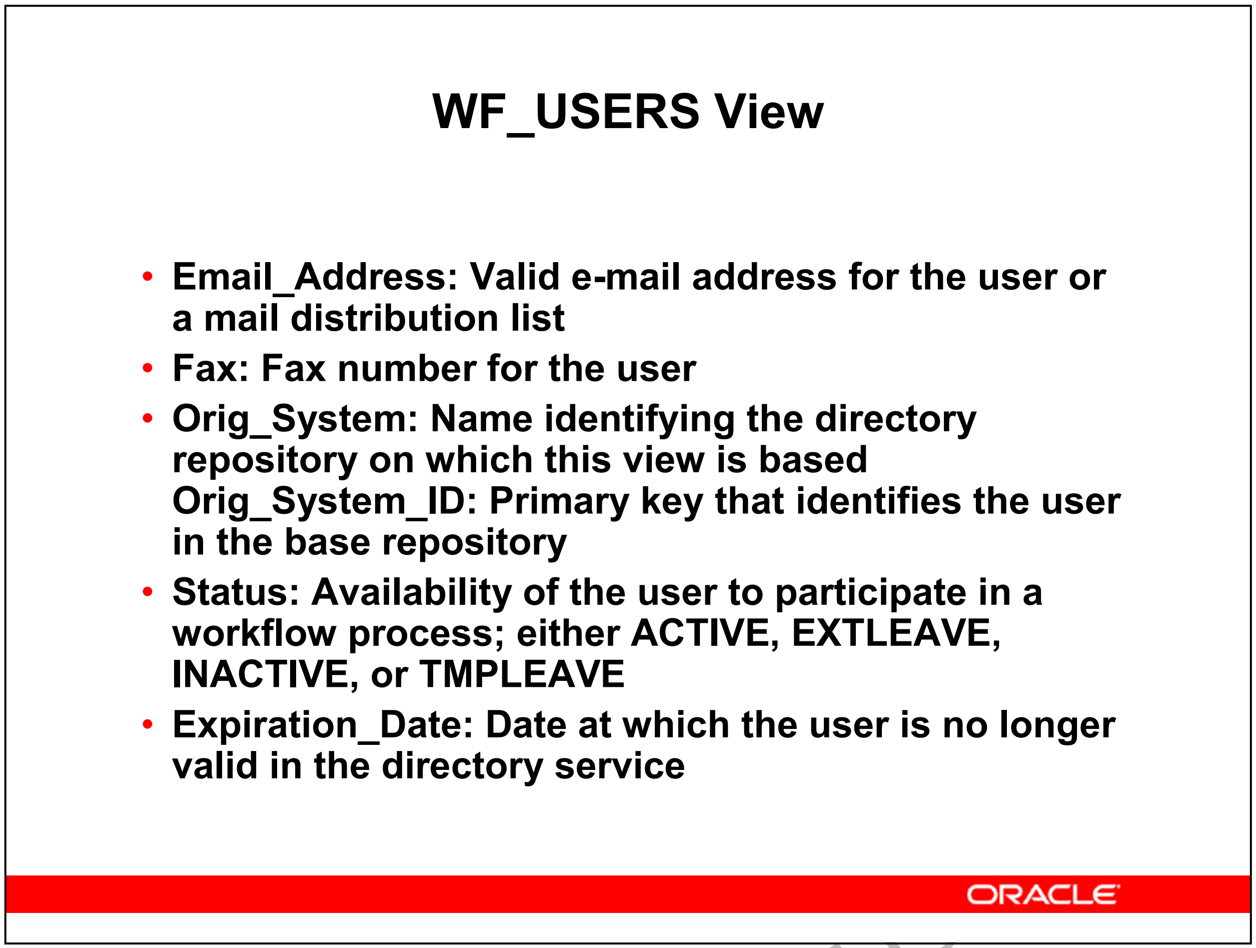
WF_USERS View



WF_USERS View

WF_USERS contains the following columns:

- Name: Internal name of the user
- Display Name: Display name of the user
- Description: Description of the user
- Notification Preference: A value of MAILTEXT, MAILATTH, MAILHTML, QUERY, or SUMMARY to indicate how the user prefers to receive notifications
- Language: NLS_LANGUAGE initialization parameter that specifies the default language-dependent behavior
- Territory: NLS_TERRITORY initialization parameter that specifies the default territory-dependent date and numeric formatting used

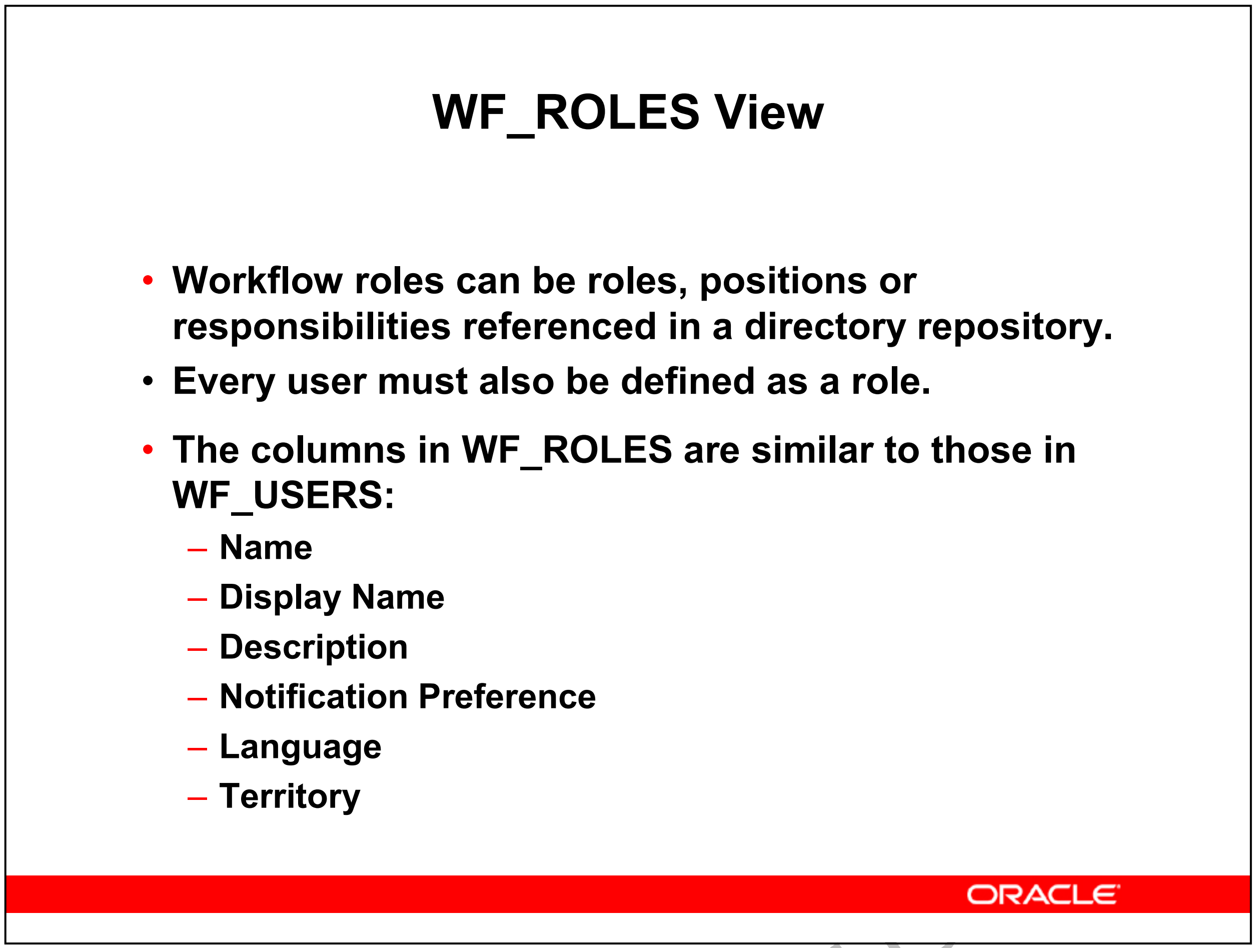


WF_USERS View

- Email_Address: Valid e-mail address for the user or a mail distribution list
- Fax: Fax number for the user
- Orig_System: Name identifying the directory repository on which this view is based
  Orig_System_ID: Primary key that identifies the user in the base repository
- Status: Availability of the user to participate in a workflow process; either ACTIVE, EXTLEAVE, INACTIVE, or TMPLEAVE
- Expiration_Date: Date at which the user is no longer valid in the directory service

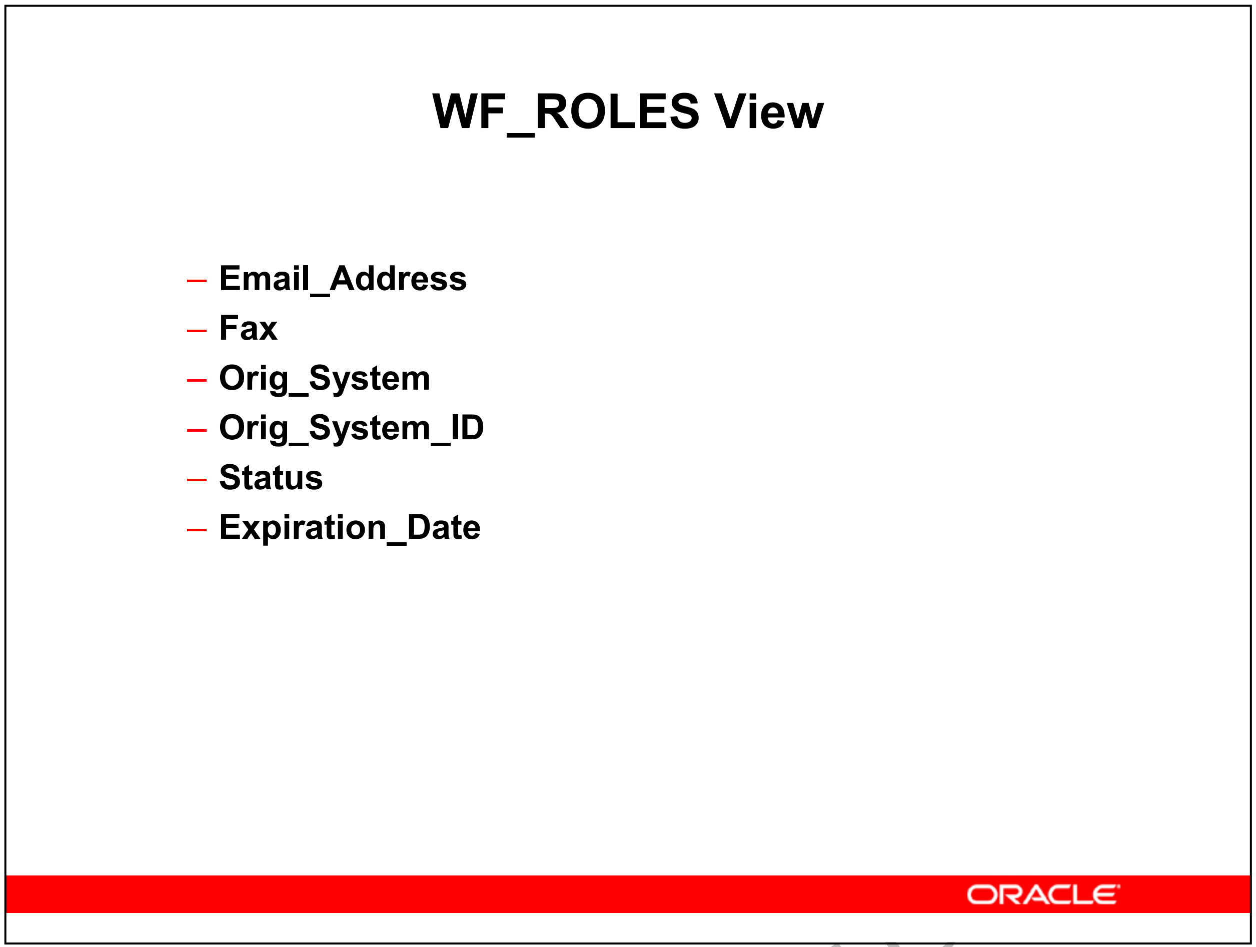
WF_USERS View

The Oracle Workflow Directory Service APIs use the ORIG_SYSTEM_ID stored in the views to allow indexed access back to the directory service base tables.



WF_ROLES View

- Workflow roles can be roles, positions or responsibilities referenced in a directory repository.
- Every user must also be defined as a role.
- The columns in WF_ROLES are similar to those in WF_USERS:
  - Name
  - Display Name
  - Description
  - Notification Preference
  - Language
  - Territory

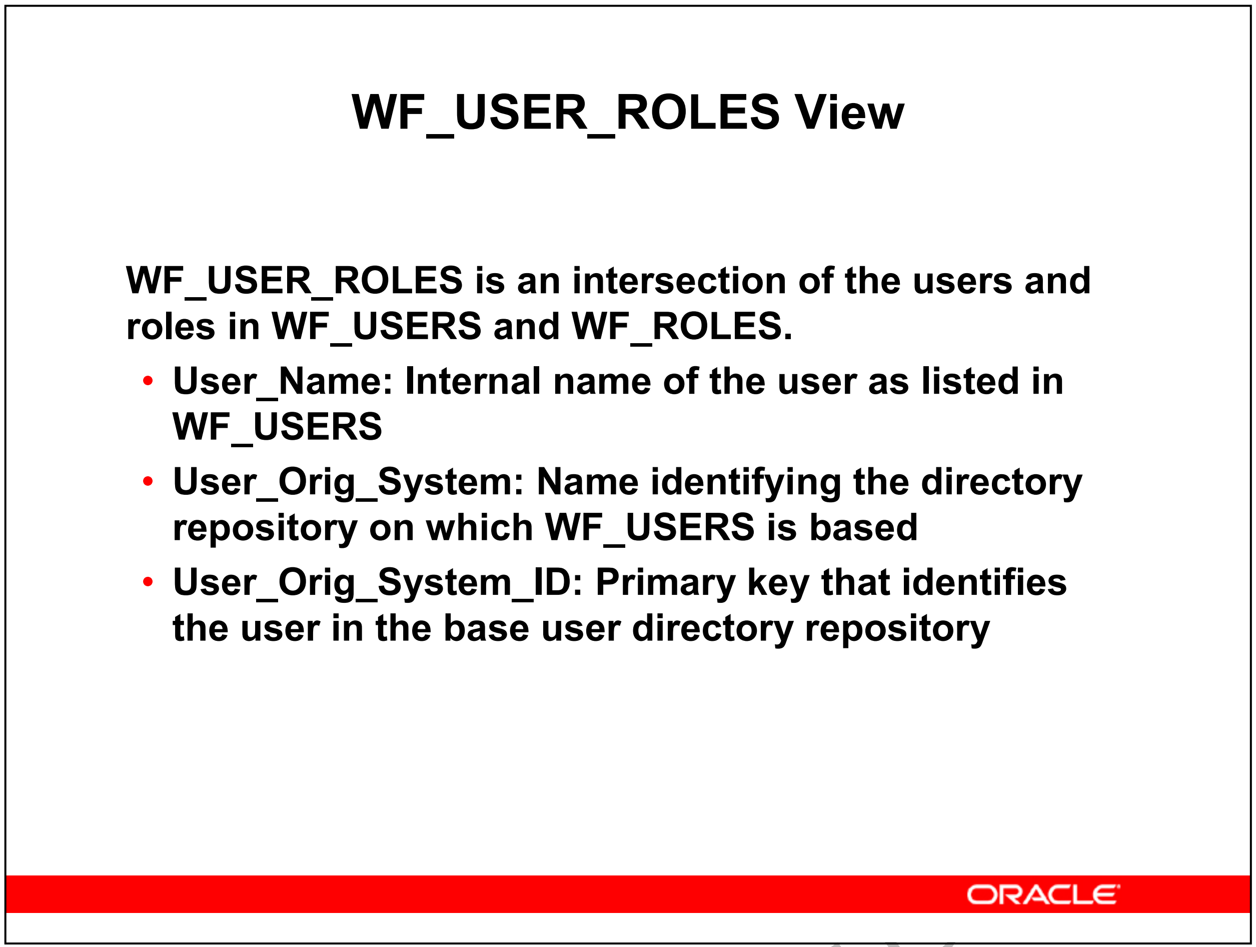


WF_ROLES View

- Email_Address
- Fax
- Orig_System
- Orig_System_ID
- Status
- Expiration_Date

WF_ROLES View

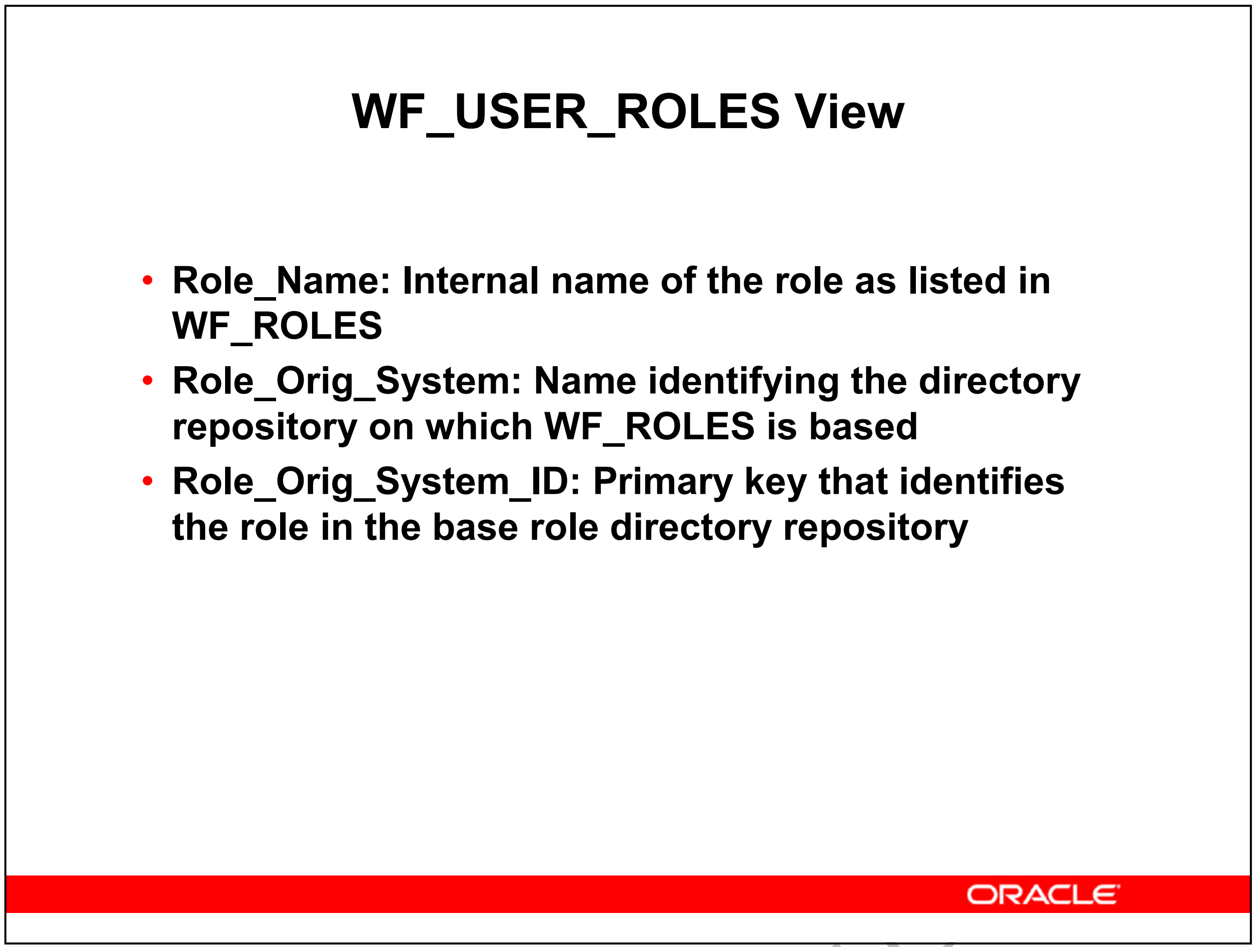
- If the e-mail address is null for a given role, the Notification Mailer sends an individual e-mail to each user within the role.
- The Oracle Workflow Directory Service APIs use the ORIG_SYSTEM_ID stored in the views to allow indexed access back to the directory service base tables.



WF_USER_ROLES View

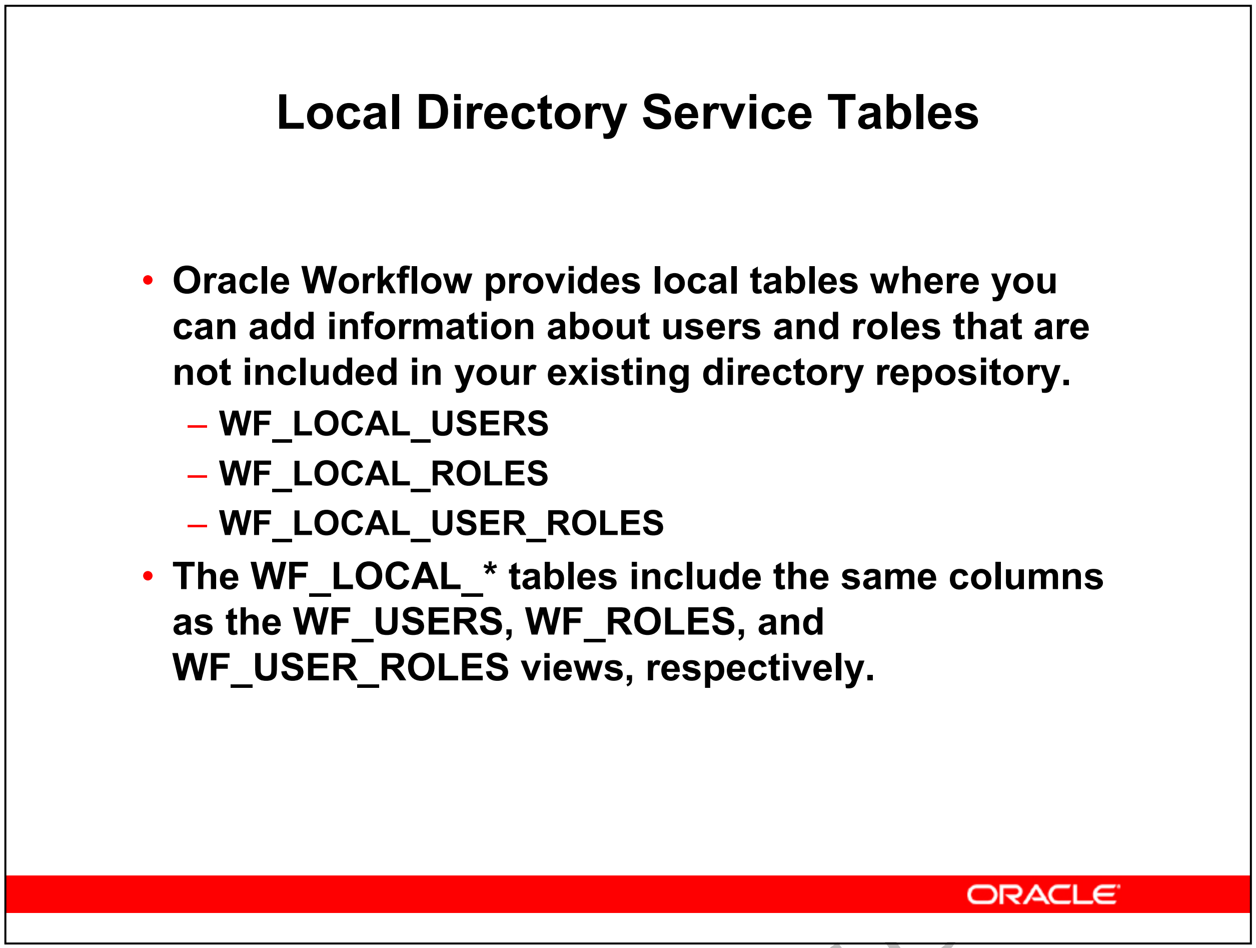
WF_USER_ROLES is an intersection of the users and roles in WF_USERS and WF_ROLES.

- **User_Name: Internal name of the user as listed in WF_USERS**
- **User_Orig_System: Name identifying the directory repository on which WF_USERS is based**
- **User_Orig_System_ID: Primary key that identifies the user in the base user directory repository**



WF_USER_ROLES View

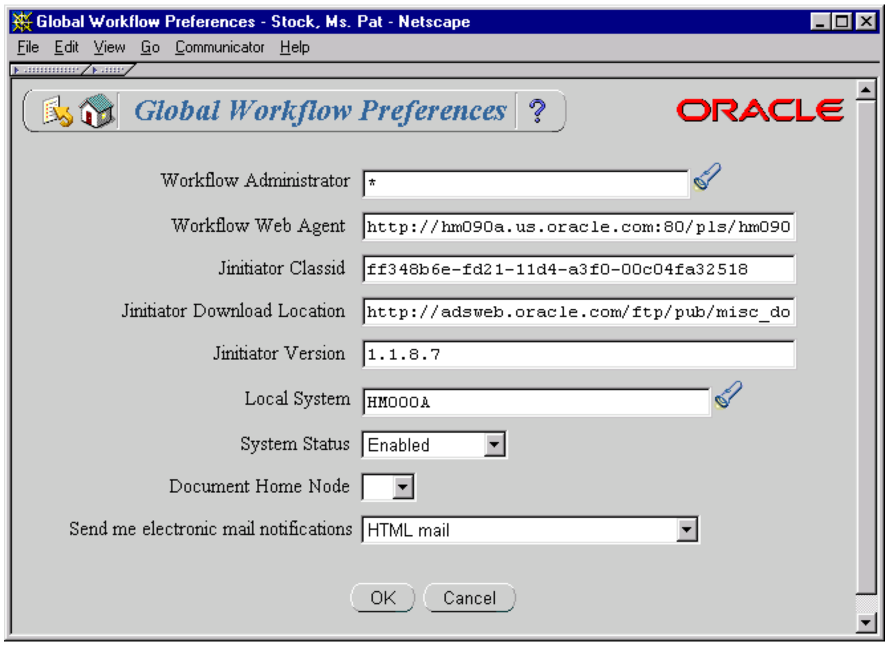
- Role_Name: Internal name of the role as listed in WF_ROLES
- Role_Orig_System: Name identifying the directory repository on which WF_ROLES is based
- Role_Orig_System_ID: Primary key that identifies the role in the base role directory repository



Local Directory Service Tables

- Oracle Workflow provides local tables where you can add information about users and roles that are not included in your existing directory repository.
  - WF_LOCAL_USERS
  - WF_LOCAL_ROLES
  - WF_LOCAL_USER_ROLES
- The WF_LOCAL_* tables include the same columns as the WF_USERS, WF_ROLES, and WF_USER_ROLES views, respectively.

Setting Global Workflow Preferences


Global Workflow Preferences - Stock, Ms. Pat - Netscape
File Edit View Go Communicator Help
Global Workflow Preferences
ORACLE
OK
Cancel

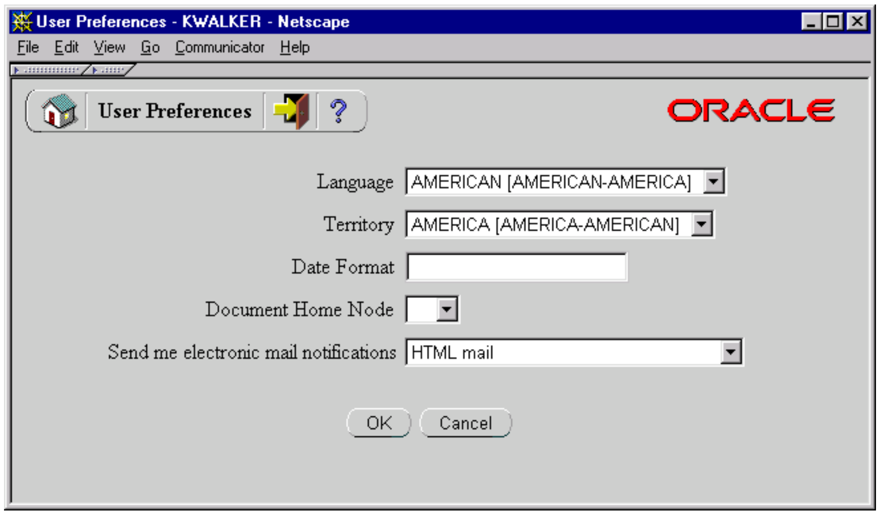
Workflow Administrator	*
Workflow Web Agent	http://hm090a.us.oracle.com:80/pls/hm090
Jinitiator Classid	ff348b6e-fd21-11d4-a3f0-00c04fa32518
Jinitiator Download Location	http://adsweb.oracle.com/ftp/pub/misc_do
Jinitiator Version	1.1.8.7
Local System	HM000A
System Status	Enabled
Document Home Node	
Send me electronic mail notifications	HTML mail

Setting Global Workflow Preferences

Control your users' interaction with Oracle Workflow by globally setting default preference values for the entire enterprise. Only the workflow administrator has access to the Global Workflow Preferences web page.

An individual user can override a default preference at any time by changing the value of the preference in the User Preferences web page.

Setting User Preferences

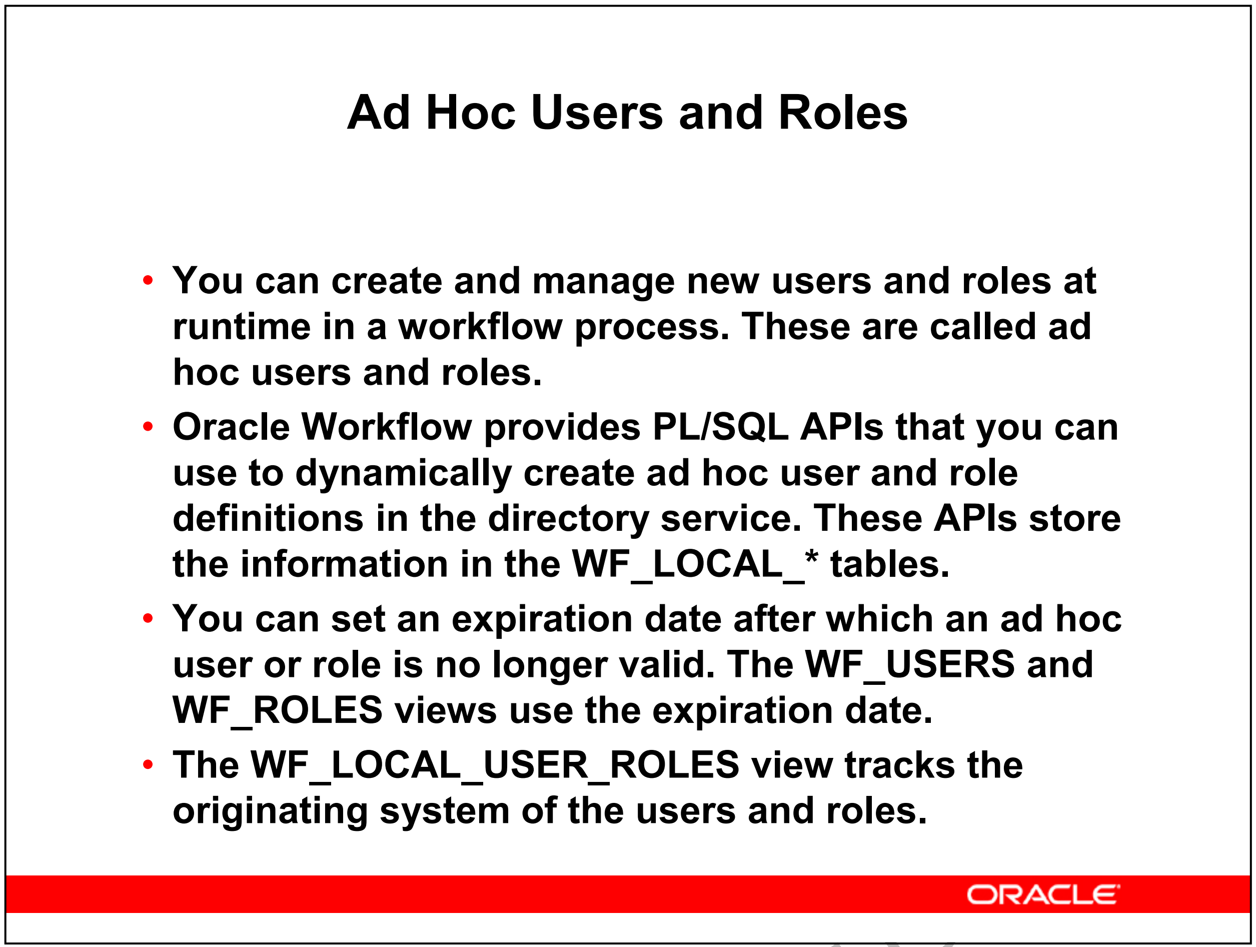

User Preferences - KWALKER - Netscape
File Edit View Go Communicator Help
User Preferences
ORACLE
OK
Cancel

Language	AMERICAN [AMERICAN-AMERICA]
Territory	AMERICA [AMERICA-AMERICAN]
Date Format	
Document Home Node	
Send me electronic mail notifications	HTML mail

Setting User Preferences

Users can specify preferences in the User Preferences web page to override the default global preferences set by the workflow administrator in the Global Workflow Preferences web page.

Note: The Language, Territory, and Notification preference settings in the Global Workflow Preferences and User Preferences web pages are valid only if your directory service views map these columns to the Oracle Workflow preferences table.



Ad Hoc Users and Roles

- You can create and manage new users and roles at runtime in a workflow process. These are called ad hoc users and roles.
- Oracle Workflow provides PL/SQL APIs that you can use to dynamically create ad hoc user and role definitions in the directory service. These APIs store the information in the WF_LOCAL_* tables.
- You can set an expiration date after which an ad hoc user or role is no longer valid. The WF_USERS and WF_ROLES views use the expiration date.
- The WF_LOCAL_USER_ROLES view tracks the originating system of the users and roles.

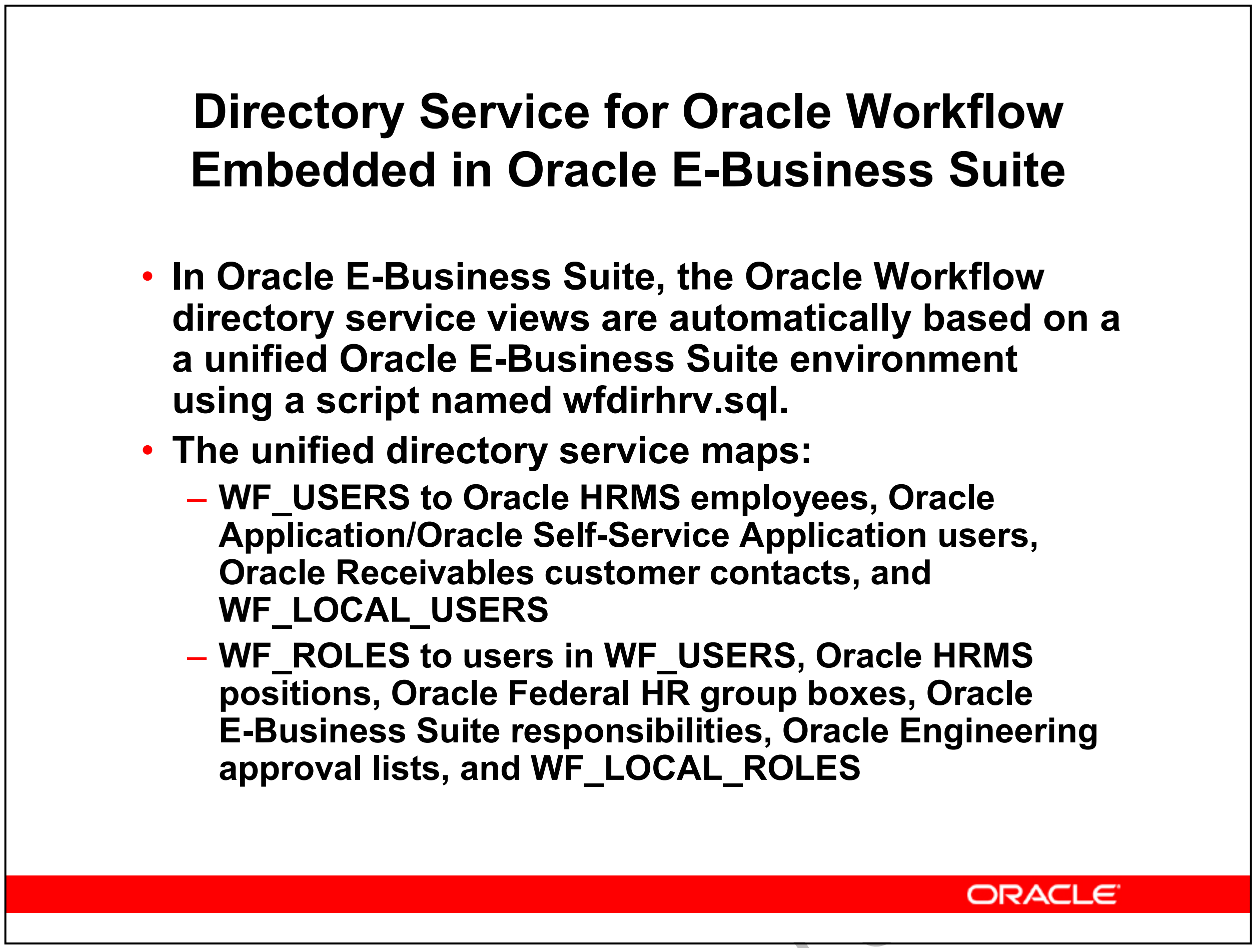
ORACLE

Ad Hoc Users/Roles

The WF_DIRECTORY package contains the APIs supporting ad hoc users and roles, including:

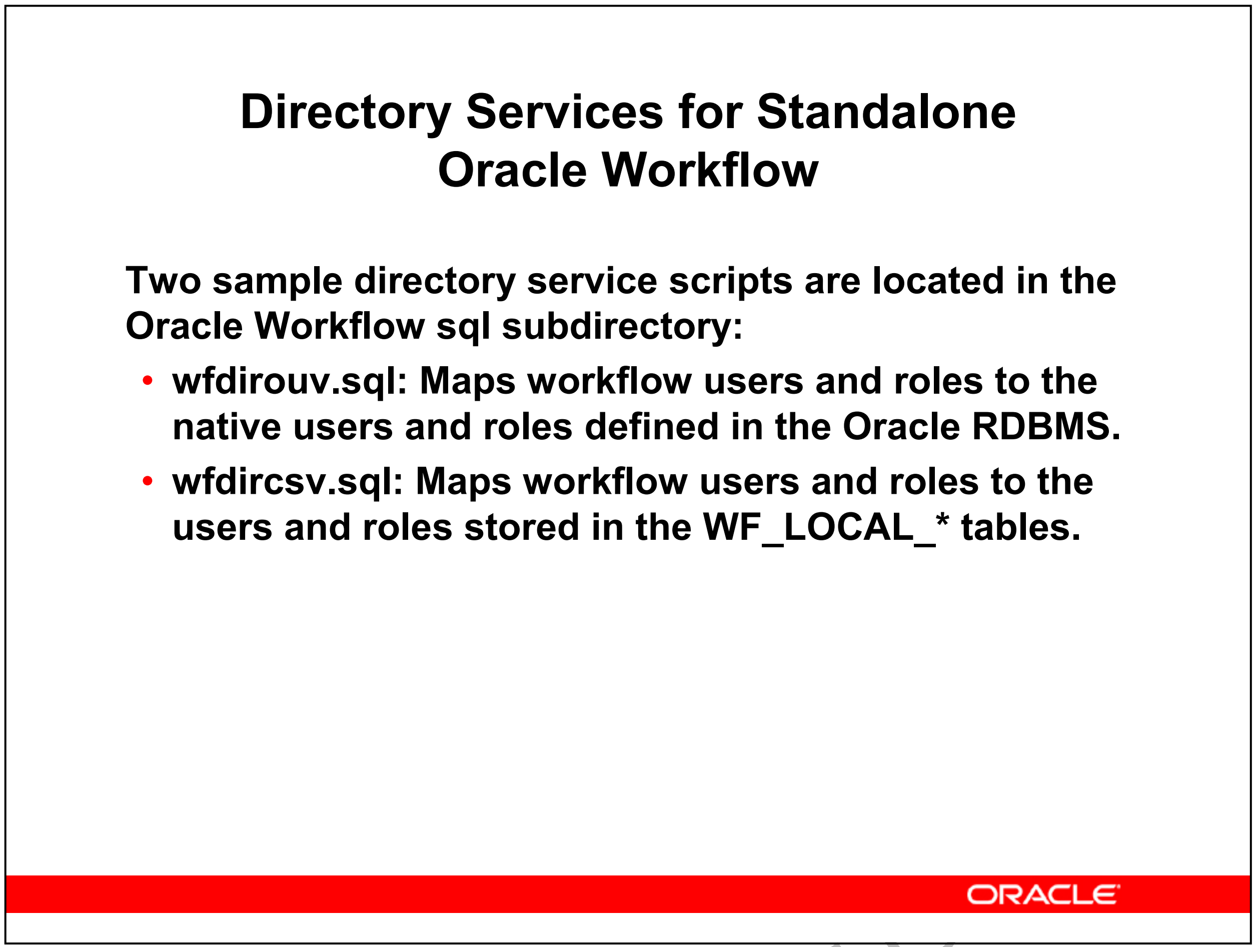
- CreateAdHocUser()
- CreateAdHocRole()
- AddUsersToAddHocRole()
- GetRoleDisplayName()
- SetAdHocUserStatus()
- SetAdHocRoleStatus()
- SetAdHocUserExpiration()
- SetAdHocRoleExpiration()
- SetAdHocUserAttr()
- SetAdHocRoleAttr()
- RemoveUsersFromAdHocRole()

Directory Service for Oracle Workflow Embedded in Oracle E-Business Suite



Directory Service for Oracle Workflow Embedded in Oracle E-Business Suite

- **In Oracle E-Business Suite, the Oracle Workflow directory service views are automatically based on a a unified Oracle E-Business Suite environment using a script named wfdirhrv.sql.**
- **The unified directory service maps:**
  - **WF_USERS to Oracle HRMS employees, Oracle Application/Oracle Self-Service Application users, Oracle Receivables customer contacts, and WF_LOCAL_USERS**
  - **WF_ROLES to users in WF_USERS, Oracle HRMS positions, Oracle Federal HR group boxes, Oracle E-Business Suite responsibilities, Oracle Engineering approval lists, and WF_LOCAL_ROLES**



Directory Services for Standalone Oracle Workflow

Two sample directory service scripts are located in the Oracle Workflow sql subdirectory:

- wfdirouv.sql: Maps workflow users and roles to the native users and roles defined in the Oracle RDBMS.
- wfdircsv.sql: Maps workflow users and roles to the users and roles stored in the WF_LOCAL_* tables.

Directory Services for Standalone Oracle Workflow

wfdirouv.sql

This script creates three views based on the following native Oracle tables:

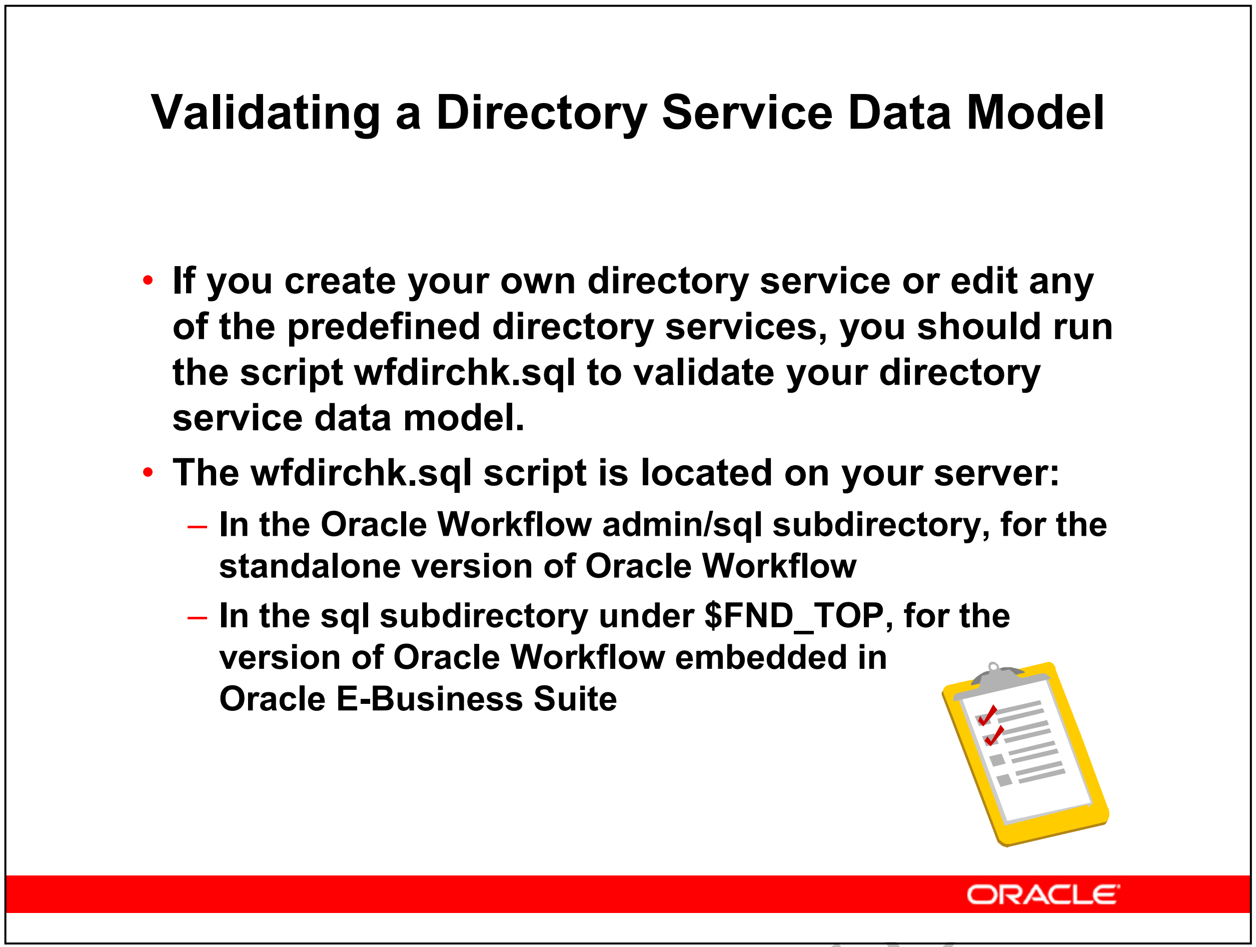
- DBA_USERS
- DBA_ROLES
- DBA_ROLE_PRIVS

wfdircsv.sql

This script creates three views based on the following tables:

- WF_LOCAL_USERS
- WF_LOCAL_ROLES
- WF_LOCAL_USER_ROLES

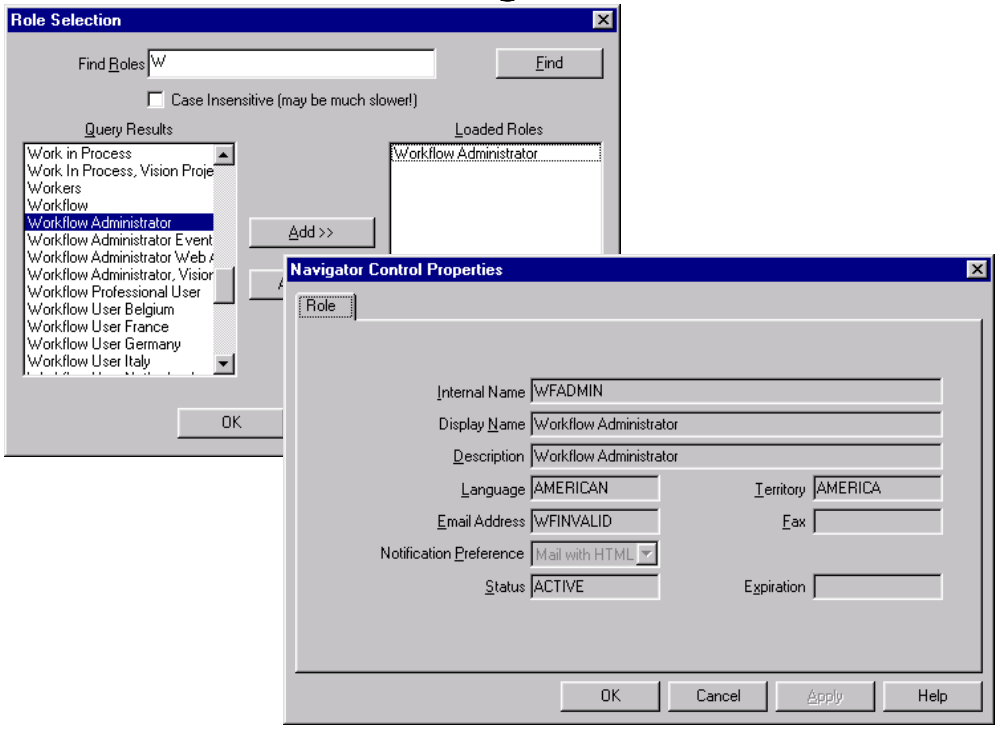
Note: You can use the WF_LOCAL_* tables to store users and roles not included in your central data repository. When you create your directory service views, you will need to select from the tables in your central repository as well as from the WF_LOCAL tables to get the complete list of users and roles.



Validating a Directory Service Data Model

- If you create your own directory service or edit any of the predefined directory services, you should run the script wfdirchk.sql to validate your directory service data model.
- The wfdirchk.sql script is located on your server:
  - In the Oracle Workflow admin/sql subdirectory, for the standalone version of Oracle Workflow
  - In the sql subdirectory under $FND_TOP, for the version of Oracle Workflow embedded in Oracle E-Business Suite

Loading Roles



Role Selection

Find Roles: W

☐ Case Insensitive (may be much slower!)

Query Results:
- Work in Process
- Work In Process, Vision Proje
- Workers
- Workflow
- Workflow Administrator
- Workflow Administrator Event
- Workflow Administrator Web A
- Workflow Administrator, Vision
- Workflow Professional User
- Workflow User Belgium
- Workflow User France
- Workflow User Germany
- Workflow User Italy

Loaded Roles:
- Workflow Administrator

Navigator Control Properties — Role

Internal Name	WFADMIN		
Display Name	Workflow Administrator		
Description	Workflow Administrator		
Language	AMERICAN	Territory	AMERICA
Email Address	WFINVALID	Fax	
Notification Preference	Mail with HTML		
Status	ACTIVE	Expiration	

Loading Roles

If you want to reference specific roles in a workflow process definition, such as by setting the recipient of a notification to a constant value, you must first load the roles stored in your Oracle Workflow directory service into Oracle Workflow Builder.

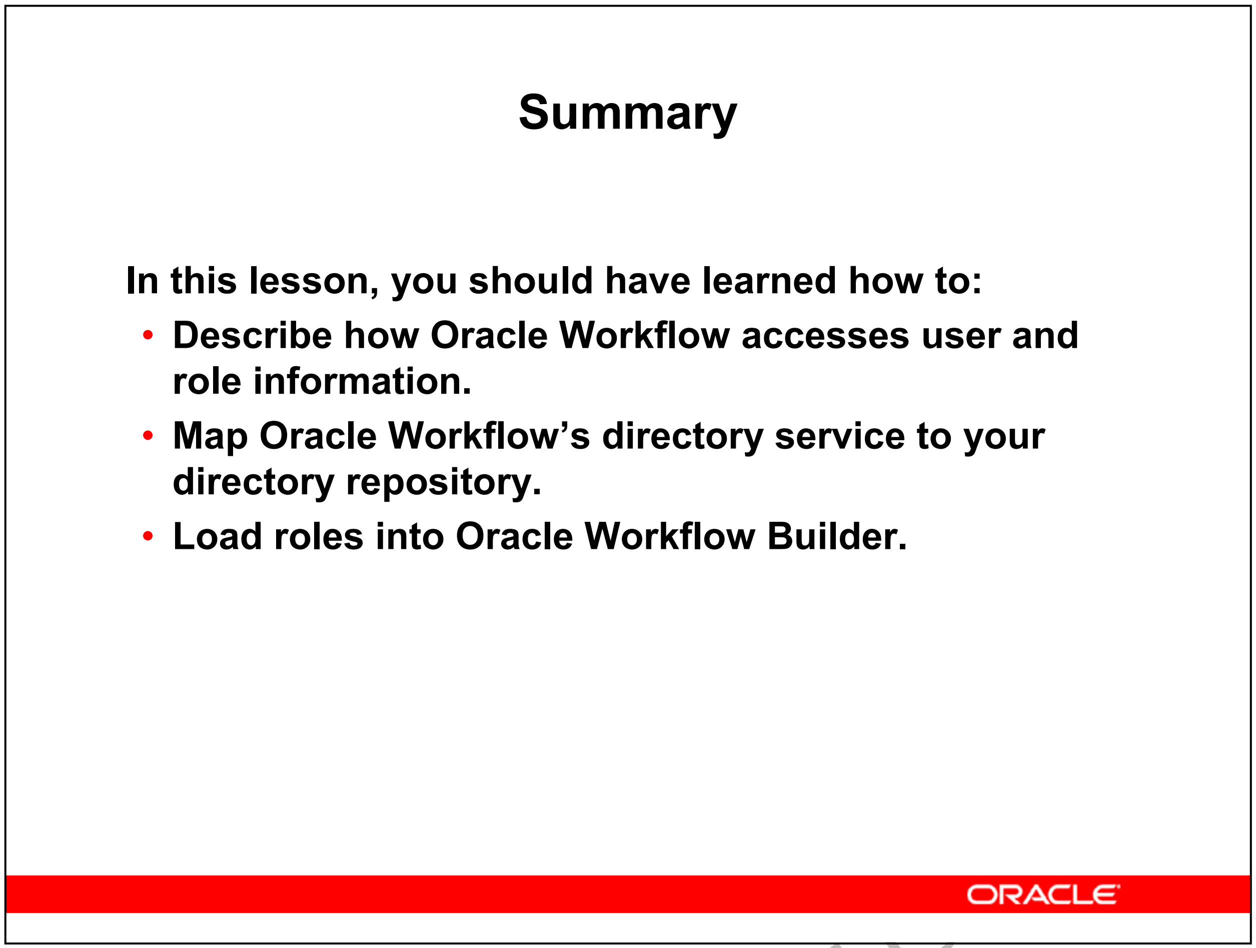
To load roles:

1. In Oracle Workflow Builder, choose Open from the File menu to connect to your database and open the item type you want.
2. From the File menu, choose Load Roles from Database.
3. Specify search criteria in the Find Roles field of the Role selection window and then choose Find.
4. Select the roles you want to load in the Query Results list, and choose Add to add them to the Loaded Roles list.
5. Choose OK to load the roles and make them available to the objects in your open item type.

After you load roles from the database, you can expand the Directory Service branch in the navigator tree to view information about the roles.

The property pages for workflow objects that reference role information, such as attributes of type role, contain role fields whose lists of values are populated by the roles you loaded from the database. You can select the role you want from the list.

Summary



Summary

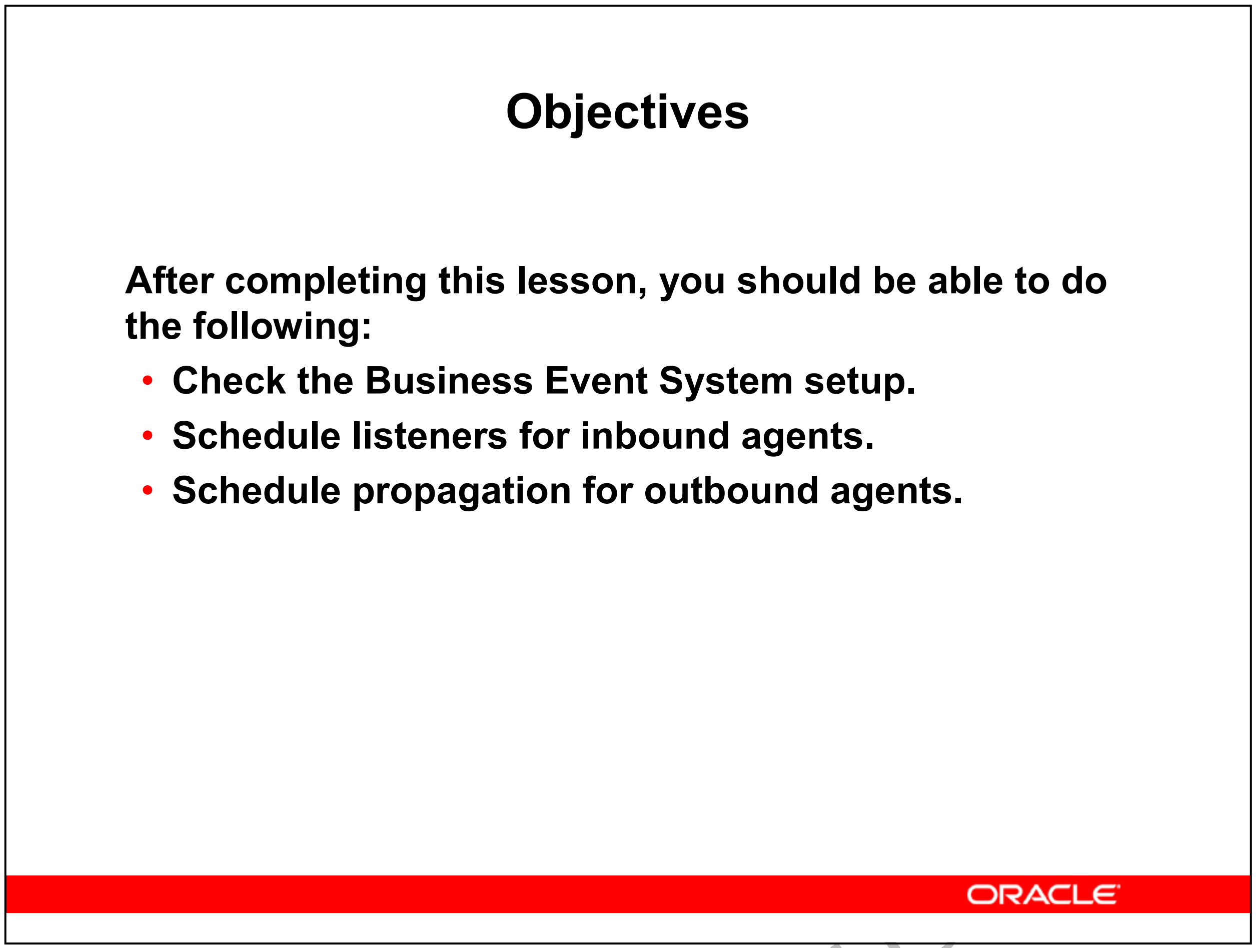
In this lesson, you should have learned how to:

- Describe how Oracle Workflow accesses user and role information.
- Map Oracle Workflow's directory service to your directory repository.
- Load roles into Oracle Workflow Builder.

Configuring Oracle Workflow for Event Communication

Chapter 26

Configuring Oracle Workflow for Event Communication

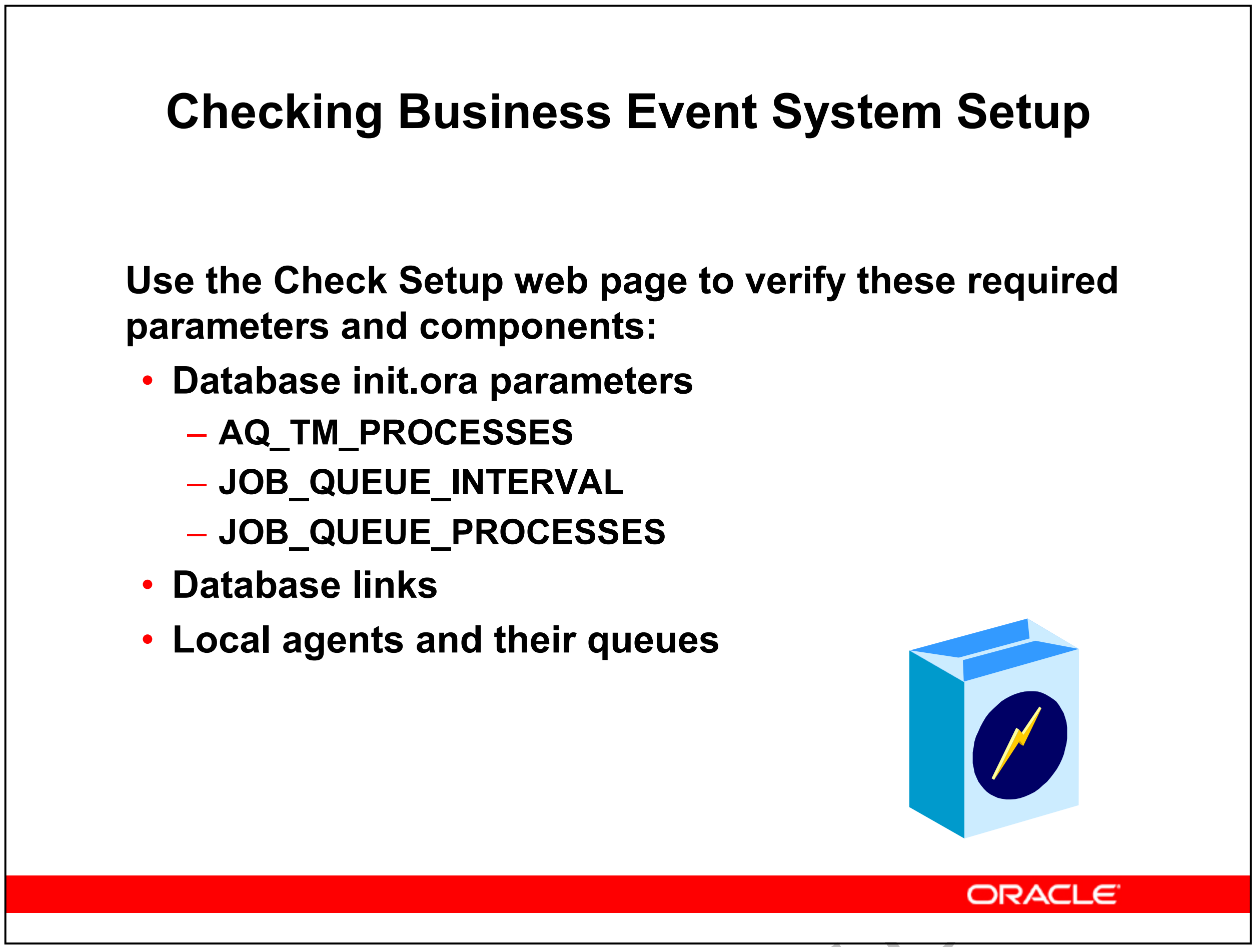


Objectives

After completing this lesson, you should be able to do the following:

- Check the Business Event System setup.
- Schedule listeners for inbound agents.
- Schedule propagation for outbound agents.

Checking Business Event System Setup



Checking Business Event System Setup

Use the Check Setup web page to verify these required parameters and components:

- **Database init.ora parameters**
  - **AQ_TM_PROCESSES**
  - **JOB_QUEUE_INTERVAL**
  - **JOB_QUEUE_PROCESSES**
- **Database links**
- **Local agents and their queues**

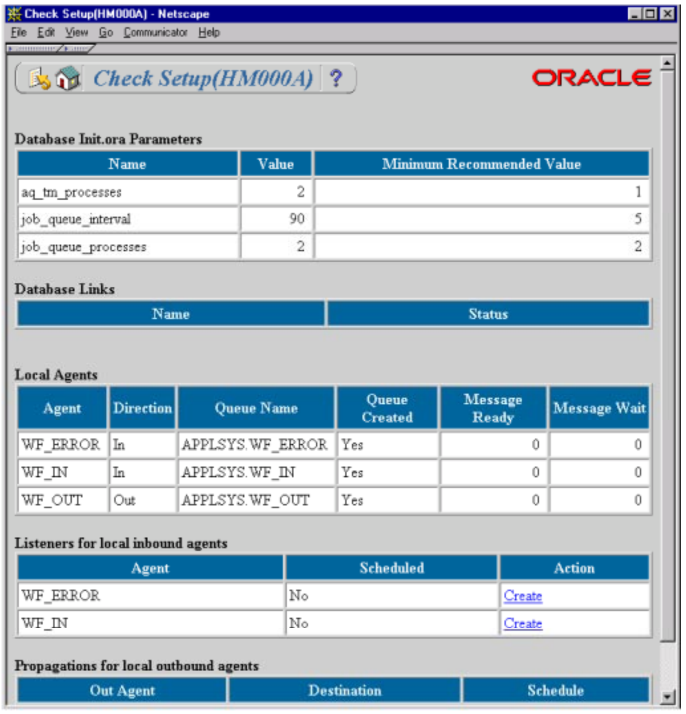
Checking Business Event System Setup

Database Init.ora Parameters

- AQ_TM_PROCESSES—This parameter enables the time manager process in Oracle8i Advanced Queuing (AQ). The time manager process is required by Oracle Workflow to monitor delay events in queues. The minimum recommended number of time manager processes for Oracle Workflow is one.
- JOB_QUEUE_INTERVAL—If you are using Oracle8*i*, specify the job queue interval to determine how frequently each SNP job queue process in your instance wakes up. SNP job queue processes are background processes in the Oracle database server that automatically refresh table snapshots, execute job requests, and propagate queued messages. Oracle Workflow requires the job queue interval to be less than or equal to the latency parameter defined for your AQ propagation schedules, to allow queues to be rechecked for messages with the specified latency. The recommended job queue interval for Oracle Workflow is five seconds. Note that because the JOB_QUEUE_INTERVAL parameter is desupported in Oracle9*i*, the Check Setup page does not display this parameter if you are using Oracle9*i*, and you do not need to set a value for it.

- JOB_QUEUE_PROCESSES—This parameter defines the number of SNP job queue processes for your instance. Oracle Workflow requires job queue processes to handle propagation of Business Event System event messages by AQ queues. You must start at least one job queue process to enable message propagation. The minimum recommended number of processes for Oracle Workflow is two; however, if no processes are being assigned for propagation you may need to increase this parameter to 5 or 10, or even higher.

Checking Business Event System Setup


Check Setup(HM000A) - Netscape
File Edit View Go Communicator Help
Check Setup(HM000A)
ORACLE

Database Init.ora Parameters

Name	Value	Minimum Recommended Value
aq_tm_processes	2	1
job_queue_interval	90	5
job_queue_processes	2	2

Database Links

Name	Status

Local Agents

Agent	Direction	Queue Name	Queue Created	Message Ready	Message Wait
WF_ERROR	In	APPLSYS.WF_ERROR	Yes	0	0
WF_IN	In	APPLSYS.WF_IN	Yes	0	0
WF_OUT	Out	APPLSYS.WF_OUT	Yes	0	0

Listeners for local inbound agents

Agent	Scheduled	Action
WF_ERROR	No	Create
WF_IN	No	Create

Propagations for local outbound agents

Out Agent	Destination	Schedule

ORACLE

To Check the Business Event System Setup:

1. Use a web browser to connect to the Check Setup web page.

   For standalone Oracle Workflow, connect to the following URL:

   <*webagent*>/wf_setup.check_all

   Replace <*webagent*> with the base URL of the web agent configured for Oracle Workflow in your Web server.

   You can also access the Check Setup web page from the Oracle Workflow home page, <*webagent*>/wfa_html.home.

   For Oracle Workflow embedded in Oracle E-Business Suite, choose the Check Event Manager Setup option from a Workflow administrator responsibility with Event Manager functionality.

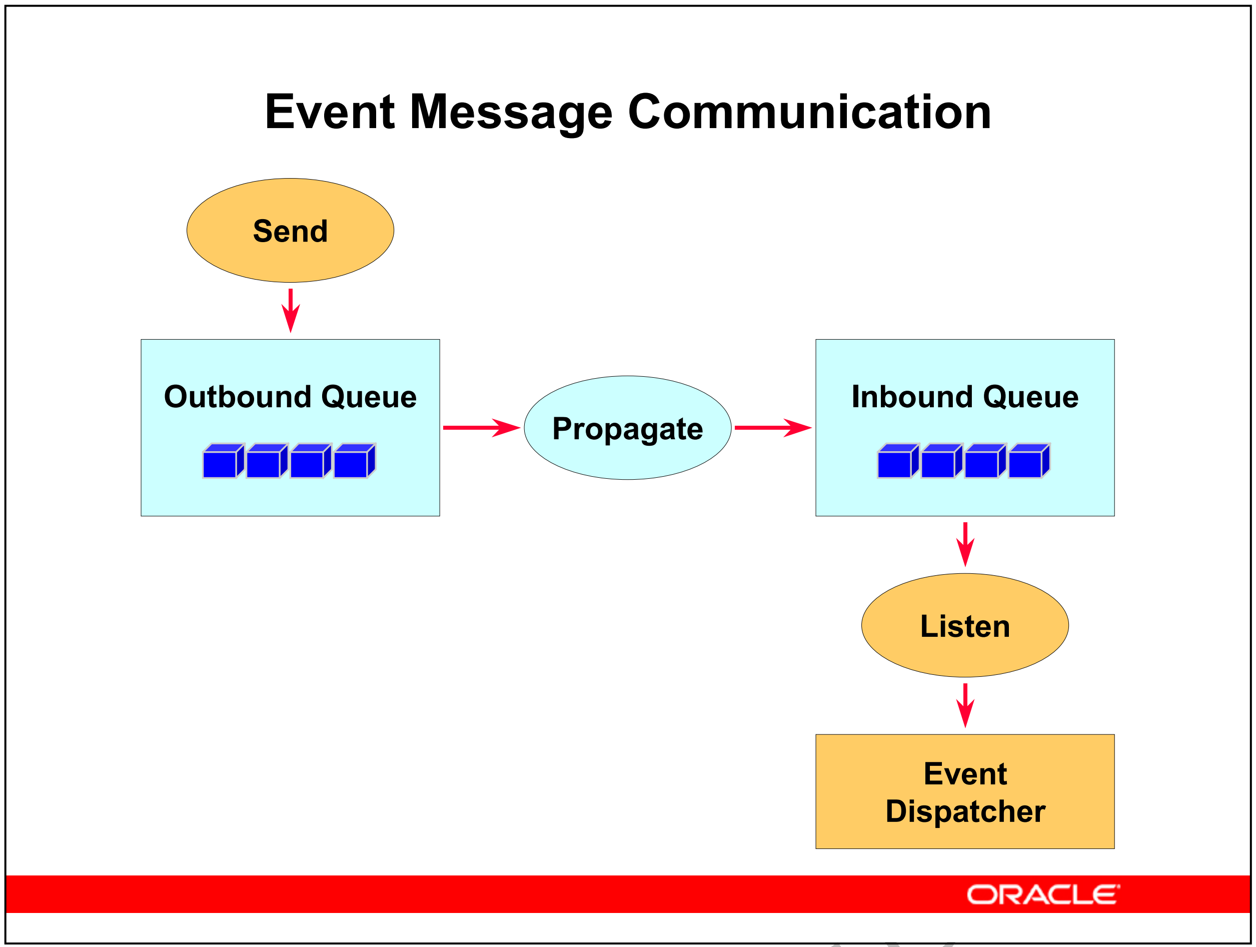
   Note: You must have workflow administrator privileges to access the Event Manager web pages.

2. In the Check Setup page, use the Database Init.ora Parameters region to verify your settings for the database initialization parameters related to AQ. The Check Setup page

displays the actual value defined for each parameter as well as the minimum recommended value for Oracle Workflow.

Note: Setting the init.ora parameters is completed as part of the Oracle Workflow installation steps. After a change to the init.ora parameters, you must restart your database to make the change effective.

3. Use the Database Links region to verify your database links. You should create any required database links that do not yet exist. The database link name as defined in the database must exactly match the database link name entered as part of the address for an agent.
4. Use the Local Agents region to verify the queues that are set up for the agents defined on your local system. You should create any required queues that do not yet exist.

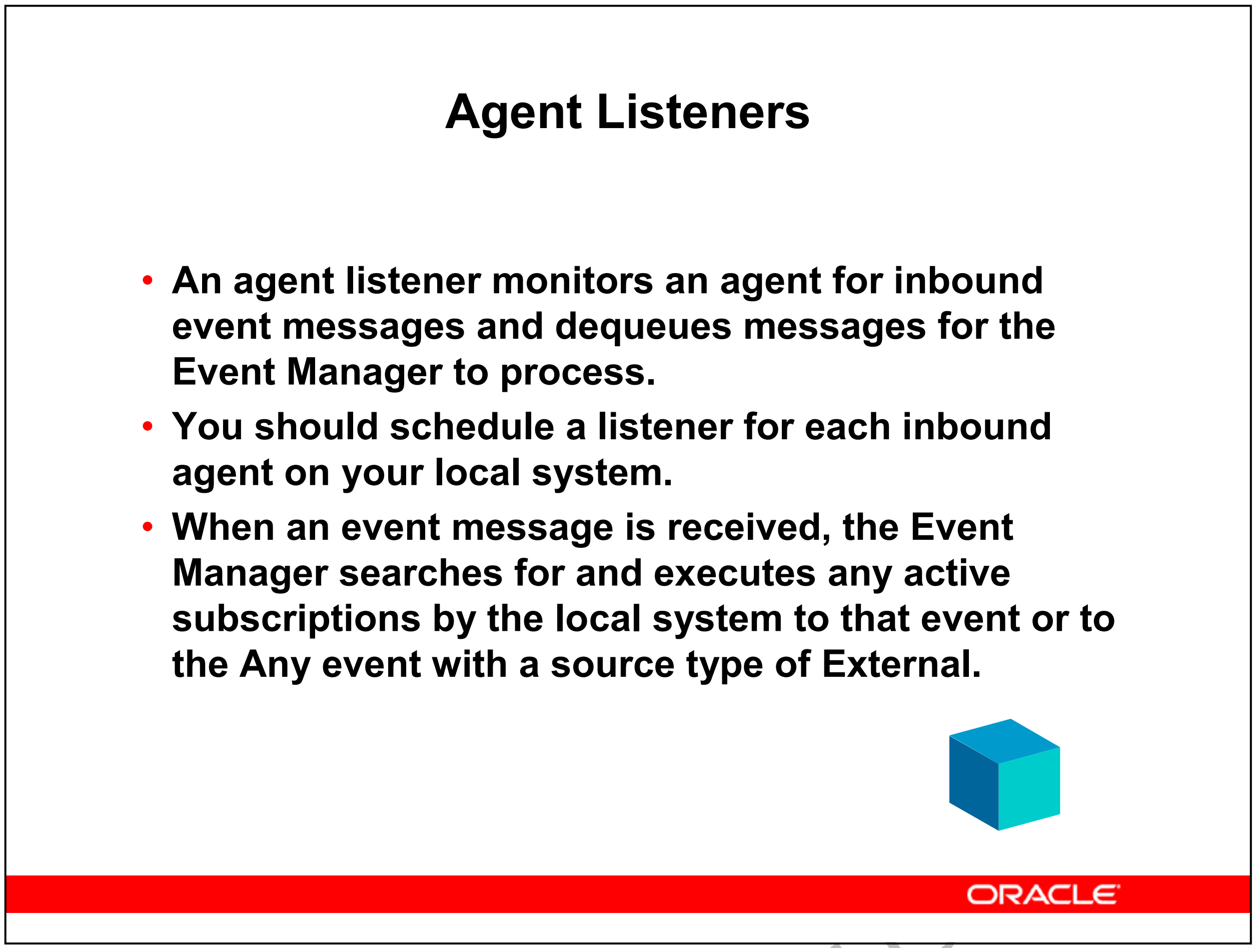


Event Message Communication

Event Message Communication

To send an event message, Oracle Workflow places the event message on a local outbound agent’s queue. You must schedule a propagation to deliver the message from there to the designated inbound agent’s queue.

To receive an event message in Oracle Workflow, you must schedule an agent listener to dequeue the message from the inbound agent’s queue for Oracle Workflow to process. A component of the Event Manager called the Event Dispatcher then searches for and executes subscriptions to the event.



Agent Listeners

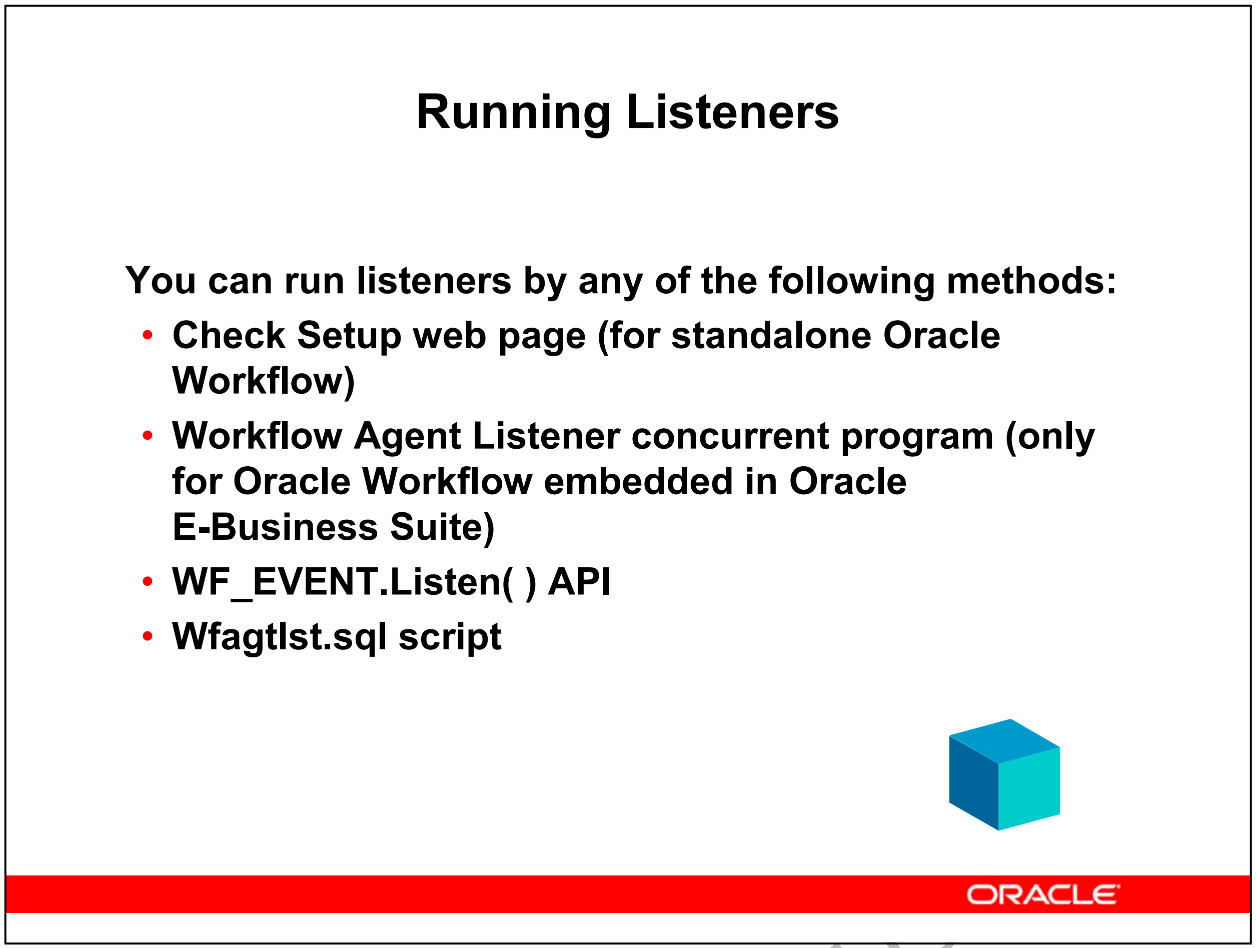
- An agent listener monitors an agent for inbound event messages and dequeues messages for the Event Manager to process.
- You should schedule a listener for each inbound agent on your local system.
- When an event message is received, the Event Manager searches for and executes any active subscriptions by the local system to that event or to the Any event with a source type of External.

ORACLE

Agent Listeners

You must schedule listeners for the standard WF_DEFERRED and WF_ERROR agents to enable deferred subscription processing and error handling for the Business Event System, respectively. Also, if you want to use the standard WF_IN agent for event message propagation, schedule a listener for that agent as well.

An agent listener exits after all event messages on the agent's queue have been dequeued.



Running Listeners

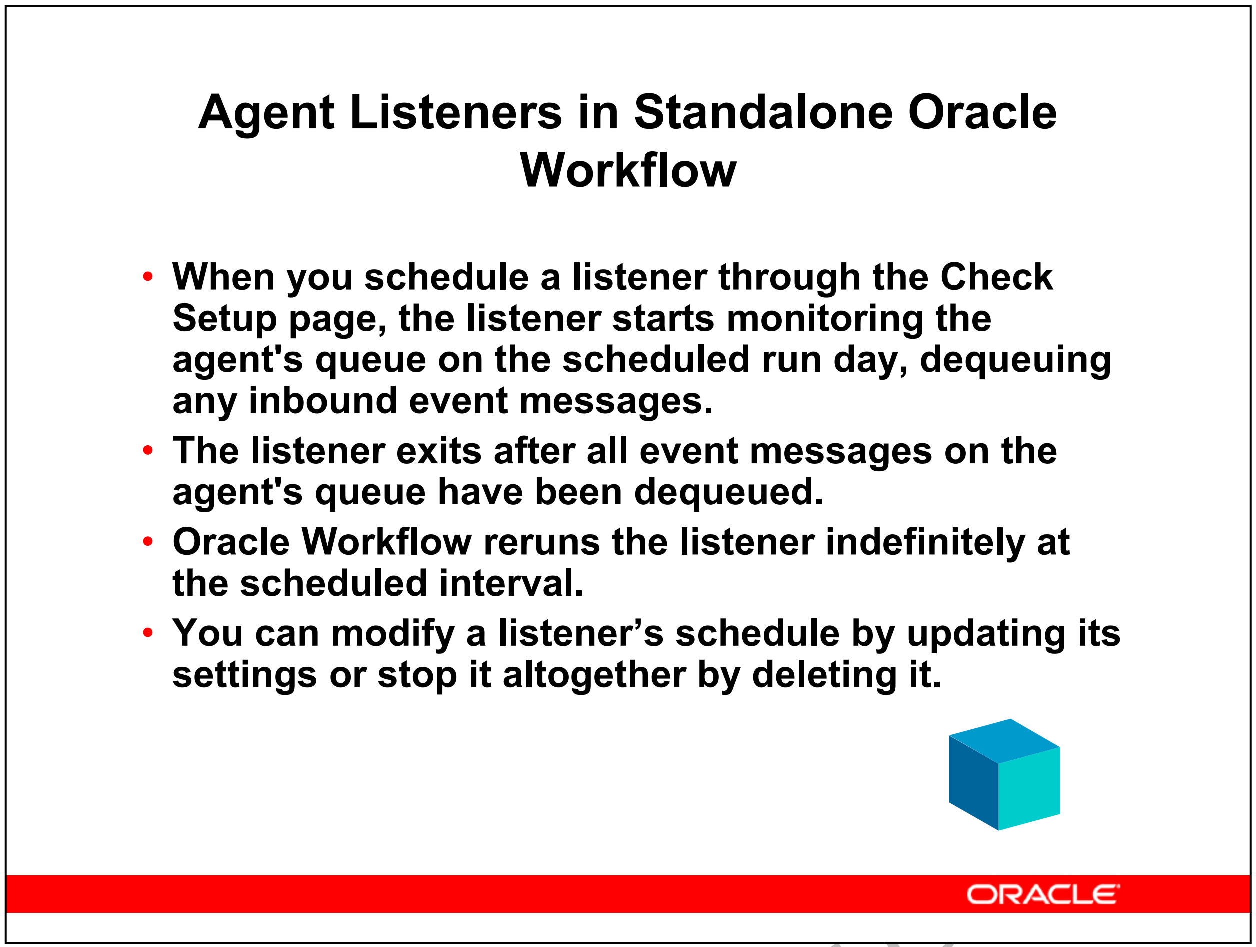
You can run listeners by any of the following methods:

- Check Setup web page (for standalone Oracle Workflow)
- Workflow Agent Listener concurrent program (only for Oracle Workflow embedded in Oracle E-Business Suite)
- WF_EVENT.Listen( ) API
- Wfagtlst.sql script

Running Listeners

You should only schedule listeners through the Check Setup web page if you are using the standalone version of Oracle Workflow. If you are using the version of Oracle Workflow embedded in Oracle E-Business Suite, you should run listeners by submitting the Workflow Agent Listener concurrent program.

The wfagtlst.sql script is intended primarily for debugging purposes.

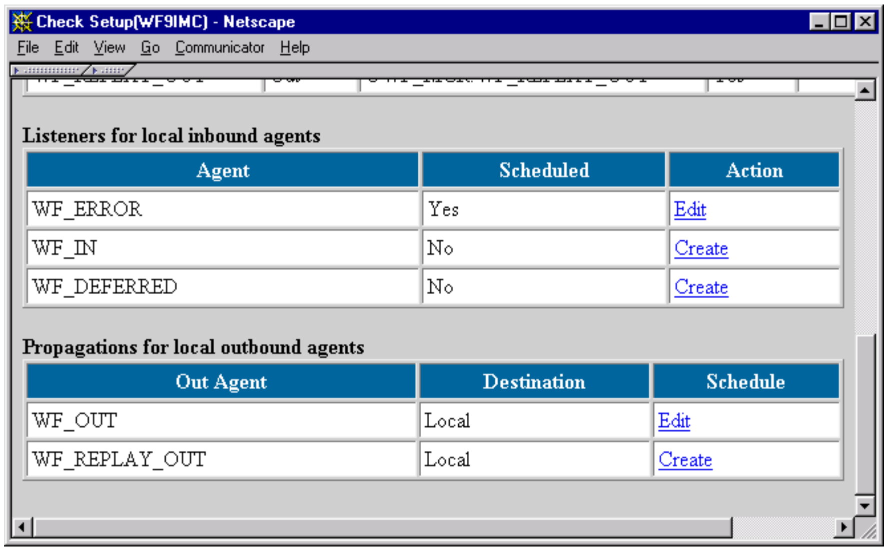


Agent Listeners in Standalone Oracle Workflow

- When you schedule a listener through the Check Setup page, the listener starts monitoring the agent's queue on the scheduled run day, dequeuing any inbound event messages.
- The listener exits after all event messages on the agent's queue have been dequeued.
- Oracle Workflow reruns the listener indefinitely at the scheduled interval.
- You can modify a listener’s schedule by updating its settings or stop it altogether by deleting it.

ORACLE

Scheduling Listeners in Standalone Oracle Workflow


Check Setup(WF9IMC) - Netscape
File Edit View Go Communicator Help

Listeners for local inbound agents

Agent	Scheduled	Action
WF_ERROR	Yes	Edit
WF_IN	No	Create
WF_DEFERRED	No	Create

Propagations for local outbound agents

Out Agent	Destination	Schedule
WF_OUT	Local	Edit
WF_REPLAY_OUT	Local	Create

To Schedule a Listener for an Inbound Agent:

1. Use a web browser to connect to the Check Setup web page.

   For standalone Oracle Workflow, connect to the following URL:

   <*webagent*>/wf_setup.check_all

   Replace <*webagent*> with the base URL of the web agent configured for Oracle Workflow in your Web server.

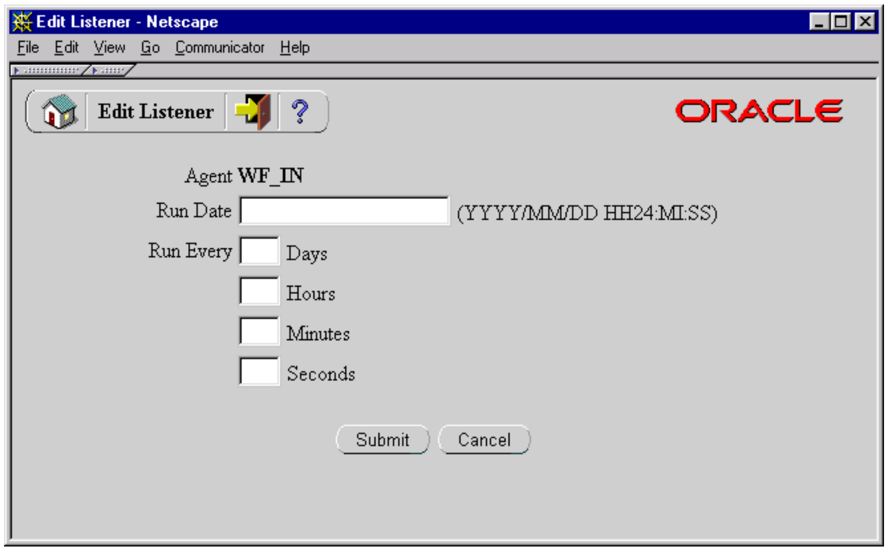
   You can also access the Check Setup web page from the Oracle Workflow home page, <*webagent*>/wfa_html.home.

   Note: You must have workflow administrator privileges to access the Event Manager web pages.
2. In the Check Setup page, locate the agent you want in the Listeners for Local Inbound Agents region.
3. If no listeners are scheduled for the agent yet, the Scheduled status for the agent is No. Choose the Create link in the Action column for that agent to create the first listener for the agent. The Edit Listener page appears, displaying the name of the selected agent.

If at least one listener is already scheduled for the agent, the Scheduled status for the agent is Yes. Choose the Edit link in the Action column for that agent to create an additional listener. The Listeners page appears, displaying a list of existing listeners for the agent. Choose the Add button. The Edit Listener page appears, displaying the name of the selected agent.

4. In Edit Listener page, enter the date on which you want to start running the listener in the Run Day field. To start the listener on the current system date, leave this field blank.
5. In the Run Every fields, enter an interval to specify how often you want the listener to be run. You can specify the interval in days, hours, minutes, and seconds.
6. Choose the Submit button to save the listener schedule.

Scheduling Listeners in Standalone Oracle Workflow



Agent WF_IN

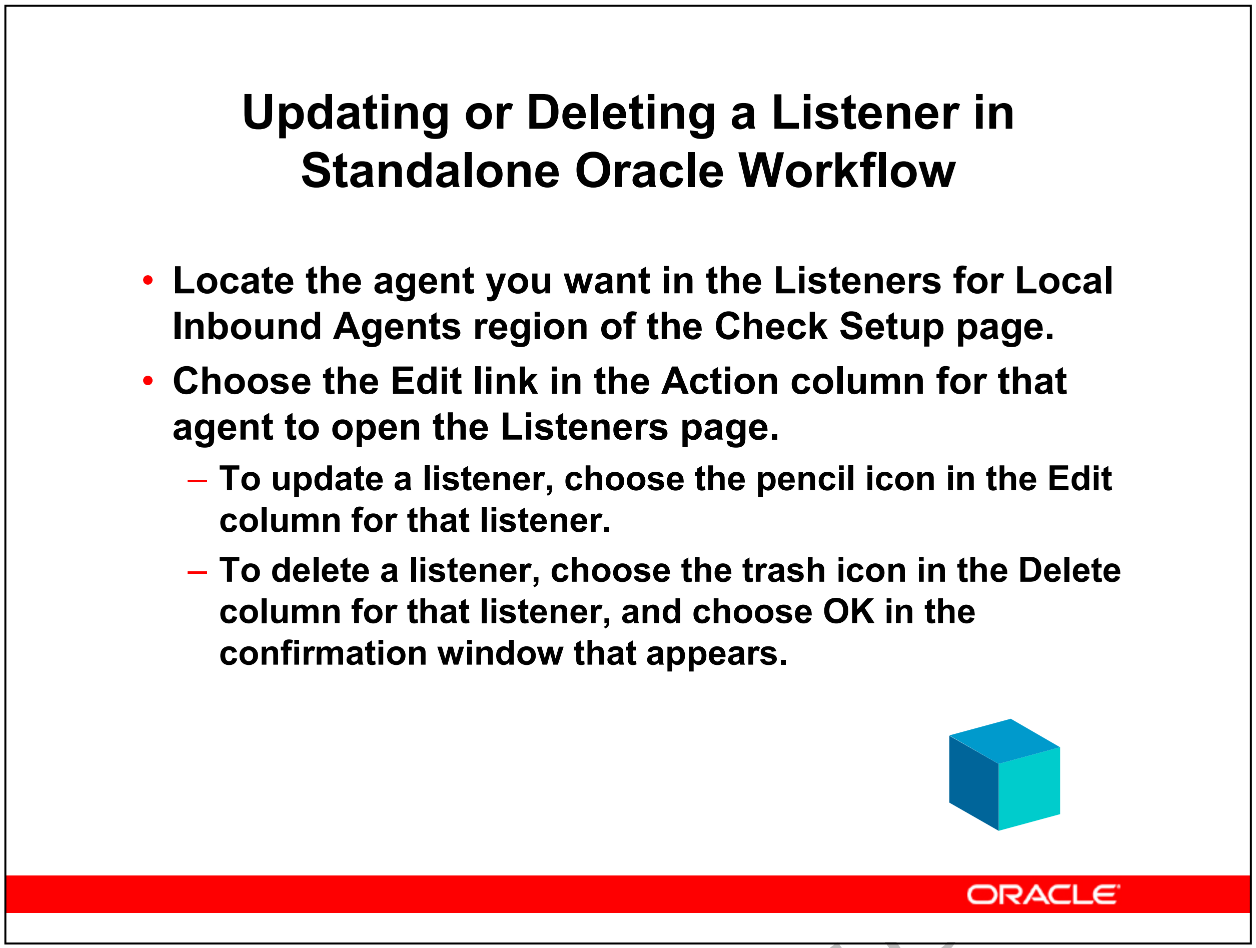
Run Date (YYYY/MM/DD HH24:MI:SS)

Run Every Days

Hours

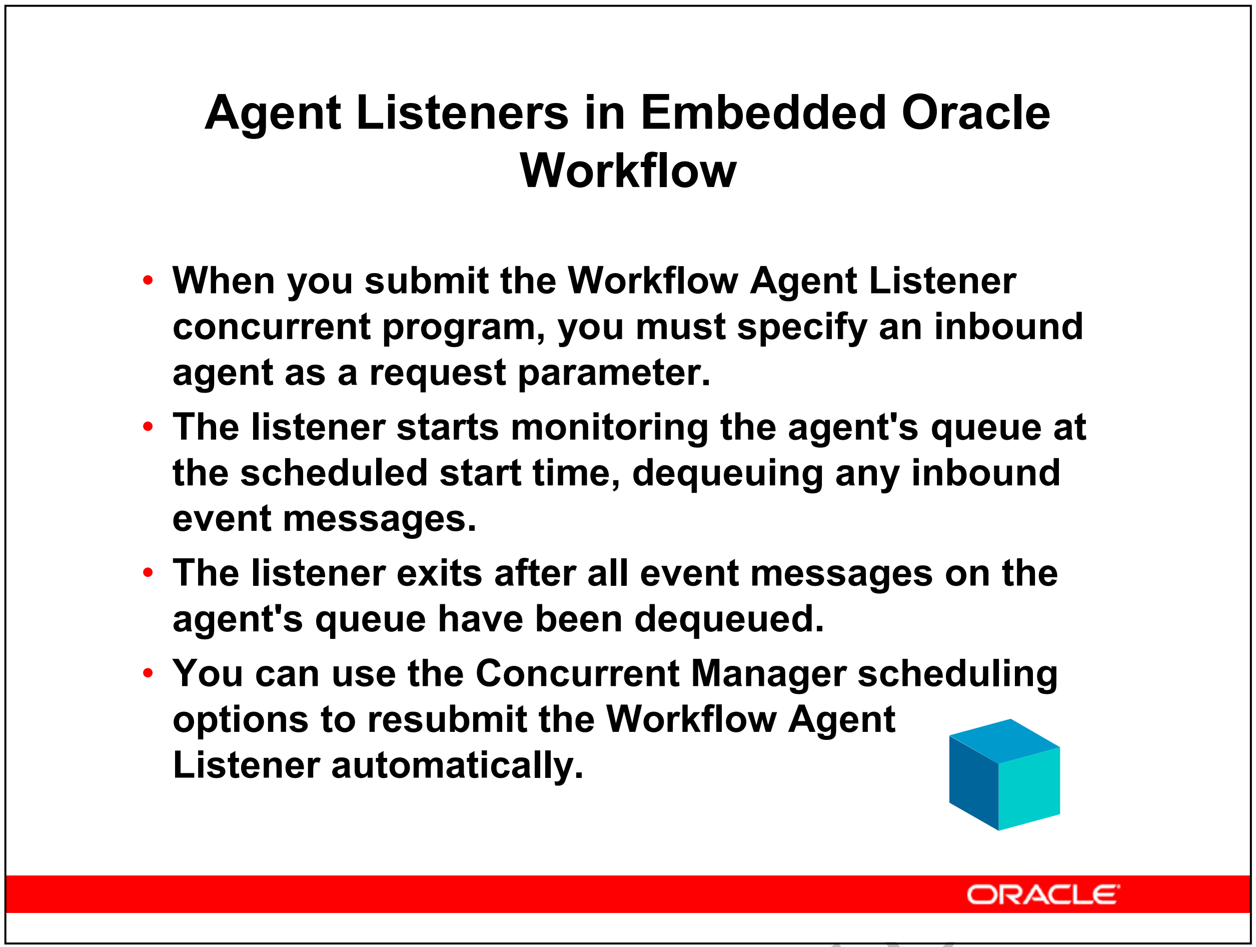
Minutes

Seconds



Updating or Deleting a Listener in Standalone Oracle Workflow

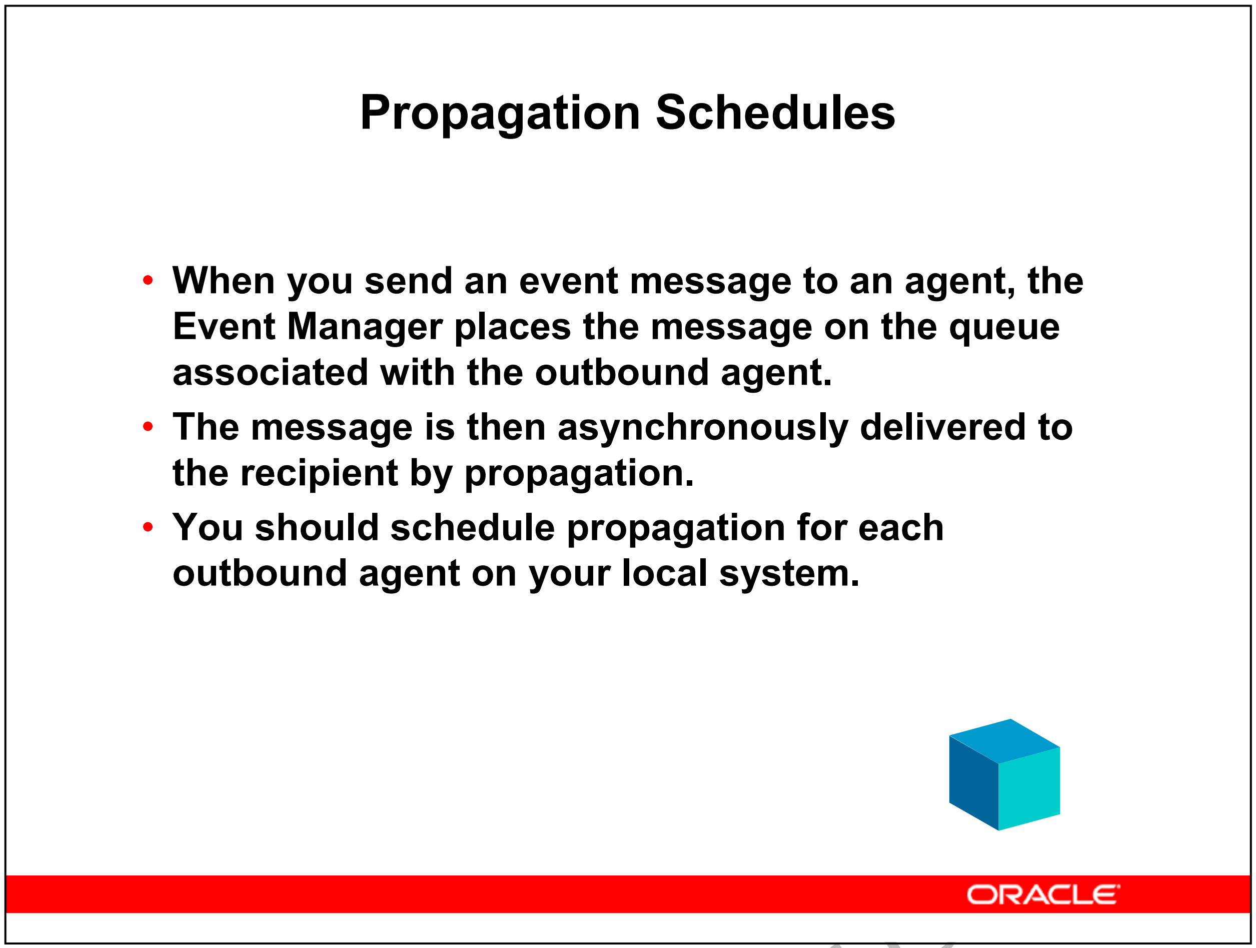
- Locate the agent you want in the Listeners for Local Inbound Agents region of the Check Setup page.
- Choose the Edit link in the Action column for that agent to open the Listeners page.
  - To update a listener, choose the pencil icon in the Edit column for that listener.
  - To delete a listener, choose the trash icon in the Delete column for that listener, and choose OK in the confirmation window that appears.



Agent Listeners in Embedded Oracle Workflow

- When you submit the Workflow Agent Listener concurrent program, you must specify an inbound agent as a request parameter.
- The listener starts monitoring the agent's queue at the scheduled start time, dequeuing any inbound event messages.
- The listener exits after all event messages on the agent's queue have been dequeued.
- You can use the Concurrent Manager scheduling options to resubmit the Workflow Agent Listener automatically.

ORACLE



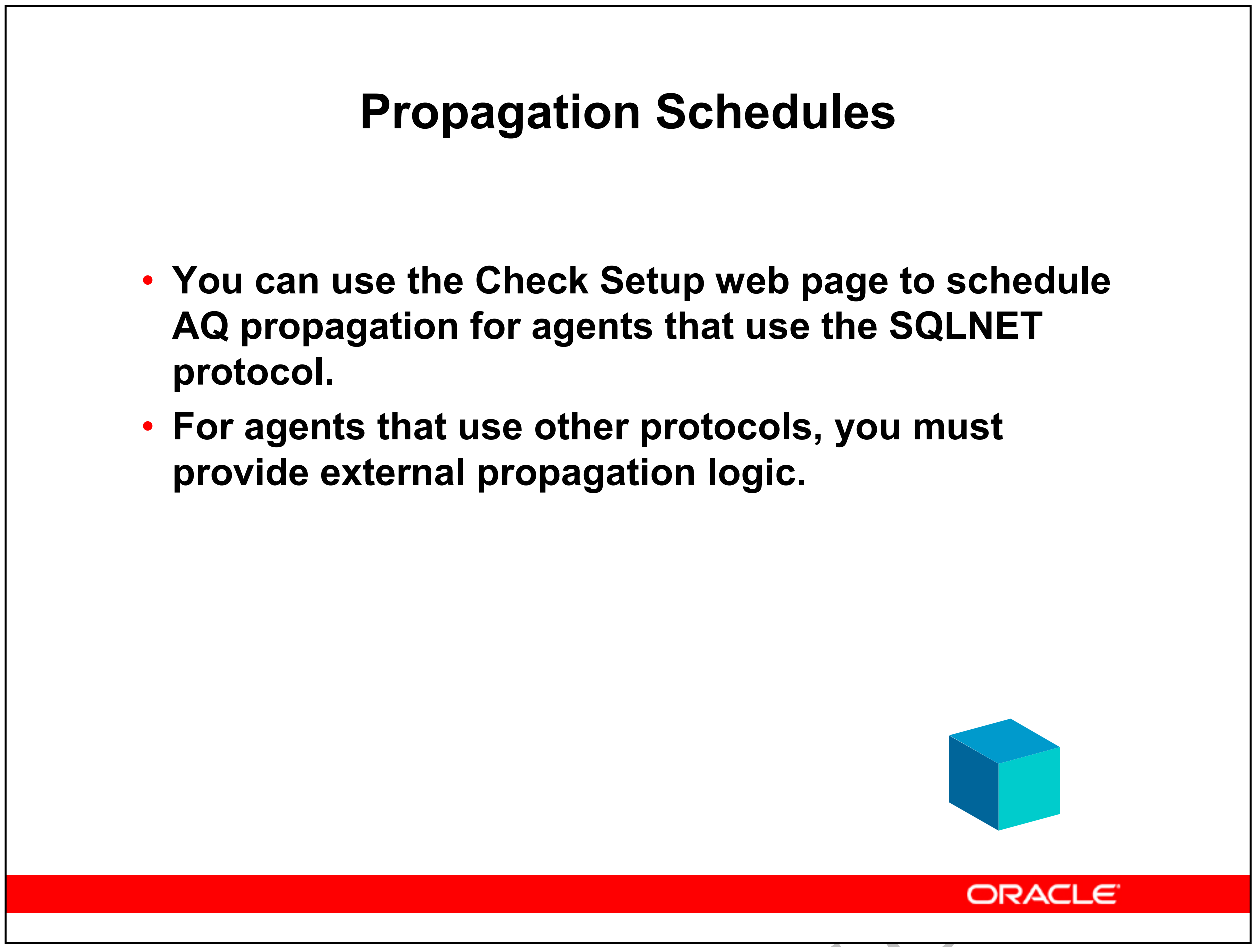
Propagation Schedules

- When you send an event message to an agent, the Event Manager places the message on the queue associated with the outbound agent.
- The message is then asynchronously delivered to the recipient by propagation.
- You should schedule propagation for each outbound agent on your local system.

ORACLE

Propagation Schedules

If you want to use the standard WF_OUT agent for event message propagation, ensure that you schedule propagation for that agent.



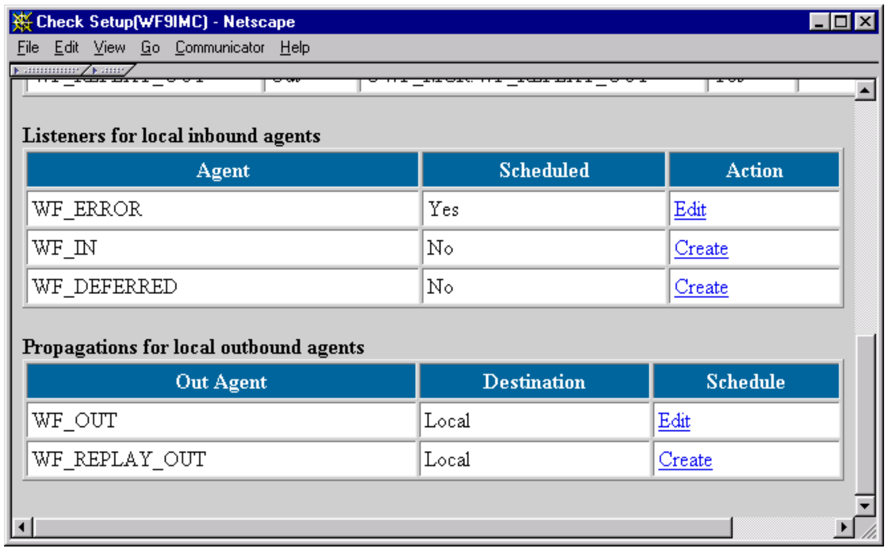
Propagation Schedules

- You can use the Check Setup web page to schedule AQ propagation for agents that use the SQLNET protocol.
- For agents that use other protocols, you must provide external propagation logic.

Propagation Schedules

You can use the Check Setup web page to schedule propagation for both standalone Oracle Workflow and Oracle Workflow embedded in Oracle E-Business Suite.

Scheduling Propagation


Check Setup(WF9IMC) - Netscape
File Edit View Go Communicator Help

Listeners for local inbound agents

Agent	Scheduled	Action
WF_ERROR	Yes	Edit
WF_IN	No	Create
WF_DEFERRED	No	Create

Propagations for local outbound agents

Out Agent	Destination	Schedule
WF_OUT	Local	Edit
WF_REPLAY_OUT	Local	Create

To Schedule Propagation for an Outbound Agent:

1. Use a web browser to connect to the Check Setup web page.

   For standalone Oracle Workflow, connect to the following URL:

   <*webagent*>/wf_setup.check_all

   Replace <*webagent*> with the base URL of the web agent configured for Oracle Workflow in your Web server.

   You can also access the Check Setup web page from the Oracle Workflow home page, <*webagent*>/wfa_html.home.

   For Oracle Workflow embedded in Oracle E-Business Suite, choose the Check Event Manager Setup option from a Workflow administrator responsibility with Event Manager functionality.

   Note: You must have workflow administrator privileges to access the Event Manager web pages.

2. In the Check Setup page, the Propagations for Local Outbound Agents region displays a list of the combinations of local outbound agents and database links that may require propagation schedules. This list matches each of your local outbound agents with each

database link to a remote system that appears in the addresses of the inbound agents you have defined. Each local outbound agent is also listed with a Local destination in the Database Link column for propagation to inbound agents on the local system.

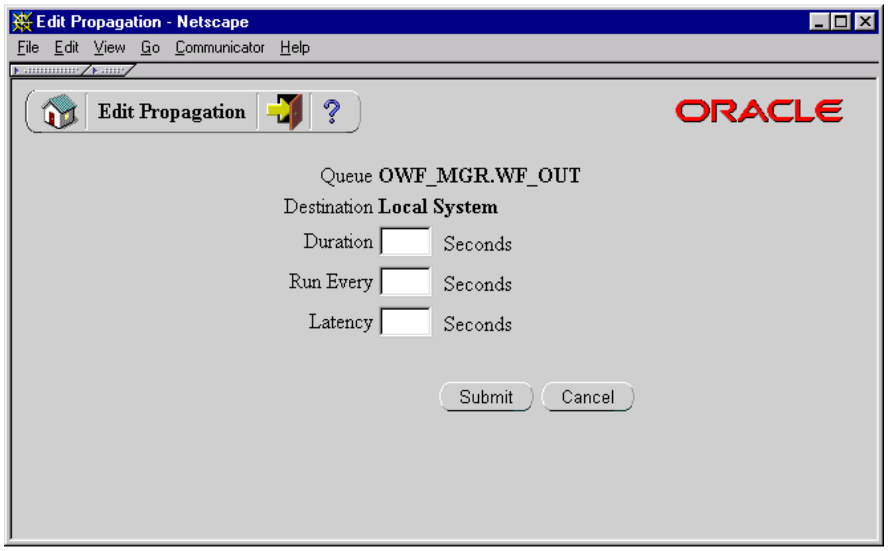
Locate the outbound agent and database link combination for which you want to schedule a propagation.

3. If no propagation is scheduled yet for the agent and database link you want, choose the Create link in the Schedule column to schedule propagation. The Edit Propagation page appears.
4. In the Duration field, enter the duration of the propagation window, in seconds.
5. In the Run Every field, enter an interval in seconds to specify how often you want a propagation window to occur.

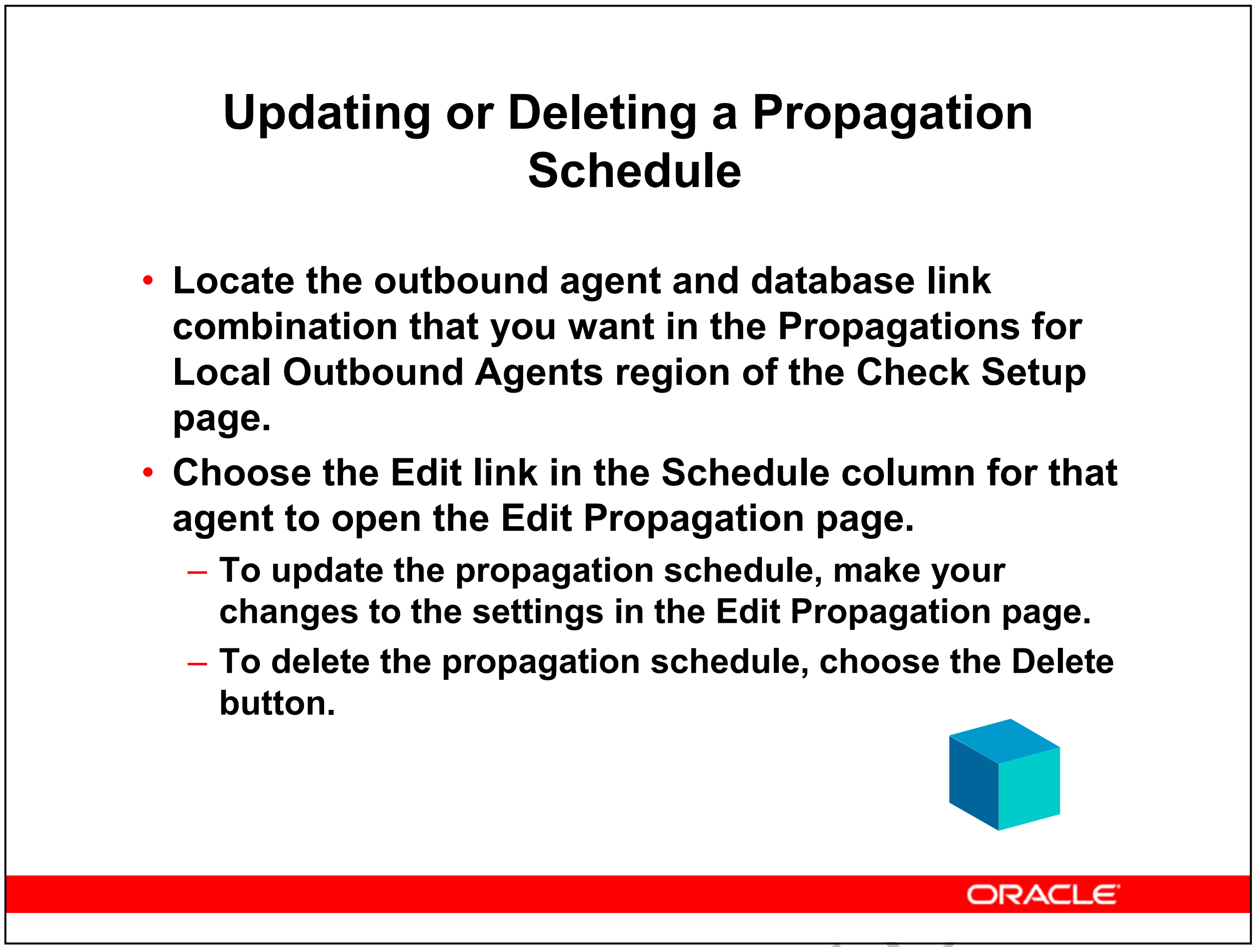
   Note: The run interval must be longer than the duration of the propagation window.
6. In the Latency field, enter a latency time in seconds to specify how long you want to wait, after all messages have been propagated, before rechecking the queue for new messages to the destination.

   The latency represents the maximum wait time during the propagation window for a message to be propagated after it is enqueued. To propagate messages as soon as possible after they are enqueued, enter a latency of zero. The default latency is 60 seconds.
7. Choose the Submit button to save the propagation schedule.

Scheduling Propagation

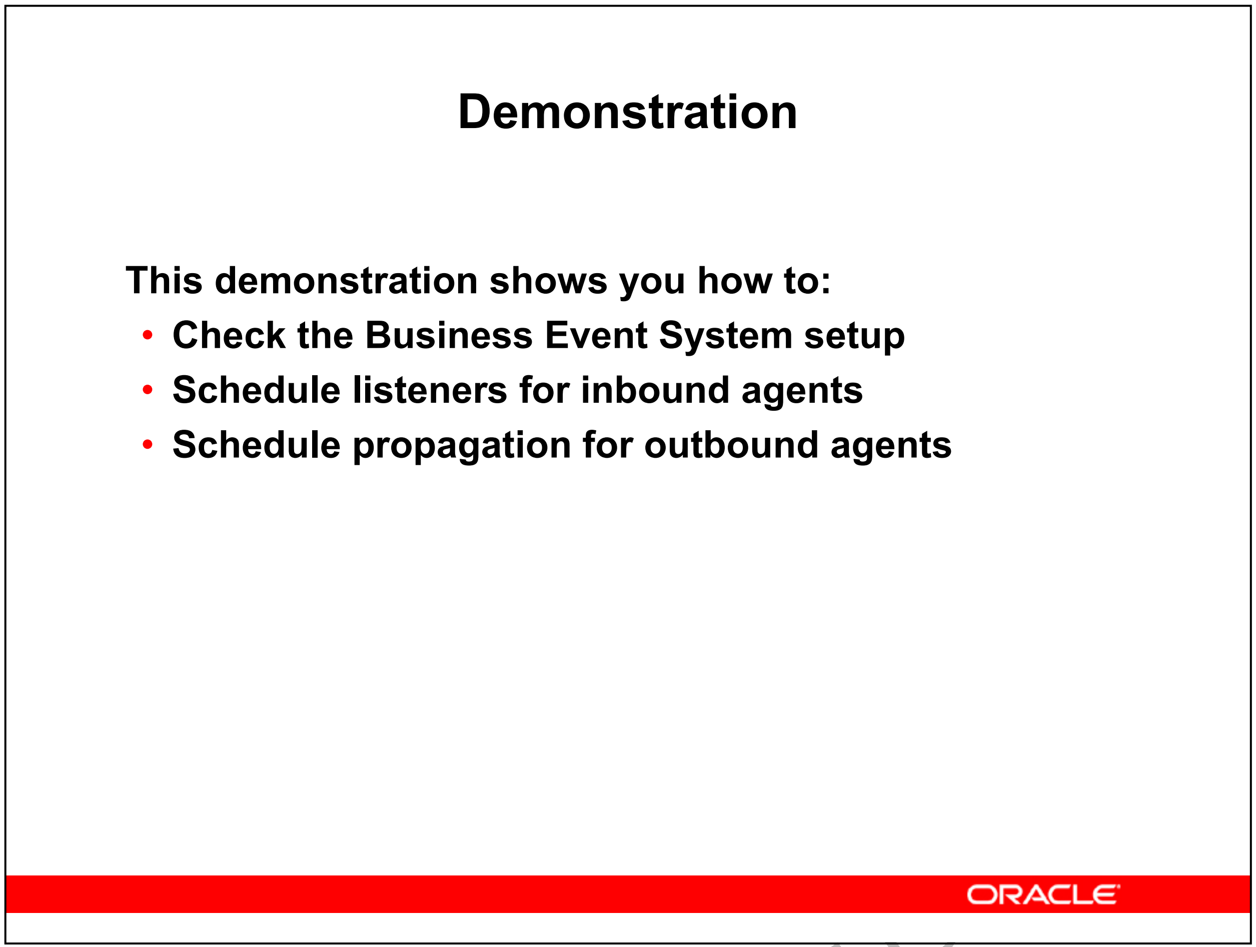

Edit Propagation - Netscape
File Edit View Go Communicator Help
Edit Propagation
ORACLE
Submit
Cancel

Queue OWF_MGR.WF_OUT
Destination Local System
Duration ___ Seconds
Run Every ___ Seconds
Latency ___ Seconds



Updating or Deleting a Propagation Schedule

- **Locate the outbound agent and database link combination that you want in the Propagations for Local Outbound Agents region of the Check Setup page.**
- **Choose the Edit link in the Schedule column for that agent to open the Edit Propagation page.**
  - **To update the propagation schedule, make your changes to the settings in the Edit Propagation page.**
  - **To delete the propagation schedule, choose the Delete button.**

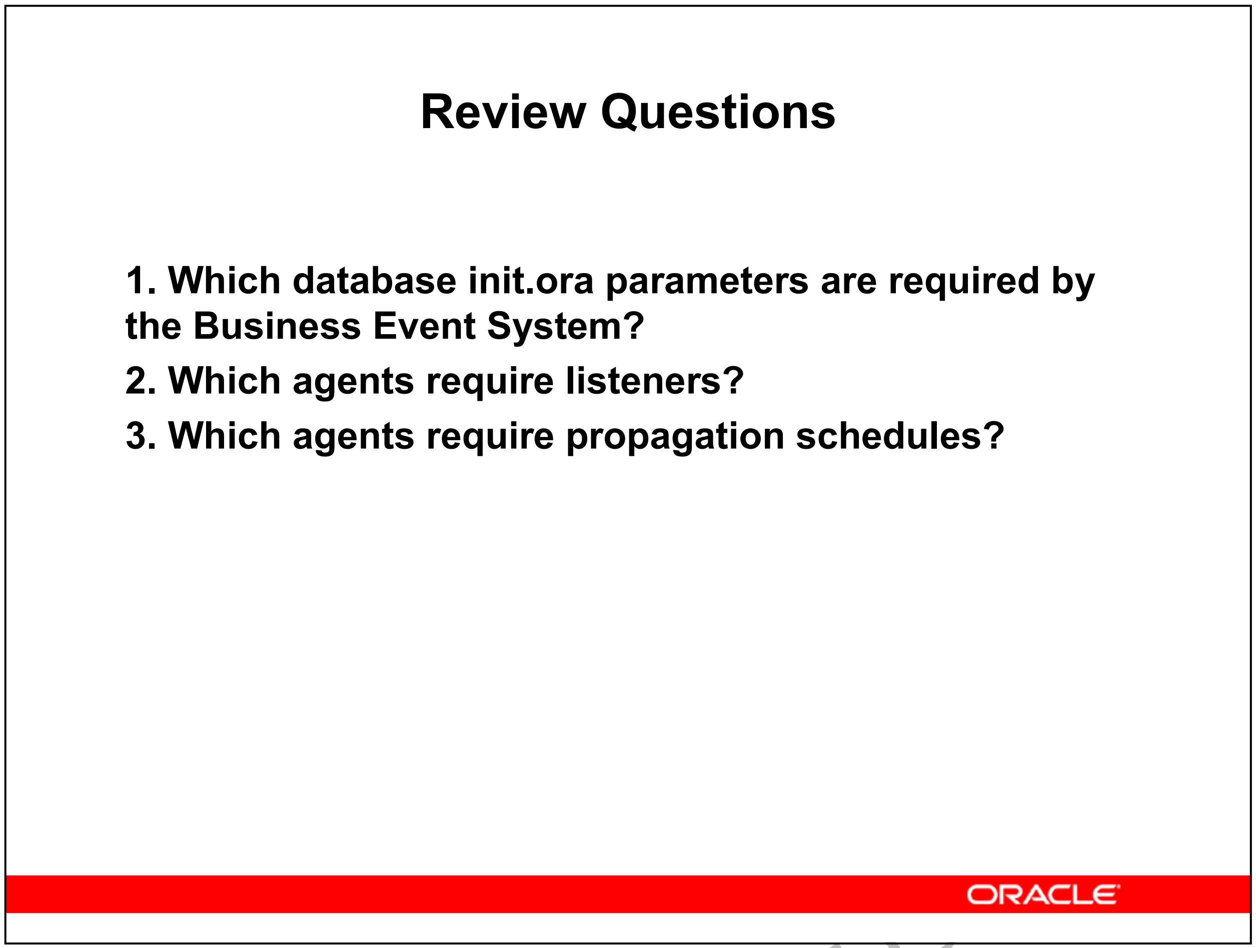


Demonstration

This demonstration shows you how to:

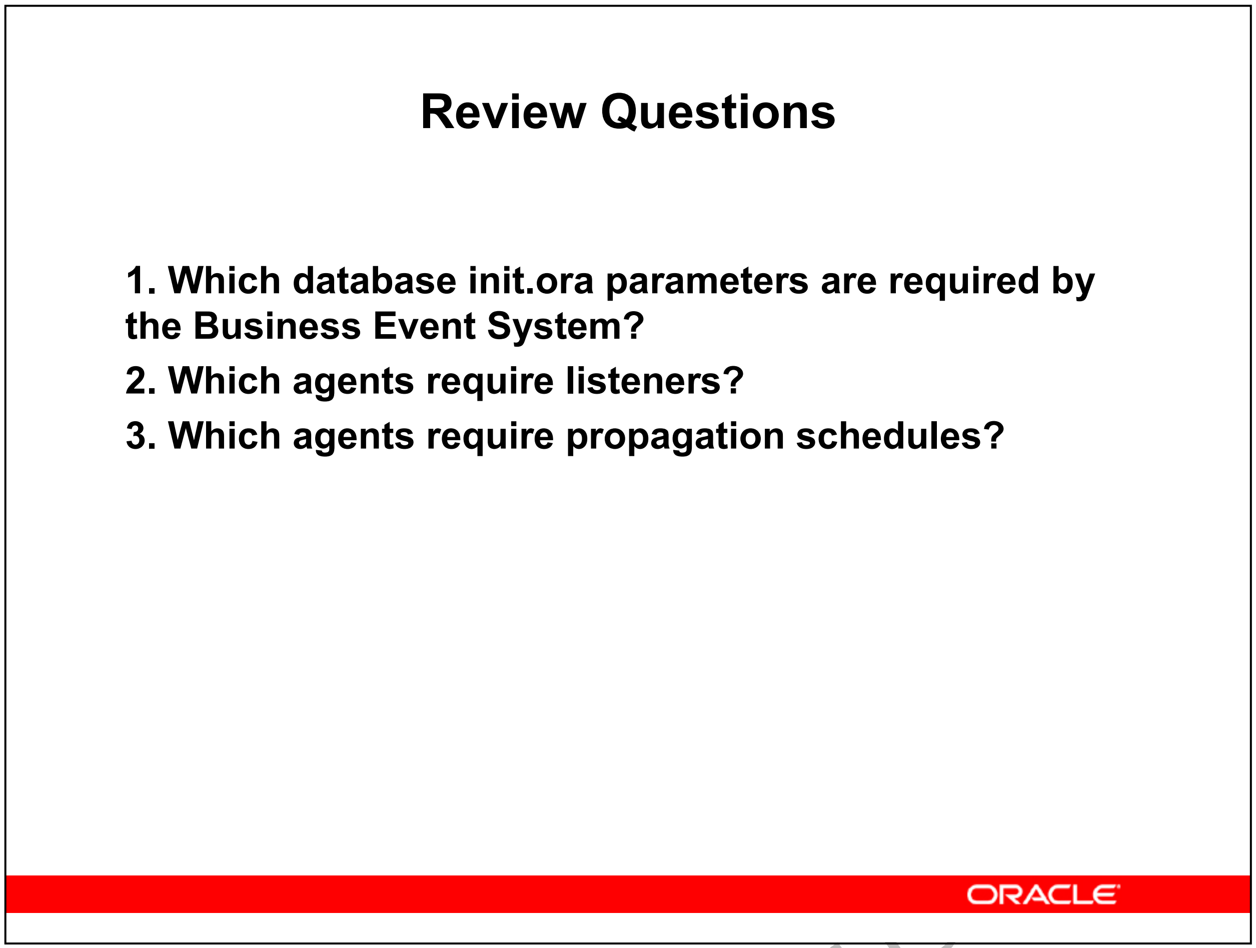
- Check the Business Event System setup
- Schedule listeners for inbound agents
- Schedule propagation for outbound agents

Review Questions



Review Questions

1. Which database init.ora parameters are required by the Business Event System?
2. Which agents require listeners?
3. Which agents require propagation schedules?



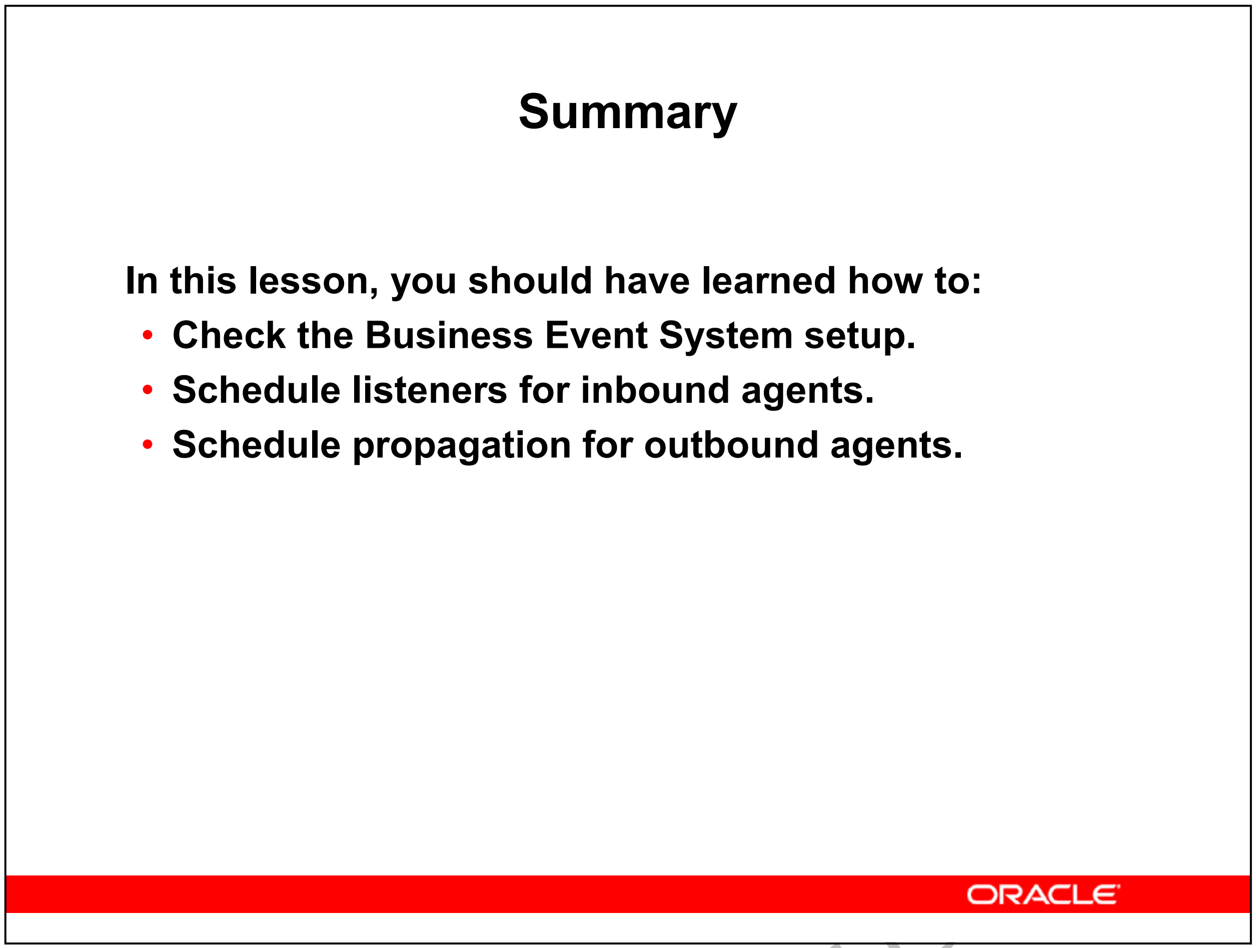
Review Questions

1. Which database init.ora parameters are required by the Business Event System?
2. Which agents require listeners?
3. Which agents require propagation schedules?

ORACLE

Review Questions and Solutions

1. Which database init.ora parameters are required by the Business Event System?
   - AQ_TM_PROCESSES
   - JOB_QUEUE_INTERVAL (Oracle8*i* only)
   - JOB_QUEUE_PROCESSES
2. Which agents require listeners?
   Local inbound agents require listeners.
3. Which agents require propagation schedules?
   Local outbound agents require propagation schedules.



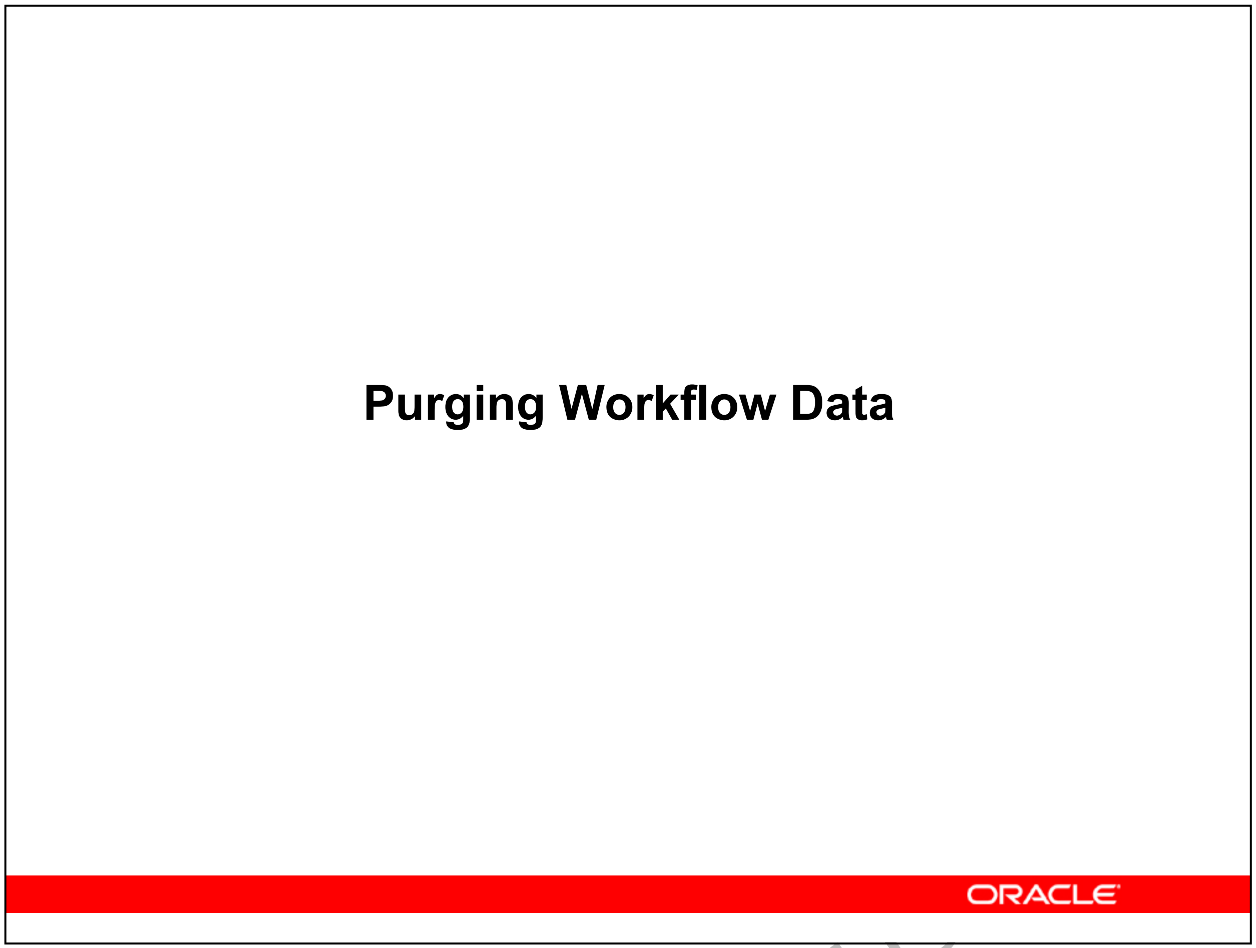
Summary

In this lesson, you should have learned how to:

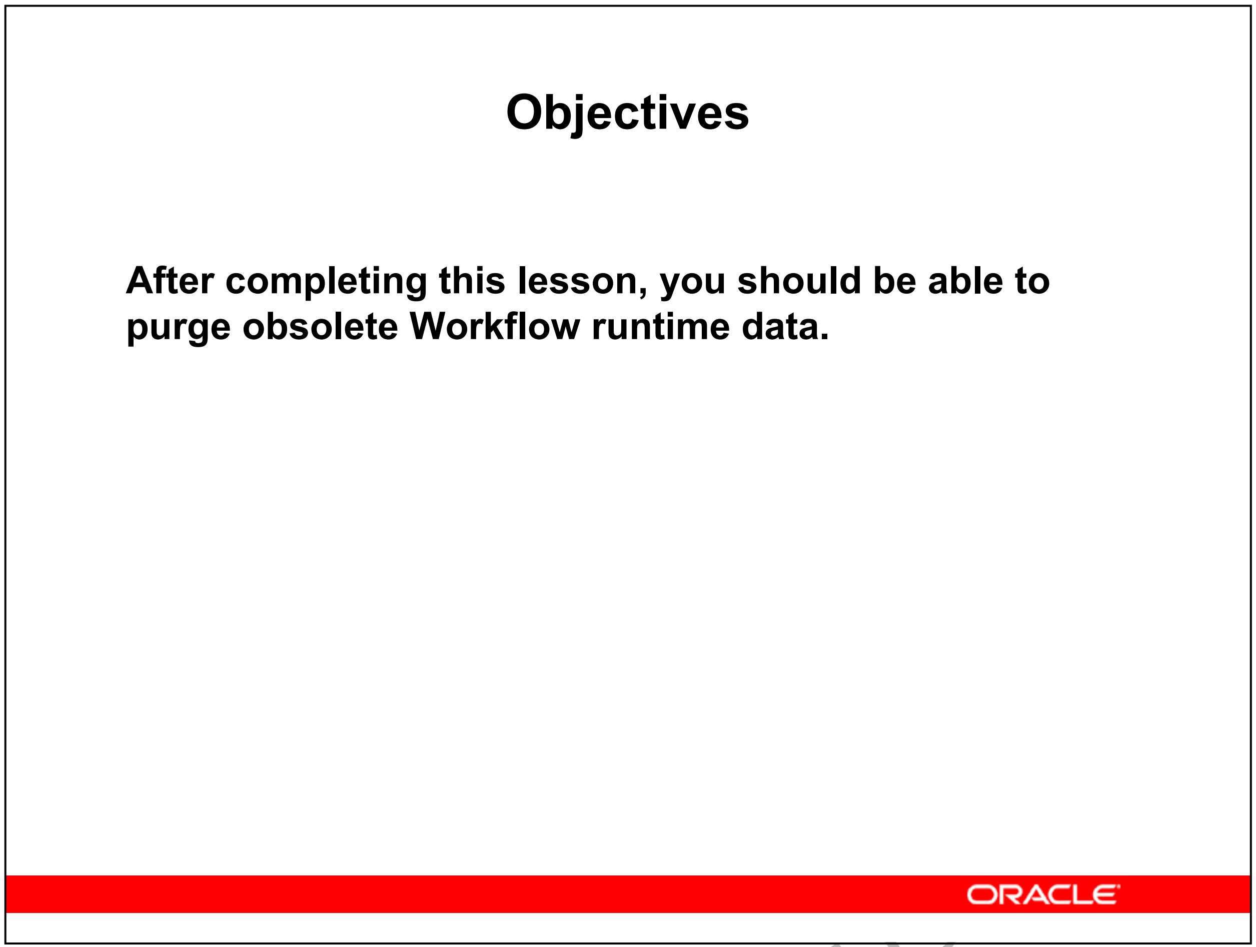
- Check the Business Event System setup.
- Schedule listeners for inbound agents.
- Schedule propagation for outbound agents.

Purging Workflow Data

Chapter 27

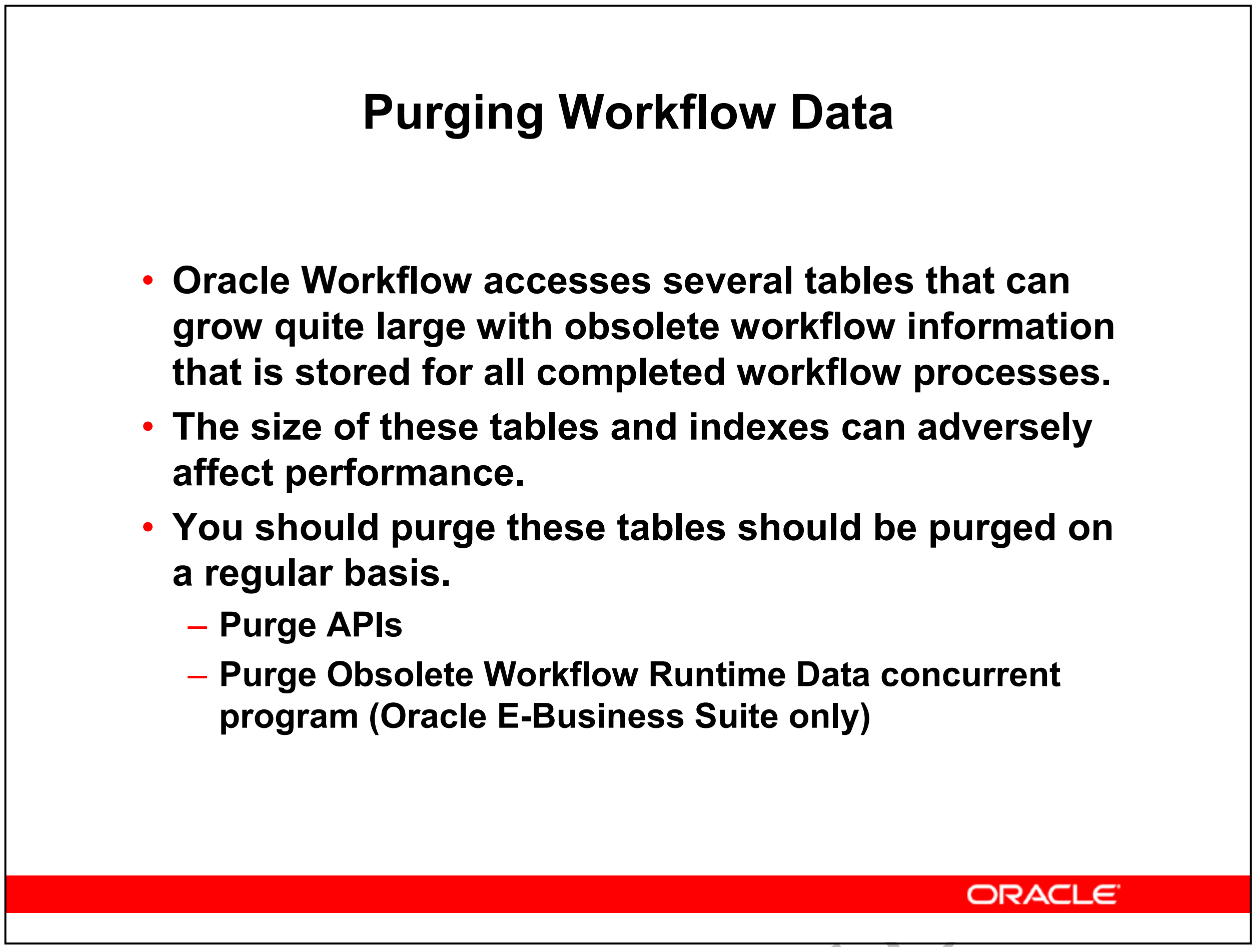


Purging Workflow Data



Objectives

After completing this lesson, you should be able to purge obsolete Workflow runtime data.



Purging Workflow Data

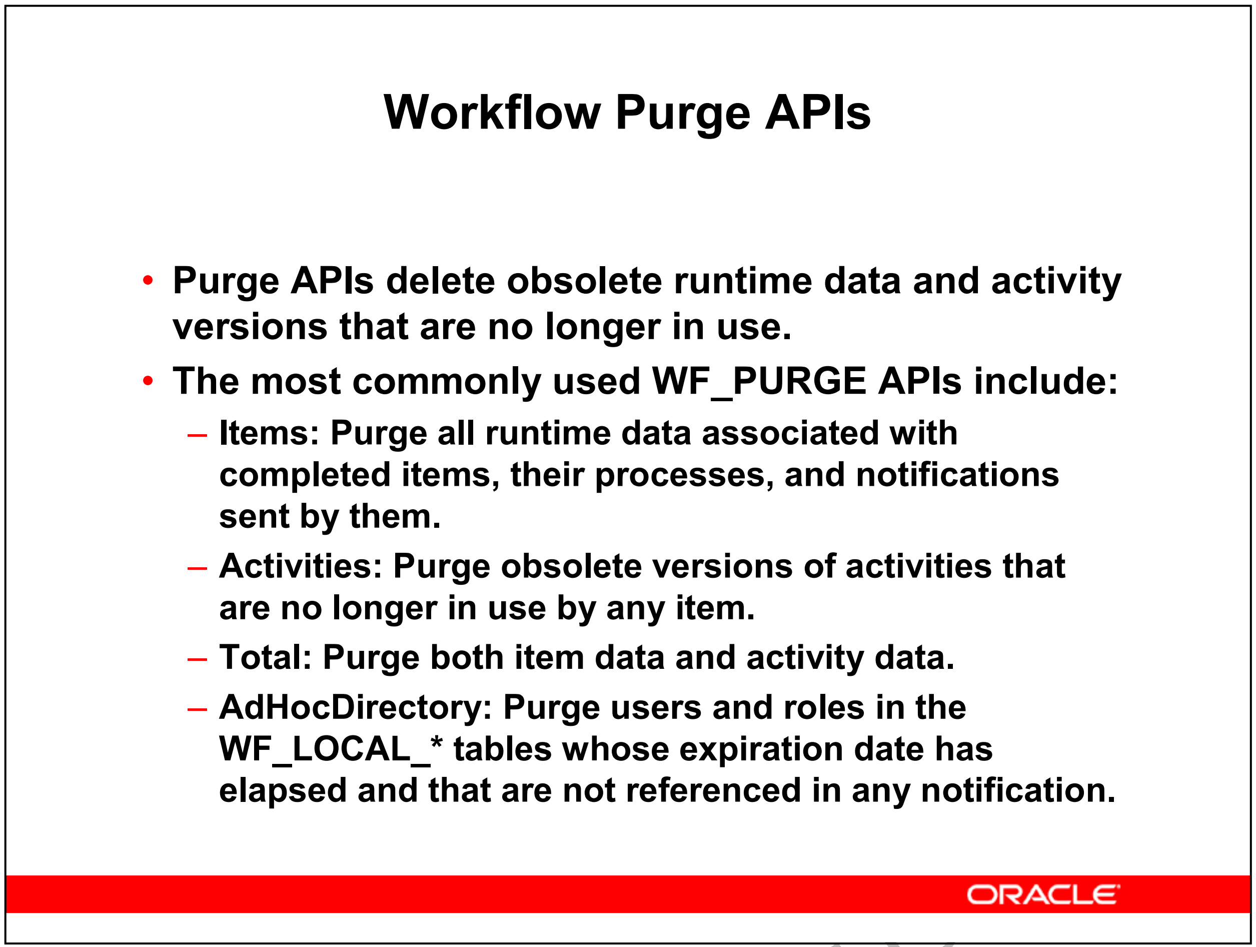
- Oracle Workflow accesses several tables that can grow quite large with obsolete workflow information that is stored for all completed workflow processes.
- The size of these tables and indexes can adversely affect performance.
- You should purge these tables should be purged on a regular basis.
  - Purge APIs
  - Purge Obsolete Workflow Runtime Data concurrent program (Oracle E-Business Suite only)

Purging Workflow Data

The persistence type of an item type controls when Oracle Workflow purges runtime status information for work items. The persistence values are:

- Temporary: Item will be deleted in *n* days
- Permanent: Item will be deleted only when forced

Note: For a work item to be considered eligible for purging, all activities must be complete. This includes FYI notifications, which must be closed.



Workflow Purge APIs

- Purge APIs delete obsolete runtime data and activity versions that are no longer in use.
- The most commonly used WF_PURGE APIs include:
  - Items: Purge all runtime data associated with completed items, their processes, and notifications sent by them.
  - Activities: Purge obsolete versions of activities that are no longer in use by any item.
  - Total: Purge both item data and activity data.
  - AdHocDirectory: Purge users and roles in the WF_LOCAL_* tables whose expiration date has elapsed and that are not referenced in any notification.

ORACLE

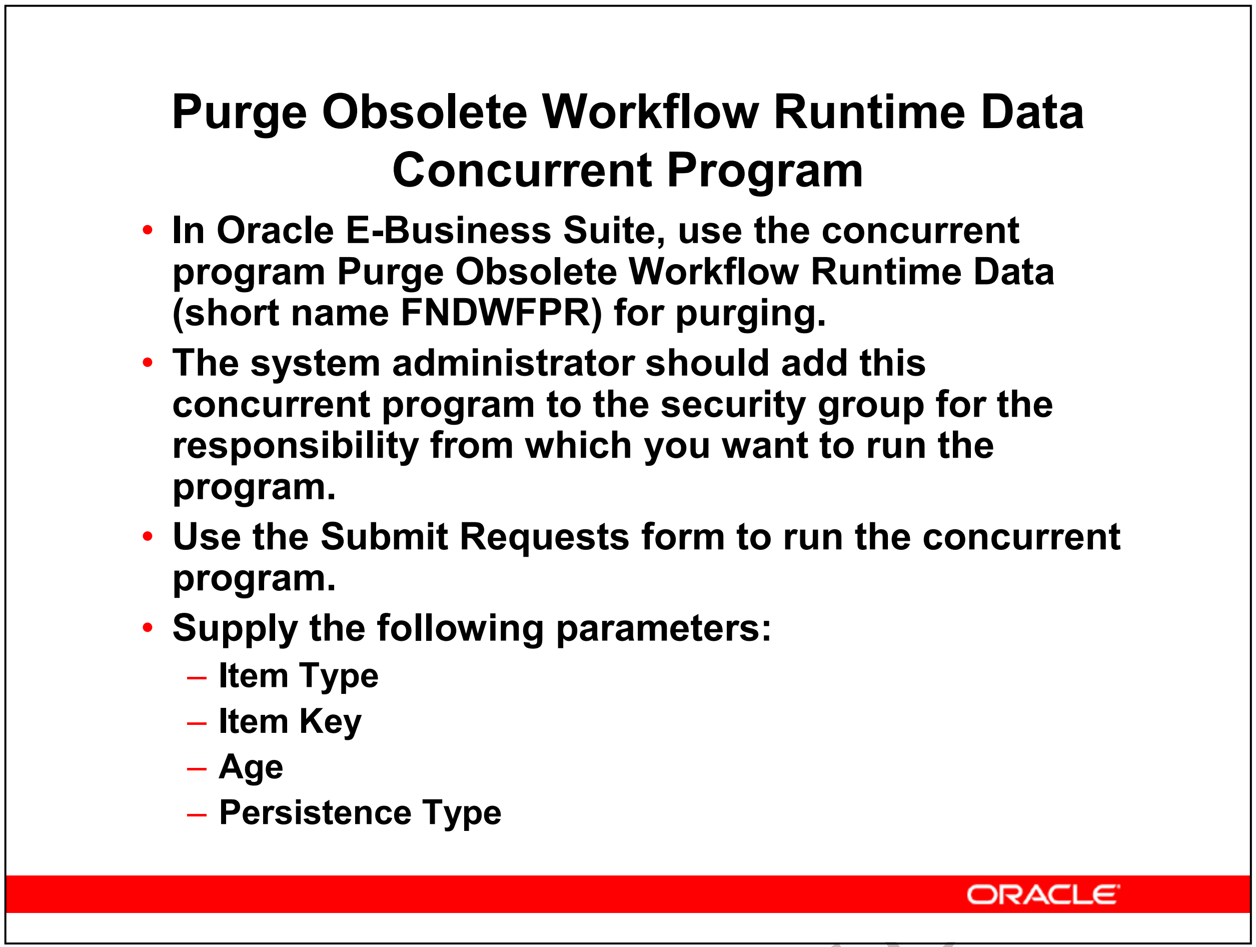
WF_PURGE APIs

Many of the Purge APIs accept the following parameters:

- Item Type: The item type associated with the obsolete runtime data you want to delete. Leave this parameter null to delete obsolete data for all item types.
- Item Key: A string generated from the application object's primary key. The string uniquely identifies the item within an item type. Leave this parameter null to purge all items in the specified item type.
- End Date: A specified date to delete up to.

Note: The WF_PURGE APIs only purge data associated with Temporary item types whose persistence, in days, has expired. A persistence type PL/SQL variable is set to 'TEMP' (Temporary) by default and should not be changed.

Use the WF_PURGE.TotalPERM API to delete all eligible obsolete runtime data associated with item types of with a persistence type of Permanent.

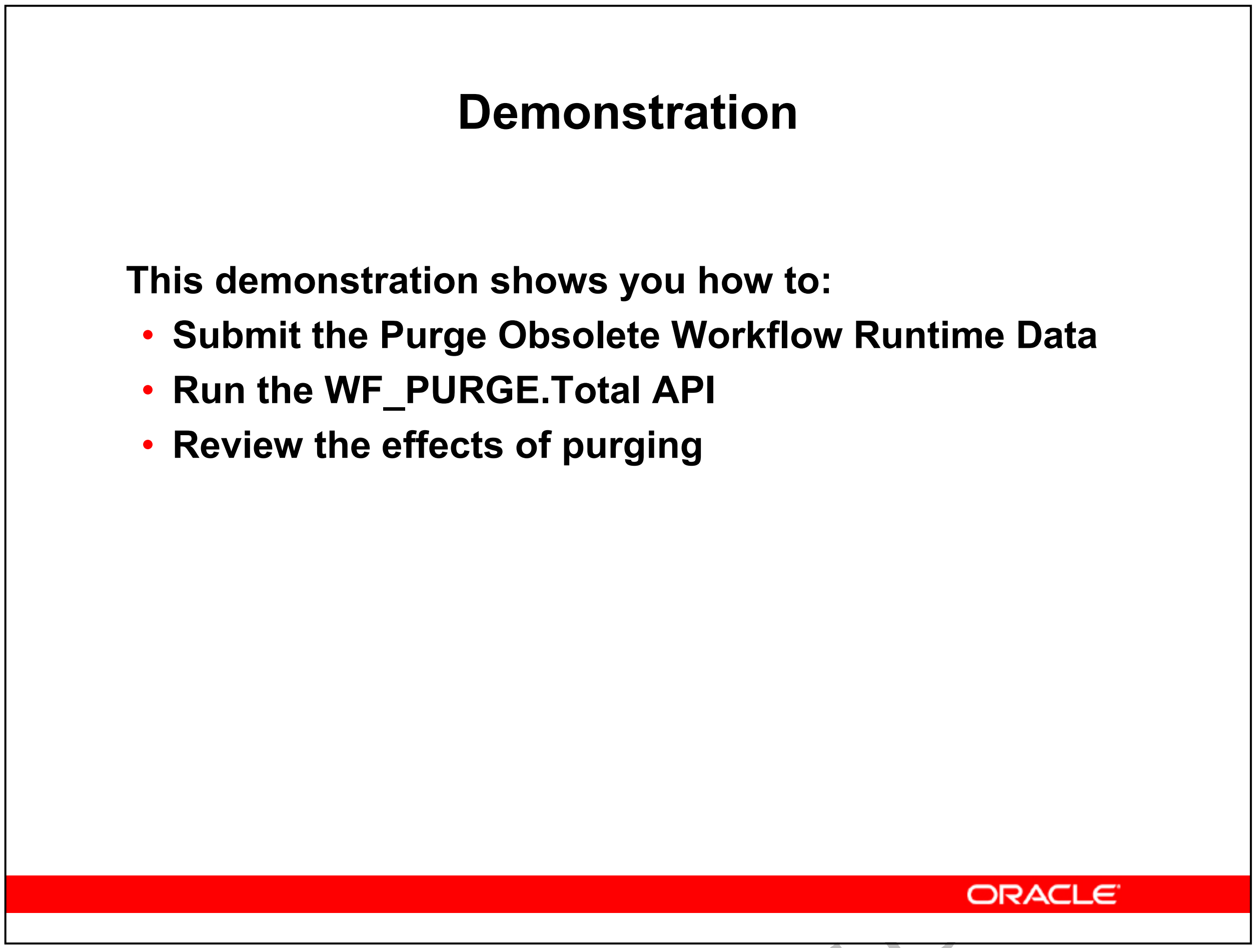


Purge Obsolete Workflow Runtime Data Concurrent Program

- In Oracle E-Business Suite, use the concurrent program Purge Obsolete Workflow Runtime Data (short name FNDWFPR) for purging.
- The system administrator should add this concurrent program to the security group for the responsibility from which you want to run the program.
- Use the Submit Requests form to run the concurrent program.
- Supply the following parameters:
  - Item Type
  - Item Key
  - Age
  - Persistence Type

Concurrent Program Parameters for Purging

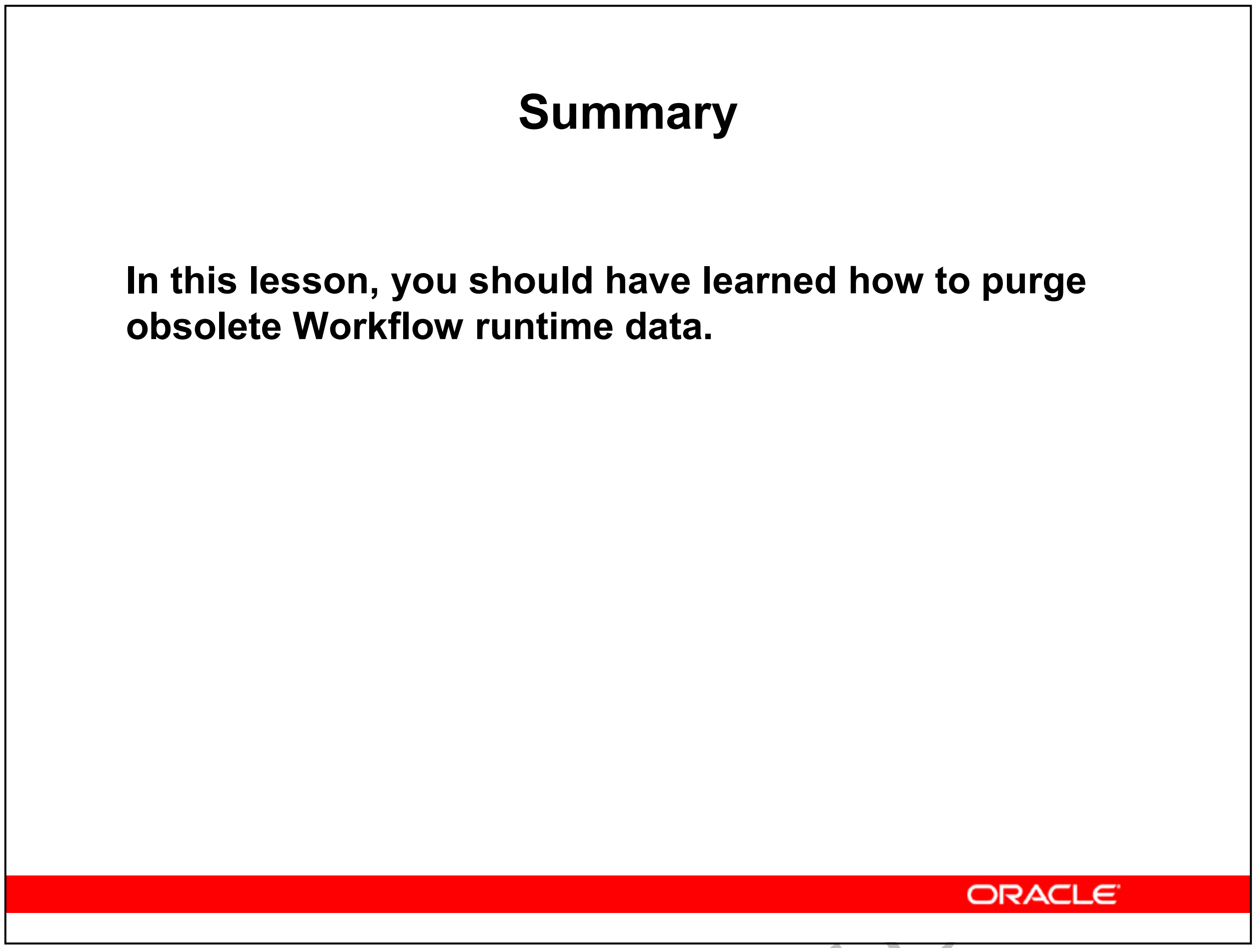
- Item Type - Item type associated with the obsolete runtime data you want to delete. Leave null to delete obsolete data for all item types.
- Item Key - String generated from the application object's primary key. The string uniquely identifies the item within an item type. If null, purges all items in the specified item type.
- Age - Minimum age of data to purge, in days if Persistence Type is set to 'Temporary'. Default is 0.
- Persistence Type - Persistence type to be purged, either 'Temporary' or 'Permanent'. Default is 'Temporary'.



Demonstration

This demonstration shows you how to:

- Submit the Purge Obsolete Workflow Runtime Data
- Run the WF_PURGE.Total API
- Review the effects of purging



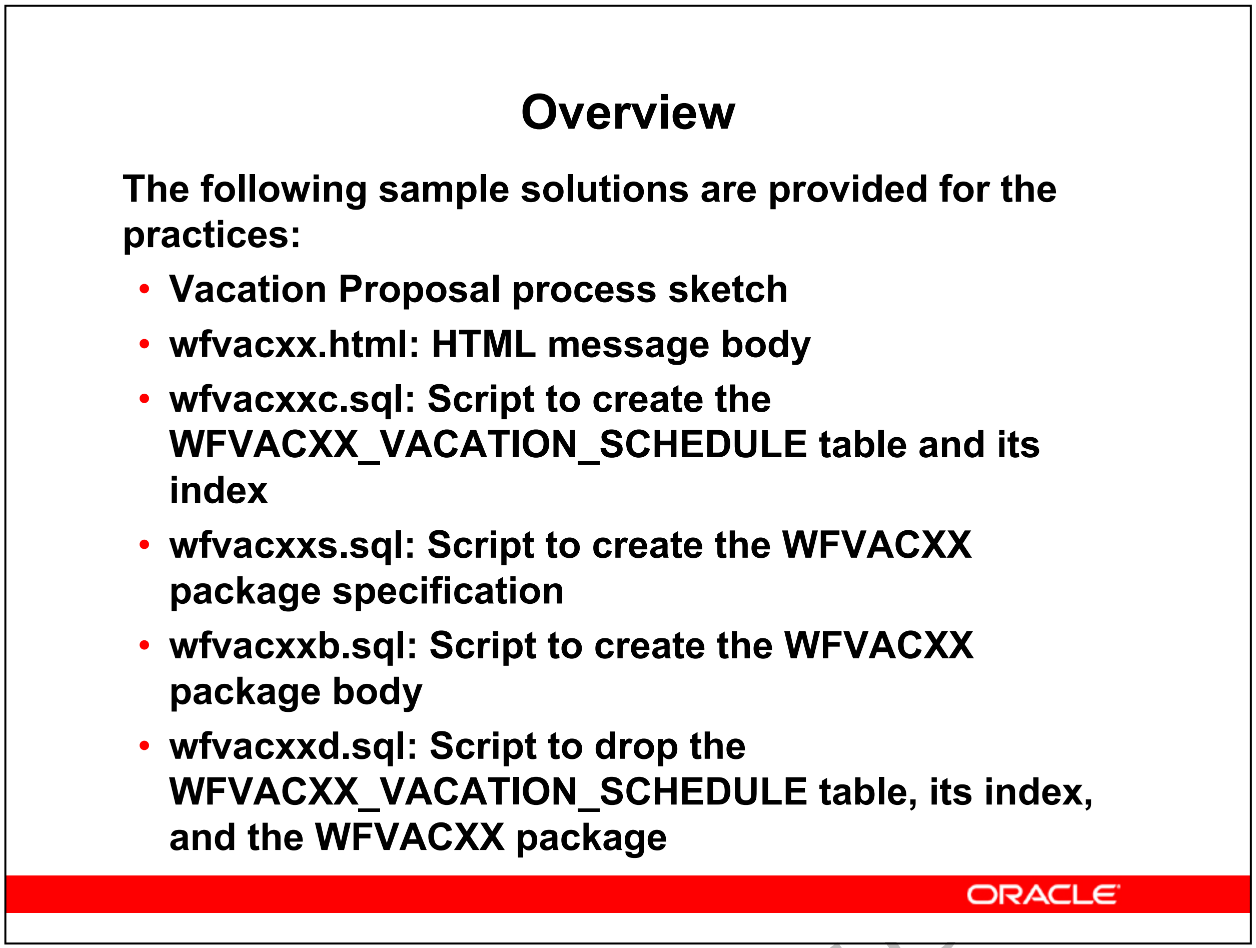
Summary

In this lesson, you should have learned how to purge obsolete Workflow runtime data.

Sample Solutions

Chapter 28

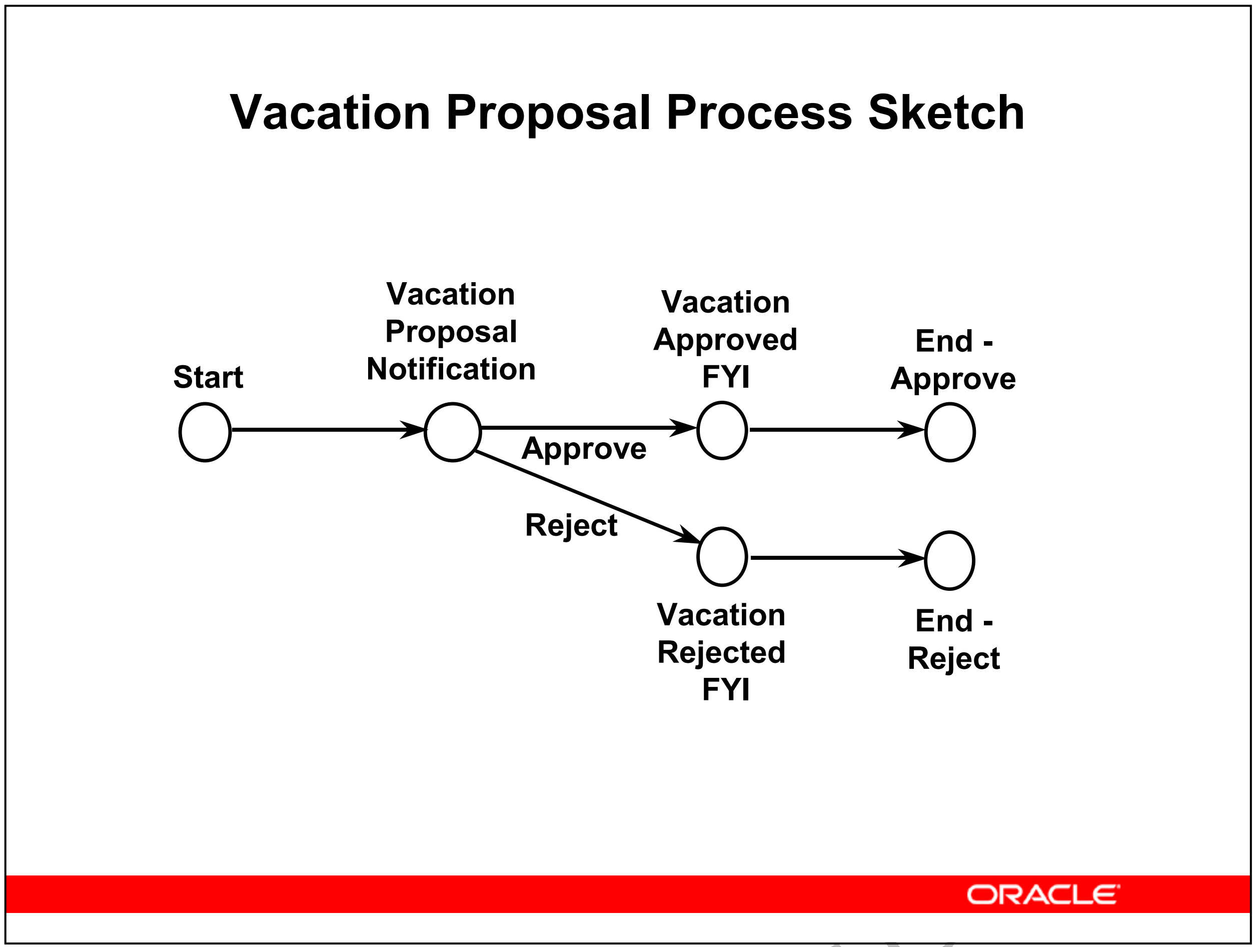
Sample Solutions



Overview

The following sample solutions are provided for the practices:

- **Vacation Proposal process sketch**
- **wfvacxx.html: HTML message body**
- **wfvacxxc.sql: Script to create the WFVACXX_VACATION_SCHEDULE table and its index**
- **wfvacxxs.sql: Script to create the WFVACXX package specification**
- **wfvacxxb.sql: Script to create the WFVACXX package body**
- **wfvacxxd.sql: Script to drop the WFVACXX_VACATION_SCHEDULE table, its index, and the WFVACXX package**

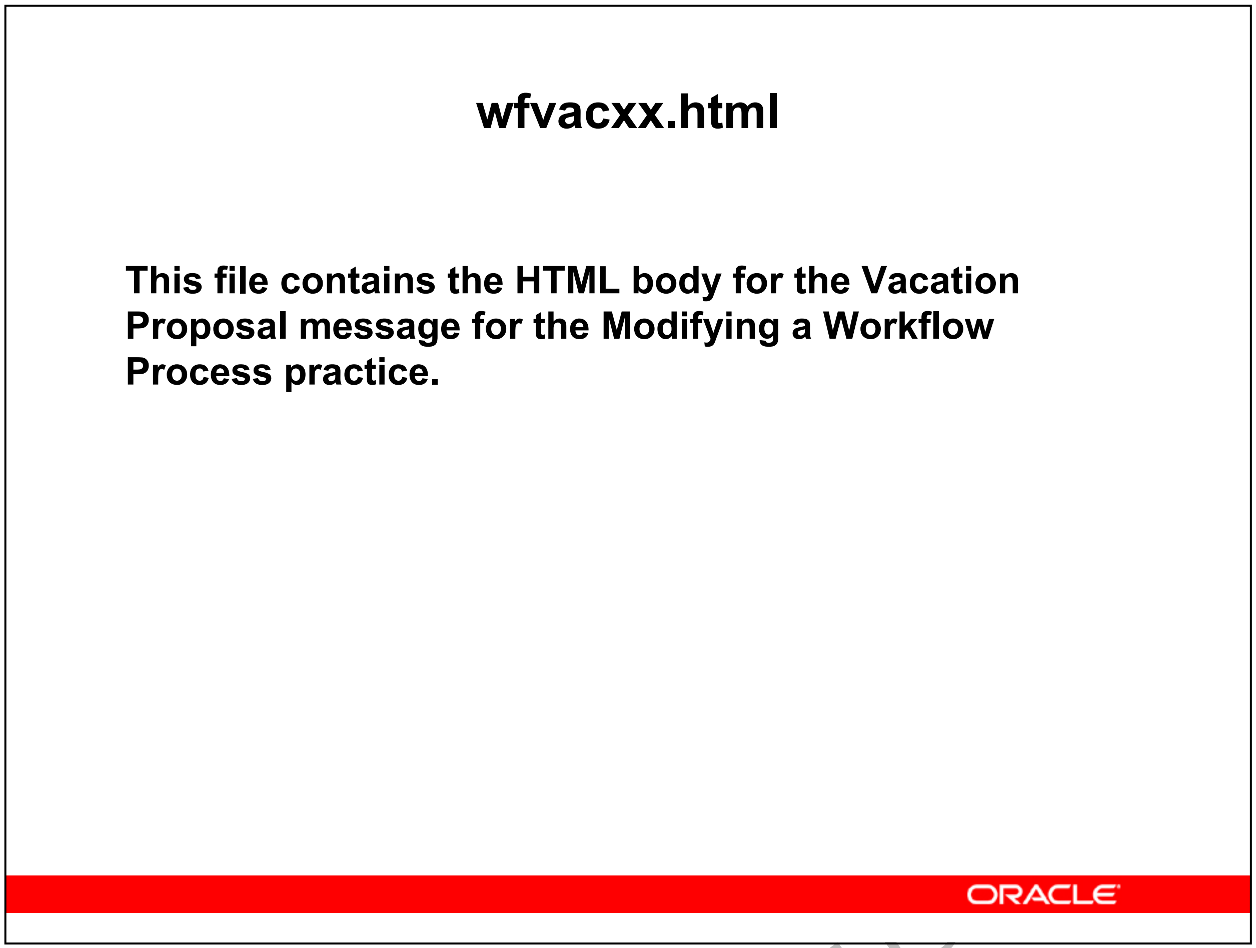


Vacation Proposal Process Sketch

$$\begin{tikzcd}
\text{Start} \arrow[r] & \text{Vacation Proposal Notification} \arrow[r, "\text{Approve}"'] \arrow[dr, "\text{Reject}"'] & \text{Vacation Approved FYI} \arrow[r] & \text{End - Approve} \\
 & & \text{Vacation Rejected FYI} \arrow[r] & \text{End - Reject}
\end{tikzcd}$$

Vacation Proposal Process Sketch

This sketch shows a sample plan for the Vacation Proposal process as described in the Planning a Workflow Process practice.

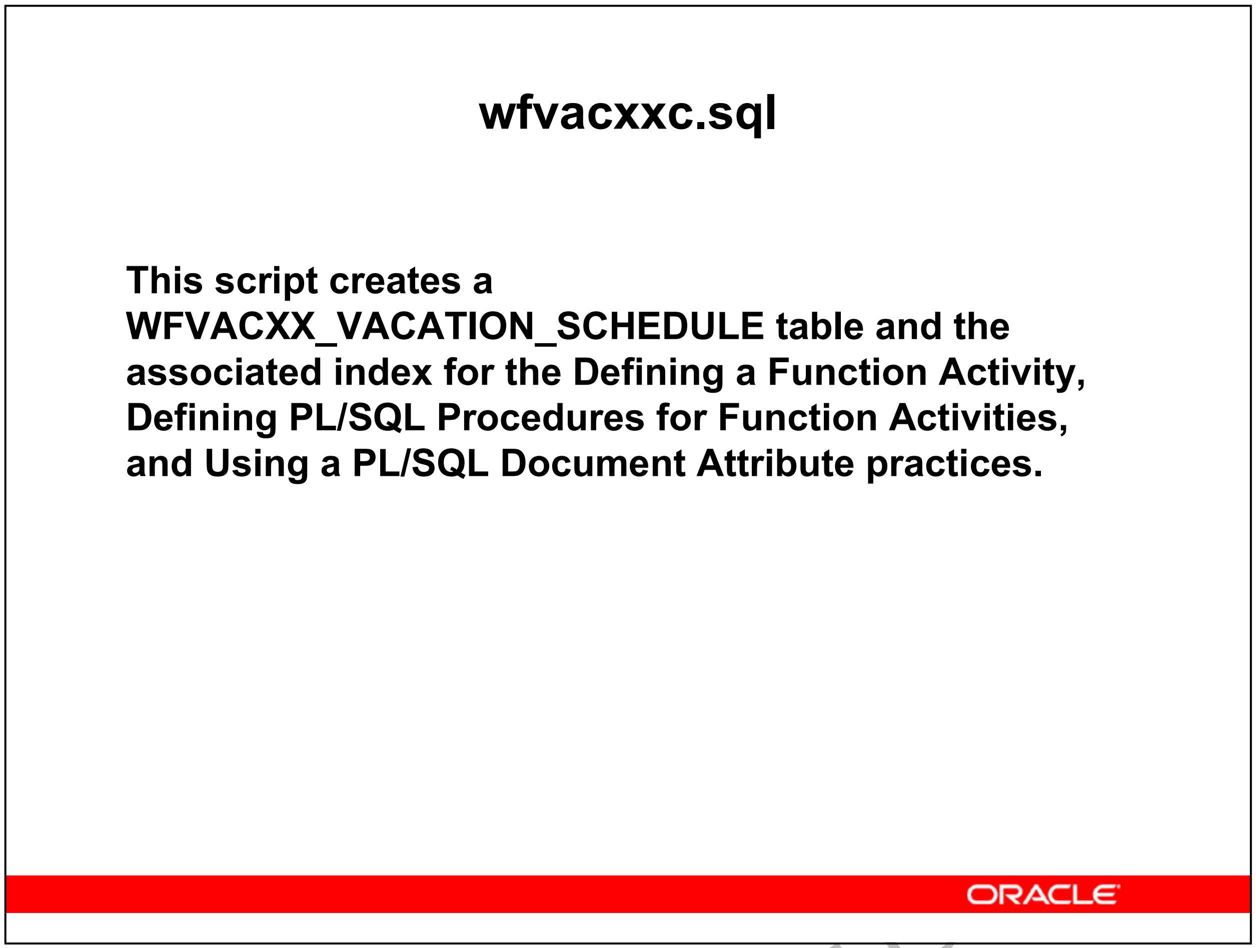


wfvacxx.html

This file contains the HTML body for the Vacation Proposal message for the Modifying a Workflow Process practice.

wfvacxx.html

```
<TABLE>
<TR><TD><img SRC="/OA_MEDIA/calendar.gif"></TD></TR>
<TR><TD>Vacation Proposal from <B>&REQUESTOR</B> for
     <B>&FROM_DATE</B> to <B>&TO_DATE</B></TD></TR>
</TABLE>
Please approve vacation as proposed or suggest alternate dates.
```



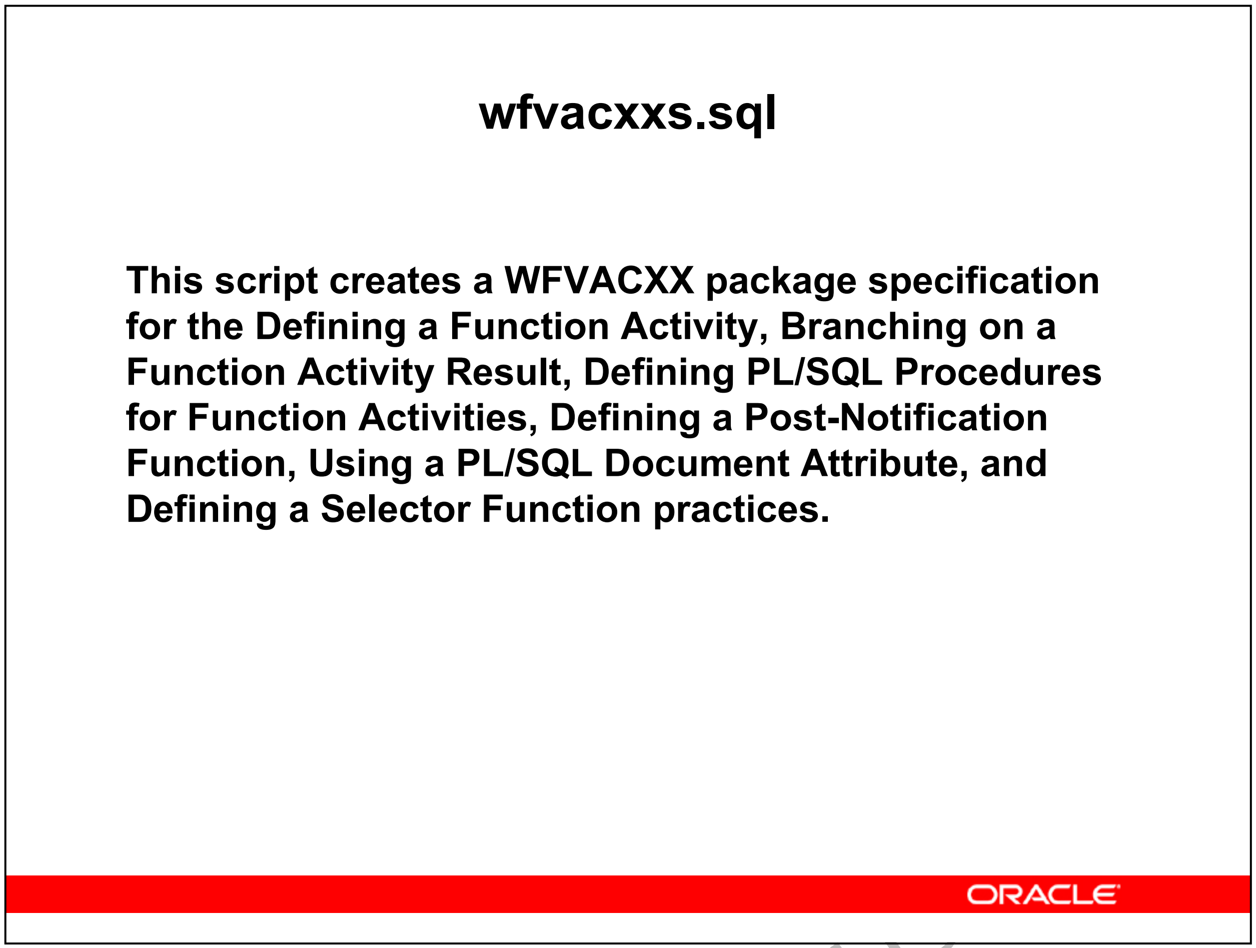
wfvacxxc.sql

This script creates a WFVACXX_VACATION_SCHEDULE table and the associated index for the Defining a Function Activity, Defining PL/SQL Procedures for Function Activities, and Using a PL/SQL Document Attribute practices.

wfvacxxc.sql

```
/*=======================================================================+
 |  Copyright (c) 1995 Oracle Corporation Redwood Shores, California, USA|
 |                           All rights reserved.                        |
 +=======================================================================+
 | FILENAME
 |   wfvacxxc.sql
 |
 | DESCRIPTION
 |   Create Workflow Vacation Schedule table and index
 | NOTES
 |   This file is a SAMPLE that should be modified with your own
 |   names before installing.  Names that include
 |   XX should be replaced with values for your implementation.
 |
 *=======================================================================*/

/* $Header$ */
```

```
whenever sqlerror continue;

drop   table WFVACXX_VACATION_SCHEDULE;
create table WFVACXX_VACATION_SCHEDULE
(    requestor_username  varchar2(30) not null,
     approver_username   varchar2(30) not null,
     from_date           date         not null,
     to_date             date         not null);
--
create index WFVACXX_VACATION_SCHEDULE_N1 on WFVACXX_VACATION_SCHEDULE
(requestor_username,approver_username);
--
commit;

--exit
```



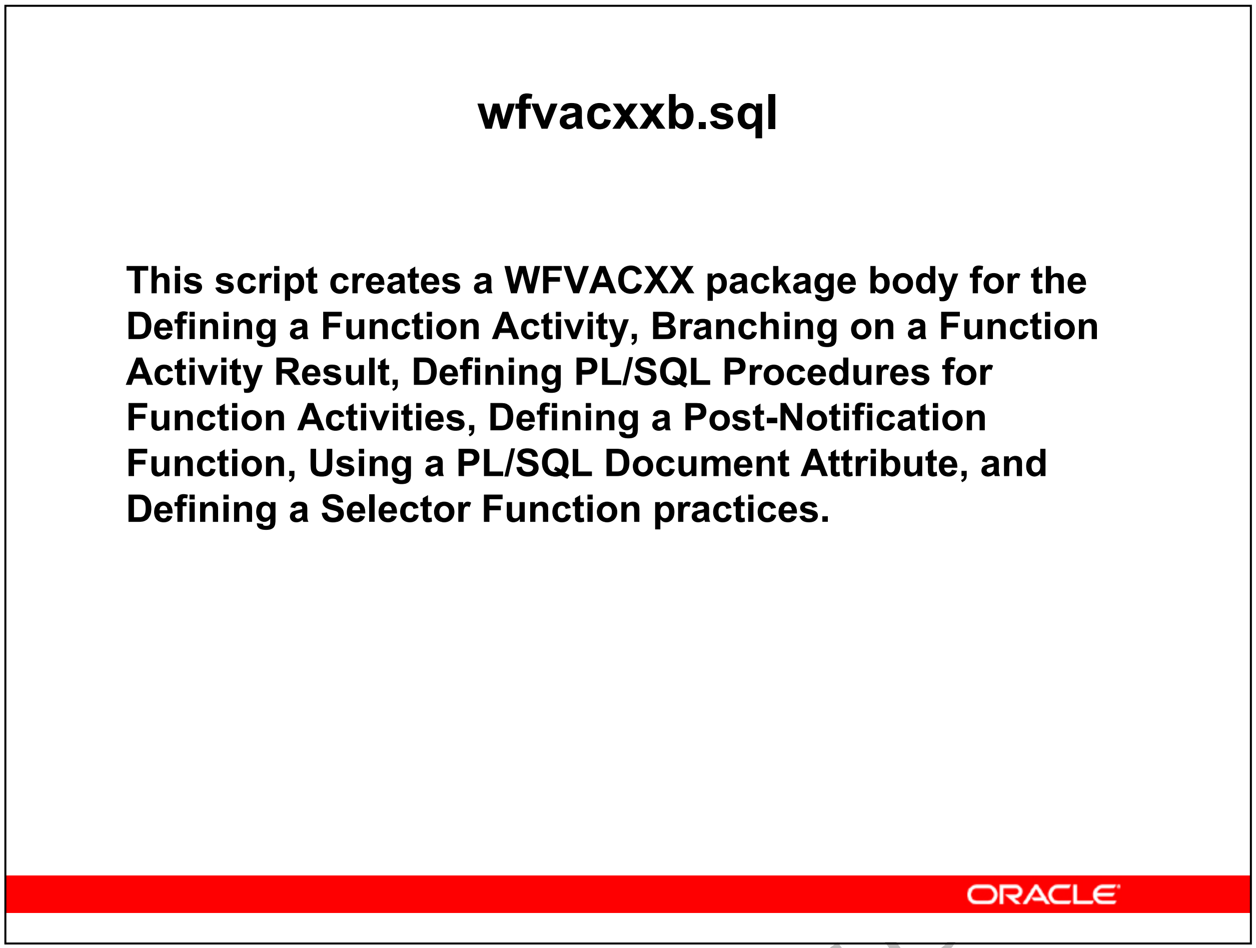
wfvacxxs.sql

This script creates a WFVACXX package specification for the Defining a Function Activity, Branching on a Function Activity Result, Defining PL/SQL Procedures for Function Activities, Defining a Post-Notification Function, Using a PL/SQL Document Attribute, and Defining a Selector Function practices.

wfvacxxs.sql

```
/*=========================================================================+
 |  Copyright (c) 1995 Oracle Corporation Redwood Shores, California, USA|
 |                           All rights reserved.                        |
 +=========================================================================+
 | FILENAME
 |   wfvacxxs.sql
 |
 | DESCRIPTION
 |   CLASS SAMPLE PL/SQL spec for package WFVACXX
 |
 | NOTES
 |   This file is a SAMPLE that should be modified with your own
 |   package and procedure names before installing.
 |   The script is written so that you can simply replace XX with the
 |   unique number assigned to your station.
 |
 |   It may be convenient to use the following naming standard
 |       - package name is equivalent to the item type internal name
 |       - procedure names are equivalent to the workflow activity
 |         internal name that the procedure implements.
 *======================================================================*/

whenever sqlerror exit failure rollback;

create or replace package WFVACXX as
/* $Header$ */

-- PROCEDURE SCHEDULE_UPDATE
--
-- <describe the activity here>
--
-- IN
--   itemtype  - type of the current item
--   itemkey   - key of the current item
--   actid     - process activity instance id
--   funcmode  - function execution mode ('RUN', 'CANCEL', 'TIMEOUT', ...)
-- OUT
--   resultout
--       - COMPLETE[:<result>]
--           activity has completed with the indicated result
--       - WAITING
--           activity is waiting for additional transitions
--       - DEFERED
--           execution should be defered to background
--       - NOTIFIED[:<notification_id>:<assigned_user>]
--           activity has notified an external entity that this
--           step must be performed.  A call to wf_engine.CompleteActivity
--           will signal when this step is complete.  Optional
--           return of notification ID and assigned user.
--       - ERROR[:<error_code>]
--           function encountered an error.
procedure SCHEDULE_UPDATE(
     itemtype  in varchar2,
     itemkey   in varchar2,
     actid     in number,
     funcmode  in varchar2,
     resultout in out varchar2);

-- PROCEDURE NTF_VACATION_PROPOSAL
--
-- <describe the activity here>
--
-- IN
--   itemtype  - type of the current item
--   itemkey   - key of the current item
--   actid     - process activity instance id
--   funcmode  - post-notification function execution mode
--    ('RESPOND','FORWARD','TRANSFER','RUN', 'CANCEL', 'TIMEOUT')
-- OUT
--    resultout
--        - COMPLETE[:<result>]
--            activity has completed with the indicated result
--        - WAITING
--            activity is waiting for additional transitions
--        - DEFERED
--            execution should be defered to background
--        - NOTIFIED[:<notification_id>:<assigned_user>]
--            activity has notified an external entity that this
--            step must be performed.  A call to wf_engine.CompleteActivty
--            will signal when this step is complete.  Optional
--            return of notification ID and assigned user.
--        - ERROR[:<error_code>]
--            function encountered an error.
procedure NTF_VACATION_PROPOSAL(
     itemtype  in varchar2,
     itemkey   in varchar2,
     actid     in number,
     funcmode  in varchar2,
     resultout    in out varchar2);

-- PROCEDURE VACATION_SCHEDULED
--
-- Report vacation scheduled for the current Vacation Proposal requestor
--
-- IN
--    document_id  - string that uniquely identifies the document
--    display_type - text/html or text/plain
-- OUT
--    document     - outbound text buffer
--    document_type- outbound document type of text/html, text/plain, or ''

procedure VACATION_SCHEDULED(
     document_id   in varchar2,
     display_type  in varchar2,
     document      in out varchar2,
     document_type in out varchar2);

-- PROCEDURE CHECK_APPROVER
--
-- Compares the value of the Vacation Proposal Requestor to the Approver.
-- If the Approver = Requestor, return result Y (Yes)
-- If the Approver <>Requestor, return result N (No)
--
-- IN
--    itemtype  - type of the current item
--    itemkey   - key of the current item
--    actid     - process activity instance id
--   funcmode  - function execution mode ('RUN', 'CANCEL', 'TIMEOUT', ...)
-- OUT
--   resultout
--       - COMPLETE[:<result>]
--           activity has completed with the indicated result
--       - WAITING
--           activity is waiting for additional transitions
--       - DEFERED
--           execution should be defered to background
--       - NOTIFIED[:<notification_id>:<assigned_user>]
--           activity has notified an external entity that this
--           step must be performed.  A call to wf_engine.CompleteActivity
--           will signal when this step is complete.  Optional
--           return of notification ID and assigned user.
--       - ERROR[:<error_code>]
--           function encountered an error.
procedure CHECK_APPROVER(
     itemtype  in varchar2,
     itemkey   in varchar2,
     actid     in number,
     funcmode  in varchar2,
     resultout in out varchar2);

-- PROCEDURE SELECTOR
--
-- Examines the value of item attribute TIMEOUT_CHOICE to select
-- which process to run
-- If TIMEOUT_CHOICE = LOOP COUNTER, then run WFVACXX_PROCESS
-- If TIMEOUT_CHOICE = SELF LOOP, then run WFVACXX_ALTERNATE_PROCESS
--
-- IN
--   itemtype  - type of the current item
--   itemkey   - key of the current item
--   actid     - process activity instance id
--   command   - execution mode ('RUN', 'SET_CTX','TEST_CTX',...)
-- OUT
--   resultout
--       - name of process to run
--       - ERROR[:<error_code>]
--           function encountered an error.
procedure SELECTOR(
     itemtype  in varchar2,
     itemkey   in varchar2,
     actid     in number,
     command   in varchar2,
     resultout in out varchar2);

end WFVACXX;
/
```

```
show errors package WFVACXX
commit;
--exit;
```



wfvacxxb.sql

This script creates a WFVACXX package body for the Defining a Function Activity, Branching on a Function Activity Result, Defining PL/SQL Procedures for Function Activities, Defining a Post-Notification Function, Using a PL/SQL Document Attribute, and Defining a Selector Function practices.

wfvacxxb.sql

```
/*=========================================================================+
 |  Copyright (c) 1995 Oracle Corporation Redwood Shores, California, USA|
 |                          All rights reserved.                          |
 +=========================================================================+
 | FILENAME
 |   wfvacxxb.sql
 |
 | DESCRIPTION
 |   CLASS SAMPLE PL/SQL spec for package WFVACXX
 |
 | NOTES
 |   This file is a SAMPLE that should be modified with your own
 |   names and procedures before installing.
 |   The script is written so that you can simply replace XX with the
 |   unique number assigned to your station.
 |
 |   It may be convenient to use the following naming standard
 |       - package name is equivalent to the item type internal name
 |       - procedure names are equivalent to the workflow activity
 |         internal name that the procedure implements.
 *=======================================================================*/

whenever sqlerror exit failure rollback;

create or replace package body WFVACXX as
/* $Header$ */

-- PROCEDURE SCHEDULE_UPDATE
--
-- Insert a row into an employee based vacation schedule table
--
-- IN
--   itemtype  - type of the current item
--   itemkey   - key of the current item
--   actid     - process activity instance id
--   funcmode  - function execution mode. this is set by the engine
--               as either 'RUN', 'CANCEL', 'TIMEOUT'
-- OUT
--   resultout
--       - COMPLETE[:<result>]
--           activity has completed with the indicated result
--       - WAITING
--           activity is waiting for additional transitions
--       - DEFERED
--           execution should be defered to background
--       - NOTIFIED[:<notification_id>:<assigned_user>]
--           activity has notified an external entity that this
--           step must be performed.  A call to wf_engine.CompleteActivity
--           will signal when this step is complete.  Optional
--           return of notification ID and assigned user.
--       - ERROR[:<error_code>]
--           function encountered an error.
procedure SCHEDULE_UPDATE(
     itemtype  in varchar2,
     itemkey   in varchar2,
     actid     in number,
     funcmode  in varchar2,
     resultout in out varchar2)
is
     lrequestor_username    varchar2(30);
     lapprover_username     varchar2(30);
     lfrom_date             date;
     lto_date               date;
begin

  --
  -- RUN mode - normal process execution
  --
```

```
if (funcmode = 'RUN') then

  -- retrieve requestor, approver, from, and to dates
  lrequestor_username := wf_engine.GetItemAttrText(itemtype => itemtype,
                                  itemkey => itemkey,
                                  aname => 'REQUESTOR');
  lapprover_username := wf_engine.GetItemAttrText(itemtype => itemtype,
                                  itemkey => itemkey,
                                  aname => 'APPROVER');
  lfrom_date := wf_engine.GetItemAttrDate(itemtype => itemtype,
                                  itemkey => itemkey,
                                  aname => 'FROM_DATE');
  lto_date := wf_engine.GetItemAttrDate(itemtype => itemtype,
                                  itemkey => itemkey,
                                  aname => 'TO_DATE');


  -- insert row into vacation schedule table
  insert into WFVACXX_VACATION_SCHEDULE
         (REQUESTOR_USERNAME,
          APPROVER_USERNAME,
          FROM_DATE,
          TO_DATE)
  values (lrequestor_username,
          lapprover_username,
          lfrom_date,
          lto_date);

  -- no result needed
  resultout  := wf_engine.eng_completed||':'||wf_engine.eng_null;
  return;
end if;


--
-- CANCEL mode - activity 'compensation'
--
-- This is in the event that the activity must be undone,
-- for example when a process is reset to an earlier point
-- due to a loop back.
--
if (funcmode = 'CANCEL') then

  -- retrieve requestor, approver, from, and to dates
  lrequestor_username := wf_engine.GetItemAttrText(itemtype => itemtype,
                                  itemkey => itemkey,
                                  aname => 'REQUESTOR');
  lapprover_username := wf_engine.GetItemAttrText(itemtype => itemtype,
                                  itemkey => itemkey,
                                  aname => 'APPROVER');
```

```
        lfrom_date := wf_engine.GetItemAttrDate(itemtype => itemtype,
                                         itemkey => itemkey,
                                         aname => 'FROM_DATE');
        lto_date := wf_engine.GetItemAttrDate(itemtype => itemtype,
                                         itemkey => itemkey,
                                         aname => 'TO_DATE');


        -- delete row from vacation schedule table
        delete from WFVACXX_VACATION_SCHEDULE
        where  REQUESTOR_USERNAME = lrequestor_username
          and  APPROVER_USERNAME = lapprover_username
          and  FROM_DATE = lfrom_date
          and  TO_DATE = lto_date;

        -- no result needed
        resultout := wf_engine.eng_completed||':'||wf_engine.eng_null;
        return;
      end if;


      --
      -- Other execution modes may be created in the future.  Your
      -- activity will indicate that it does not implement a mode
      -- by returning null
      --
      resultout := wf_engine.eng_null;
      return;

    exception
      when others then
        -- The line below records this function call in the error system
        -- in the case of an exception.
        wf_core.context('WFVACXX', 'SCHEDULE_UPDATE',
                        itemtype, itemkey, to_char(actid), funcmode);
        raise;
    end SCHEDULE_UPDATE;

    -- PROCEDURE NTF_VACATION_PROPOSAL
    --
    -- <describe the activity here>
    --
    -- IN
    --   itemtype  - type of the current item
    --   itemkey   - key of the current item
    --   actid     - process activity instance id
    --   funcmode  - post-notification function execution mode
    --    ('RESPOND','FORWARD','TRANSFER','RUN', 'CANCEL', 'TIMEOUT')
    -- OUT
    --   resultout
```

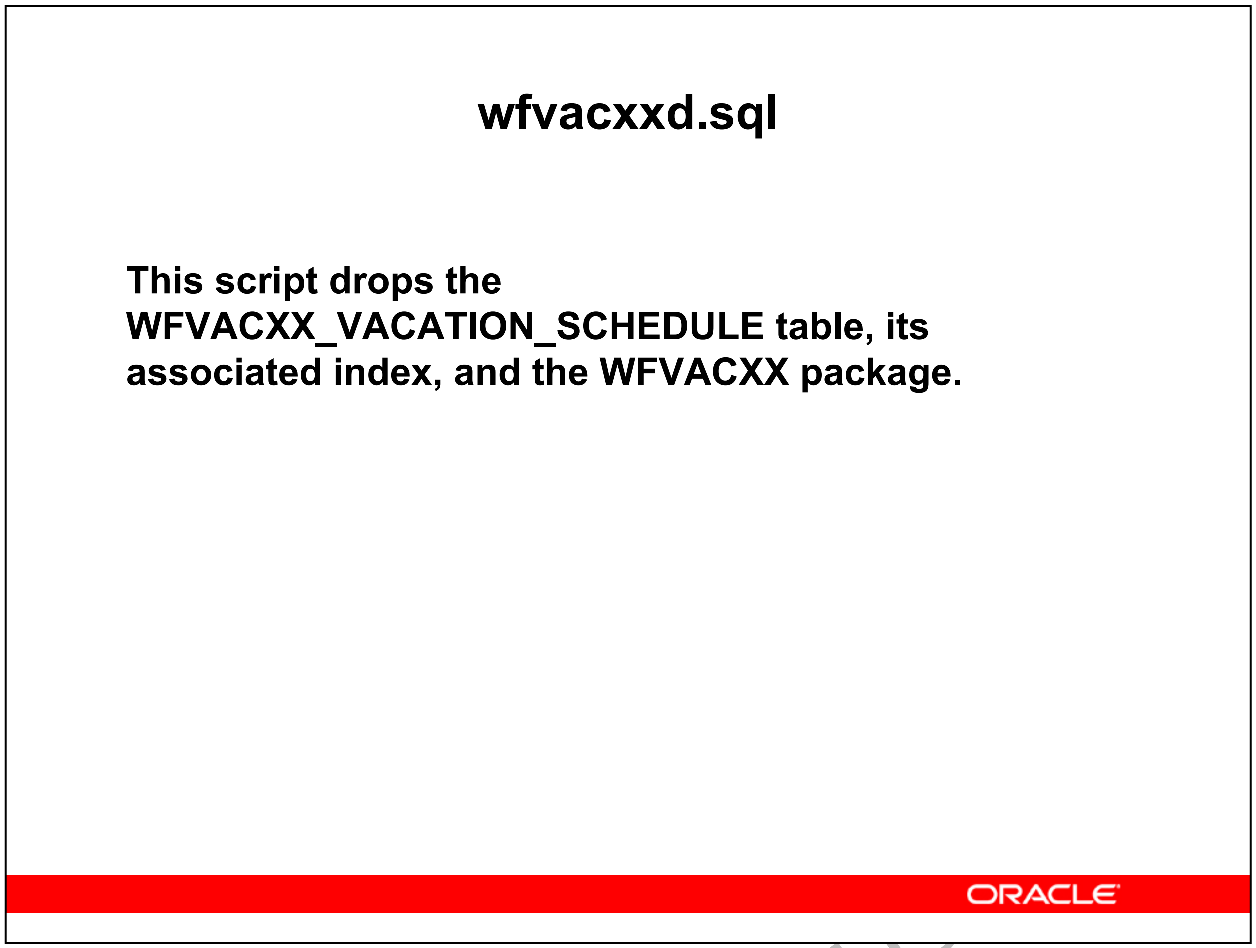
```
--       - COMPLETE[:<result>]
--           activity has completed with the indicated result
--       - WAITING
--           activity is waiting for additional transitions
--       - DEFERED
--           execution should be defered to background
--       - NOTIFIED[:<notification_id>:<assigned_user>]
--           activity has notified an external entity that this
--           step must be performed.  A call to wf_engine.CompleteActivity
--           will signal when this step is complete.  Optional
--           return of notification ID and assigned user.
--       - ERROR[:<error_code>]
--           function encountered an error.
procedure NTF_VACATION_PROPOSAL(
     itemtype  in varchar2,
     itemkey   in varchar2,
     actid     in number,
     funcmode  in varchar2,
     resultout in out varchar2)
is
     nid                number;
     ntf_responder      varchar2(30);
     ntf_result         varchar2(30);
     ntf_alt_from_date date := '';
     ntf_alt_to_date   date := '';

begin

  --
  -- RESPOND mode - recipient has supplied a response to the notification
  --
  if (funcmode = 'RESPOND') then

      -- get notification id and responder from wf_engine context variables
      nid := WF_ENGINE.CONTEXT_NID;
      ntf_responder := WF_ENGINE.CONTEXT_TEXT;

    -- if the approver rejects the vacation proposal then he/she must
    -- provide an alternate date window

    -- retrieve the notification result
    ntf_result := wf_notification.GetAttrText(nid,'RESULT');
    if (ntf_result = 'REJECTED') then

      -- retrieve the alternate vacation dates
      ntf_alt_from_date :=
      wf_notification.GetAttrDate(nid,'ALT_FROM_DATE');

      ntf_alt_to_date   :=
      wf_notification.GetAttrDate(nid,'ALT_TO_DATE');
      if (ntf_alt_from_date is null
        or ntf_alt_to_date is null) then
           -- raise an error
           resultout  := wf_engine.eng_error||':'||wf_engine.eng_null;
           wf_core.Raise('Provide Alternate Dates');
           return;
      end if;
      if (ntf_alt_from_date > ntf_alt_to_date) then
          -- raise an error
          resultout  := wf_engine.eng_error||':'||wf_engine.eng_null;
          wf_core.Raise('From Date before To Date');
          return;
      end if;
    end if;
    resultout  := wf_engine.eng_completed||':'||ntf_result;
    return;
  end if;

  --
  -- TRANSFER mode - recipient attempting to Transfer notification
  --
  if (funcmode = 'TRANSFER') then

    -- don't allow transfer
    -- raise an error
    resultout  := wf_engine.eng_error||':'||wf_engine.eng_null;
    wf_core.Raise('Transfer not allowed');
    return;
  end if;

  --
  -- FORWARD mode - recipient attempting to Delegate notification
  --
  if (funcmode = 'FORWARD') then

    -- delegate allowed
    null;
    -- set resultout to null to indicate that the mode is not implemented
    resultout := wf_engine.eng_null;
    return;
    end if;

  --
  -- RUN mode - in post-notification function, response to notification
  --            already processed and accepted
  --
  if (funcmode = 'RUN') then
    -- if implemented, your run code goes here
    null;
```

```
      -- set resultout to null to indicate that the mode is not implemented
      resultout := wf_engine.eng_null;
      return;
    end if;

    --
    -- TIMEOUT mode - recipient has allowed the notification to timeout
    --
    if (funcmode = 'TIMEOUT') then
      -- if implemented, your timeout code goes here
      null;
      -- set resultout to null to indicate that the mode is not implemented
      resultout := wf_engine.eng_null;
      return;
    end if;



    --
    -- CANCEL mode - activity 'compensation'
    --
    -- This is in the event that the activity must be undone,
    -- for example when a process is reset to an earlier point
    -- due to a loop back.
    --
    if (funcmode = 'CANCEL') then
      -- if implemented, your cancel code goes here
      null;
      -- set resultout to null to indicate that the mode is not implemented
      resultout := wf_engine.eng_null;
      return;
    end if;



    --
    -- Other execution modes may be created in the future.  Your
    -- activity will indicate that it does not implement a mode
    -- by returning null
    --
    resultout := wf_engine.eng_null;
    return;

  exception
    when others then
      -- The line below records this function call in the error system
      -- in the case of an exception.
      wf_core.context('WFVACXX', 'NTF_VACATION_PROPOSAL',
                      itemtype, itemkey, to_char(actid), funcmode);
      raise;
  end NTF_VACATION_PROPOSAL;
```

```
-- PROCEDURE VACATION_SCHEDULED
--
-- Report vacation scheduled for the current Vacation Proposal requestor
--
-- IN
--   document_id  - string that uniquely identifies the document
--   display_type - text/html or text/plain
-- OUT
--   document     - outbound text buffer
--   document_type- outbound document type of text/html, text/plain, or ''

procedure VACATION_SCHEDULED(
     document_id   in varchar2,
     display_type  in varchar2,
     document      in out varchar2,
     document_type in out varchar2) is

  cursor vacation_schedule (xrequestor in varchar2) is
     select approver_username,
            to_char(from_date,'Day DD Month YYYY') from_displayed,
            to_char(to_date,'Day DD Month YYYY') to_displayed
     from   wfvacxx_vacation_schedule
     where  requestor_username in
            (select name from wf_users
             where display_name = xrequestor)
     order by from_date, to_date;

begin
     if display_type = 'text/html' then
       document_type := 'text/html';
       document :=
         '<BR><BR><LEFT><TABLE BORDER CELLPADDING=5 BGCOLOR=#FFFFFF>'||
         '<TR BGCOLOR=#83C1C1>'||
         '<TH>From Date</TH>'||
         '<TH>To Date</TH>'||
         '<TH>Approver</TH>'||
         '</TR>';
     else
       document_type := 'text/plain';
       document :=
         chr(10)||rpad('From Date',28)||
         rpad('To Date',28)||
         rpad('Approver',30)||
         chr(10);
     end if;

     -- build table body with data
     for schedule_rec in vacation_schedule(document_id) loop
       if display_type = 'text/html' then
         document := document||
           '<TR>'||
           '<TD>'||schedule_rec.from_displayed||'</TD>'||
           '<TD>'||schedule_rec.to_displayed||'</TD>'||
           '<TD>'||
           wf_directory.getroledisplayname(schedule_rec.approver_username)
           ||'</TD>'||
           '</TR>';
       else
         document := document||
           rpad(schedule_rec.from_displayed,28)||
           rpad(schedule_rec.to_displayed,28)||
           rpad(
           wf_directory.getroledisplayname(schedule_rec.approver_username)
           ,30)||
           chr(10);
       end if;
     end loop;

     -- close the table
     if display_type = 'text/html' then
       document := document||
         '</TABLE></LEFT><BR>';
     end if;

  return;

exception
  when others then
    -- The line below records this procedure call in the error system
    -- in the case of an exception.
    wf_core.context('WFVACXX', 'VACATION_SCHEDULED',
                    document_id, display_type);
    raise;
end VACATION_SCHEDULED;

-- PROCEDURE CHECK_APPROVER
--
-- Compares the value of the Vacation Proposal Requestor to the Approver.
-- If the Approver = Requestor, return result Y (Yes)
-- If the Approver <>Requestor, return result N (No)
--
-- IN
--   itemtype  - type of the current item
--   itemkey   - key of the current item
--   actid     - process activity instance id
--   funcmode  - function execution mode ('RUN', 'CANCEL', 'TIMEOUT', ...)
-- OUT
--   resultout
--       - COMPLETE[:<result>]
--           activity has completed with the indicated result
```

```
--       - WAITING
--           activity is waiting for additional transitions
--       - DEFERED
--           execution should be defered to background
--       - NOTIFIED[:<notification_id>:<assigned_user>]
--           activity has notified an external entity that this
--           step must be performed.  A call to wf_engine.CompleteActivity
--           will signal when this step is complete.  Optional
--           return of notification ID and assigned user.
--       - ERROR[:<error_code>]
--           function encountered an error.
procedure CHECK_APPROVER(
     itemtype  in varchar2,
     itemkey   in varchar2,
     actid     in number,
     funcmode  in varchar2,
     resultout in out varchar2)
is
     lrequestor_username varchar2(30);
     lapprover_username  varchar2(30);
     wf_yes              varchar2(1) := 'Y';
     wf_no               varchar2(1) := 'N';
begin

  --
  -- RUN mode - normal process execution
  --
  if (funcmode = 'RUN') then

    -- retrieve requestor, approver
    lrequestor_username := wf_engine.GetItemAttrText(itemtype => itemtype,
                                 itemkey => itemkey,
                                 aname => 'REQUESTOR');
    lapprover_username := wf_engine.GetItemAttrText(itemtype => itemtype,
                                 itemkey => itemkey,
                                 aname => 'APPROVER');

    if lrequestor_username <> lapprover_username then
      resultout  := wf_engine.eng_completed||':'||wf_no;
    else
      resultout  := wf_engine.eng_completed||':'||wf_yes;
    end if;
    return;
  end if;


  --
  -- CANCEL mode - activity 'compensation'
  --
  -- This is in the event that the activity must be undone,
  -- for example when a process is reset to an earlier point
  -- due to a loop back.
  --
  if (funcmode = 'CANCEL') then

    -- no result needed
    resultout := wf_engine.eng_completed||':'||wf_engine.eng_null;
    return;
  end if;



  --
  -- Other execution modes may be created in the future.  Your
  -- activity will indicate that it does not implement a mode
  -- by returning null
  --
  resultout := wf_engine.eng_null;
  return;

exception
  when others then
    -- The line below records this function call in the error system
    -- in the case of an exception.
    wf_core.context('WFVACXX', 'CHECK_APPROVER',
                    itemtype, itemkey, to_char(actid), funcmode);
    raise;
end CHECK_APPROVER;

-- PROCEDURE SELECTOR
--
-- Examines the value of item key to select
-- which process to run.
-- If the first four characters of itemkey:
--   is 'CNTR', then run WFVACXX_PROCESS
--   is 'SELF', then run WFVACXX_ALTERNATE_PROCESS
-- Note:  This logic is contrived for class practice only.
--        The selector function is expected to use the itemkey as
--        the primary key to retrieve supporting application data.
--        The Application data retrieved would be used to determine
--        which process is appropriate to start.
--
-- IN
--   itemtype  - type of the current item
--   itemkey   - key of the current item
--   actid     - process activity instance id
--   command   - execution mode ('RUN', 'SET_CTX','TEST_CTX',...)
-- OUT
--   resultout
--       - name of process to run
--       - ERROR[:<error_code>]
```

```
--           function encountered an error.
procedure SELECTOR(
     itemtype  in varchar2,
     itemkey   in varchar2,
     actid     in number,
     command   in varchar2,
     resultout in out varchar2)
is
     litemkey varchar2(30) := itemkey;
begin

  --
  -- RUN mode - determine which process to run
  --
  if (command = 'RUN') then

    if UPPER(substr(litemkey,1,4)) = 'CNTR' then
      resultout  := 'WFVACXX_PROCESS';
    elsif UPPER(substr(litemkey,1,4)) = 'SELF' then
      resultout  := 'WFVACXX_ALTERNATE_PROCESS';
    else
      resultout  := wf_engine.eng_error||':'||wf_engine.eng_null;
      wf_core.Raise('Invalid item key');
    end if;
    return;
  end if;


  --
  -- SET_CTX mode
  --
  if (command = 'SET_CTX') then

    -- no result needed
    resultout := wf_engine.eng_null;
    return;
  end if;

  if (command = 'TEXT_CTX') then

    -- no result needed
    resultout := wf_engine.eng_null;
    return;
  end if;

  --
  -- Other execution modes may be created in the future.  Your
  -- selector will indicate that it does not implement a mode
  -- by returning null
  --
  resultout := wf_engine.eng_null;
  return;

exception
  when others then
    -- The line below records this function call in the error system
    -- in the case of an exception.
    wf_core.context('WFVACXX', 'SELECTOR',
                    itemtype, itemkey, to_char(actid), command);
    raise;
end SELECTOR;

end WFVACXX;
/

show errors package body WFVACXX
commit;
--exit;
```



wfvacxxd.sql

This script drops the WFVACXX_VACATION_SCHEDULE table, its associated index, and the WFVACXX package.

wfvacxxd.sql

```
/*=======================================================================+
 |  Copyright (c) 1995 Oracle Corporation Redwood Shores, California, USA|
 |                            All rights reserved.                       |
 +=======================================================================+
 | FILENAME
 |   wfvacxxd.sql
 |
 | DESCRIPTION
 |   Drop Workflow Vacation Schedule index, table and package.
 | NOTES
 |   This file is a SAMPLE that should be modified with your own
 |   names before installing.  Names that include
 |   XX should be replaced with values for your implementation.
 |
 *=======================================================================*/

/* $Header$ */
```

```
whenever sqlerror continue;
--
drop   index WFVACXX_VACATION_SCHEDULE_N1;
--
drop   table WFVACXX_VACATION_SCHEDULE;
--
drop   package WFVACXX;

commit;

--exit
```