PEOPLESOFT ENTERPRISE

Portal Solutions 9.1 PeopleBook: Events and Notifications

September 2009

Portal Solutions 9.1 PeopleBook: Events and Notifications
SKU ps91pbr0

Contents

Events and Notifications Framework Preface

This preface provides an overview of the Enterprise Components: PeopleSoft Events and Notifications Framework.

Events and Notifications Framework

The *Events and Notifications Framework PeopleBook* provides you with implementation and processing information for the PeopleSoft Events and Notifications feature. However, additional, essential information describing the setup and design of your system resides in companion documentation. The companion documentation consists of important topics that apply to many or all Oracle's PeopleSoft applications across the PeopleSoft product lines. You should be familiar with the contents of the PeopleBooks that apply to your organization.

Pages With Deferred Processing

Several pages in PeopleSoft Inventory operate in deferred processing mode. Most fields on these pages are not updated or validated until you save the page or refresh it when you click a button, link, or tab. This delayed processing has various implications for the field values on the page; for example, if a field contains a default value, any value you enter before the system updates the page overrides the default. Another implication is that the system updates quantity balances or totals only when you save or otherwise refresh the page.

PeopleBooks and the Online PeopleSoft Library

A companion PeopleBook called PeopleBooks and the Online PeopleSoft Library contains general information, including:

- Understanding the PeopleSoft online library and related documentation.
- How to send PeopleSoft documentation comments and suggestions to Oracle.
- How to access hosted PeopleBooks, downloadable HTML PeopleBooks, and downloadable PDF PeopleBooks as well as documentation updates.
- Understanding PeopleBook structure.
- Typographical conventions and visual cues used in PeopleBooks.
- ISO country codes and currency codes.
- PeopleBooks that are common across multiple applications.

- Common elements used in PeopleBooks.
- Navigating the PeopleBooks interface and searching the PeopleSoft online library.
- Displaying and printing screen shots and graphics in PeopleBooks.
- How to manage the PeopleSoft online library including full-text searching and configuring a reverse proxy server.
- Understanding documentation integration and how to integrate customized documentation into the library.
- Glossary of useful PeopleSoft terms that are used in PeopleBooks.

You can find this companion PeopleBook in your PeopleSoft online library.

Common Elements Used in This PeopleBook

Business Unit	An identification code that represents a high-level organization of business information. You can use a business unit to define regional or departmental units within a larger organization.
Description	Free-flow text up to 30 characters.
Effective Date	Date on which a table row becomes effective; the date that an action begins. For example, if you want to close out a ledger on June 30, the effective date for the ledger closing would be July 1. This date also determines when you can view and change the information. Pages or panels and background processes that use the information use the current row.
Language or Language Code	The language in which you want the field labels and report headings of the reports to print. The field values appear as you enter them. Language also refers to the language spoken by an employee, applicant, or nonemployee.
Process Frequency (group box)	Designates the appropriate frequency in the Process Frequency group box: Process Once executes the request the next time the batch process runs. After the batch process runs, the process frequency is automatically set to Don't Run. Always Process executes the request every time the batch process runs. Don't Run ignores the request when the batch process runs.
Process Monitor	This link takes you to the Process List page, where you can view the status of submitted process requests.
Report ID	The report identifier.
Report Manager	This link takes you to the Report List page, where you can view report content, check the status of a report, and see content detail messages (which show you a description of the report and the distribution list).

Request ID	A request identification that represents a set of selection criteria for a report or process.
Run	This button takes you to the Process Scheduler request page, where you can specify the location where a process or job runs and the process output format.
SetID	An identification code that represents a set of control table information or TableSets. A TableSet is a group of tables (records) necessary to define your organization's structure and processing options.
Short Description	Free-flow text up to 15 characters.
Status	the options in this field are *Active* or *Inactive*. By linking status and effective date, you can retain historical information and plan future implementation. For auditing purposes, PeopleSoft encourages inactivating data that is no longer in use instead of deleting it.
User ID	The system identifier for the individual who generates a transaction

Chapter 1

Getting Started with PeopleSoft Events and Notifications Framework

This chapter provides an overview of the PeopleSoft Events and Notifications Framework and discusses implementation information.

PeopleSoft Events and Notifications Framework Overview

The PeopleSoft Events and Notifications framework provides three features that can be configured and used to monitor business processes and create messages when unusual situations or errors occur within a PeopleSoft process or table. These messages can be routed to different users (PeopleSoft or non-PeopleSoft) to prompt the user to resolve the issue. Links can take the user directly to the page for correction or resolution. The three features are:

- Events
- Notifications
- Alerts

Events

Events is a feature and framework that enables you to define, implement, and run business logic for business events.

Using the Events framework, you can define the business events that the system raises when you change certain data in application components or run certain PeopleTools Application Engine processes. You can also define the event handlers that the Events framework executes to react to these specific event instances. *Raising* an event is the act of creating an event instance. If an event is raised, the Events framework automatically executes the business logic for the registered event handlers of the event.

Events allow you to:

- Isolate, automatically process, and monitor the business logic asynchronously.
- Define your own business events and build your own event handlers for these events with minimal impact to the delivered code.
- Troubleshoot based on detail provided in the Event Monitor.

Notifications and Alerts

The notifications feature enables you to monitor the transaction flow and alert your organization to any errors, changes, and stalled transactions.

With notifications and alerts, you can:

- Monitor PeopleSoft processes and receive notifications when exceptions are found.
- Scan PeopleSoft tables using PeopleSoft Query and receive notifications when exceptions are found.
- Use the Notification Dashboard to monitor messages and link to the PeopleSoft page where you can review or correct the issue.
- Receive notifications to your email account.
- Receive notifications to your PeopleSoft worklist.
- Pass messages to a third-party system using the XML format.
- Create a custom action for a notification using an PeopleSoft application class.

Events and Notifications Framework Implementation

Many PeopleSoft applications are delivered with pre-defined events and notifications. Refer to your specific application for implementation details.

PeopleSoft Setup Manager enables you to review a list of setup tasks for your organization for the products that you are implementing. The setup tasks include the components that you must set up, listed in the order in which you must enter data into the component tables, and links to the corresponding PeopleBook documentation.

Other Sources of Information

In the planning phase of your implementation, take advantage of all PeopleSoft sources of information, including the installation guides, table-loading sequences, data models, and business process maps.

Implementing Events

To implement Event Manager into your application processing:

Step	***Reference***
Identify the business events that drive the execution of your business logic and, if necessary, define a new event.	See Chapter 4, "Setting Up Events," Defining and Registering Events, page 26.
Write the event handlers in PeopleTools Application Designer to execute the business logic that is specific to the event.	See Chapter 4, "Setting Up Events," Creating Event Handlers, page 30.

Step	***Reference***
Register the event handlers to the event through the Event Registry component.	See Chapter 4, "Setting Up Events," Defining and Registering Events, page 26.
Test your event and its registered event handlers.	See Chapter 4, "Setting Up Events," Testing an Event Handler, page 38.

Implementing Notifications

To implement notifications:

Step	***Reference***
Add your process name and category to the Notification Registry page.	See Chapter 3, "Setting Up Notifications," Setting Up the Notification Registry, page 13.
Create a message context record to pass all transactional data through the message framework.	All context records must include the EOEN_LOG_KEY subrecord. Any additional transactional fields can be added to the context record.
Implement the EOEN interface object within your application process by using the EOEN_MVC:EOEN_MODEL.EOEN_INTERFACE class.	See Chapter 3, "Setting Up Notifications," Modifying PeopleSoft Processes to Create Notifications, page 19.

Implementing Alerts

Use these steps to implement alerts:

Step	***Reference***
Define the query to use with the alert.	See Chapter 5, "Creating Alerts," Defining PeopleSoft Queries in the Alert Setup Component, page 56.
Define the Alert Setup	See Chapter 5, "Creating Alerts," Defining PeopleSoft Queries in the Alert Setup Component, page 56.
Run the Alert process.	See Chapter 5, "Creating Alerts," Running the Alerts Process, page 61.

Chapter 2

Understanding the Events and Notifications Framework

This chapter provides an overview of the Events and Notification framework, including:

- Process flow of notification messages generated by a PeopleSoft process.
- Process flow of alert messages generated by a PeopleSoft Query.

Understanding the Process Flow of the Events and Notifications Framework

As your organization processes the continuous stream of transactions within your PeopleSoft system, the events and notifications feature enables you to monitor the transaction flow and alert your organization to any errors, changes, and stalled transactions. The messages generated by the events and notifications feature can notify you of the problem, give you a detailed description of the issue, and provide a link to the PeopleSoft page where you can resolve it.

The events and notifications delivers a flexible way for you to route these messages to different individuals, job roles, or departments to insure that the correct person handles any potential problems before they cause trouble. Based on your setup of the events and notifications feature, you can view these messages:

- On the Notification Dashboard page. This dashboard provides a central location to view all of these messages.
- (optional) Within a PeopleSoft Worklist.
- (optional) Within an email account.
- (optional) Within a third-party system as an XML-formatted message.
- (optional) Using a custom action created by an application class.

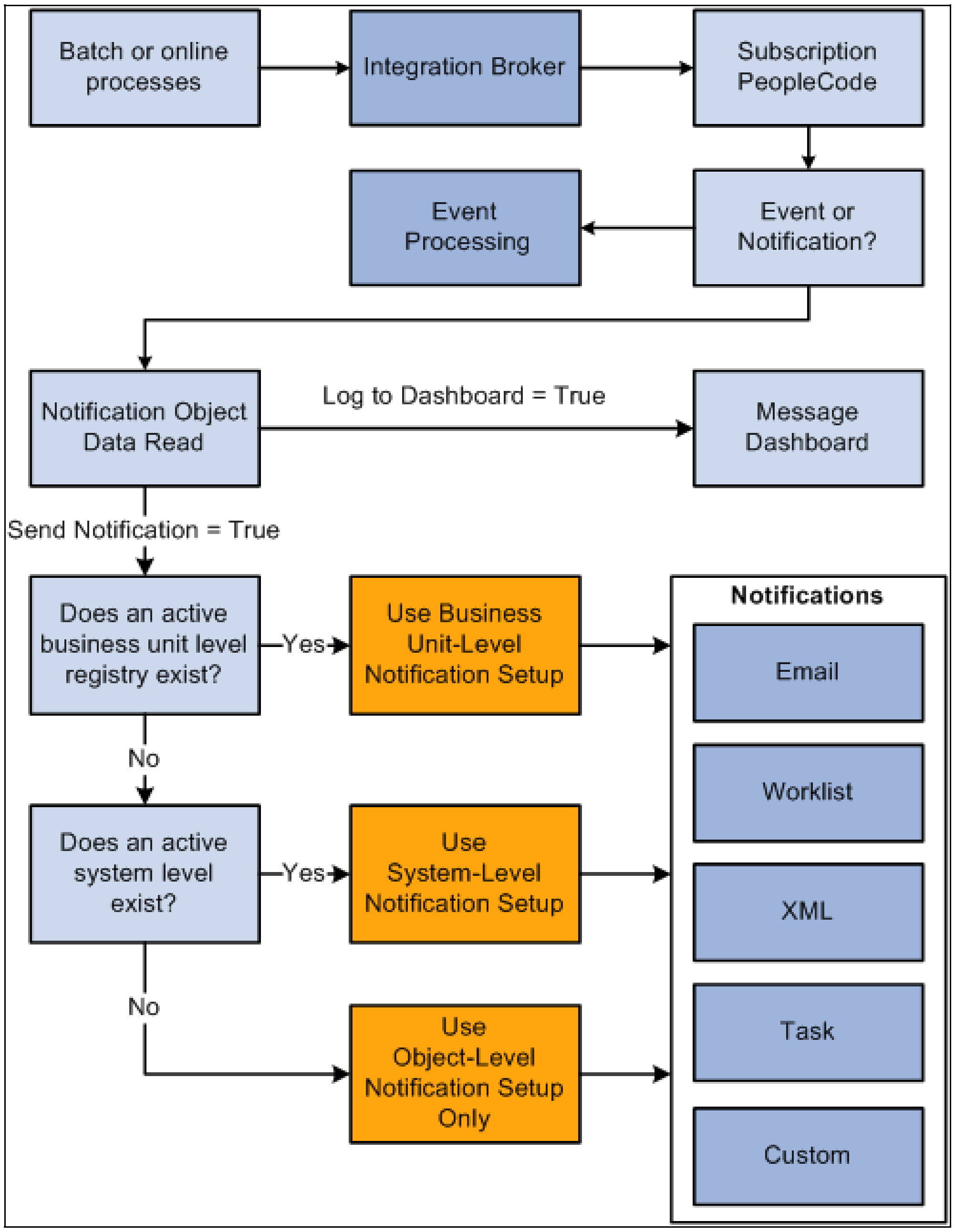
There are three ways to use notifications:

- *Use pre-defined notifications:* Some PeopleSoft applications are delivered with batch and online processes that have been pre-defined to generate notification messages. Once you complete the notification setup steps, the notification messages are delivered to the appropriate user.
- *Define new notifications for a PeopleSoft process:* Use the PeopleSoft Application Designer to alter an existing PeopleSoft process to generate messages. Then complete the notification setup steps for the new notifications. This method alters the PeopleCode of an existing PeopleSoft process and requires the assistance of your IT department.

- *Use PeopleSoft Query:* Create a PeopleSoft Query on one or more PeopleSoft tables, add the Alert using the Alert Setup component, and then complete the Alert setup steps. When the Alerts process is run, the PeopleSoft table is queried and the Alert messages can be generated. This method enables you to monitor data within a PeopleSoft table; for example, searching for stalled transactions, instead of looking at the results of a PeopleSoft process.

The Process Flow of Notification Messages Generated by a PeopleSoft Process

This diagram illustrates the process flow of notifications produced by a batch or online PeopleSoft process. The PeopleSoft process creates message object data that includes values such as, who to notify and where to post the message (Notification Dashboard, Worklist, email, XML notification, or custom action). The message is sent through Integration Broker and subscription PeopleCode determines whether it is an event or a notification. The system checks for override values at the notification registry, the business unit level, or the system level, and then notifies the appropriate person using the Notification Dashboard page, an email, a worklist entry, an XML message, or a custom action.


Batch or online processes
Integration Broker
Subscription PeopleCode
Event Processing
Event or Notification?
Notification Object Data Read
Log to Dashboard = True
Message Dashboard
Send Notification = True
Does an active business unit level registry exist?
Yes
Use Business Unit-Level Notification Setup
No
Does an active system level exist?
Yes
Use System-Level Notification Setup
No
Use Object-Level Notification Setup Only
Notifications
Email
Worklist
XML
Task
Custom

Generating notification messages from a PeopleSoft process

A PeopleSoft batch or online process, that has been coded to use the notification feature, generates message object data when certain conditions are met. For example, the reservations process in PeopleSoft Inventory sends message data when someone uses the Shortage Workbench to unreserve quantity on a material stock request. The PeopleSoft process provides several values to the notification framework, including:

- Where to log the message (any combination of the Notification Dashboard, PeopleSoft Worklist, email, XML notification, and custom action).
- The message set and message number.
- User roles, user IDs, email addresses, or PeopleSoft nodes needed to route the message to the correct location.

- Links to the PeopleSoft page where the transaction can be viewed or resolved.

The Notification Registry page is reviewed for the specific process name and process category to determine if:

- Overrides were turned on at the system-level and business unit level for worklist, email, or XML notifications.
- A custom action was created.
- A pre-processing user exit was created.

The PeopleSoft system then checks to see if override values have been entered at the business unit level using the BU-Level Notifications page. The business unit level values override the values provided by the PeopleSoft process.

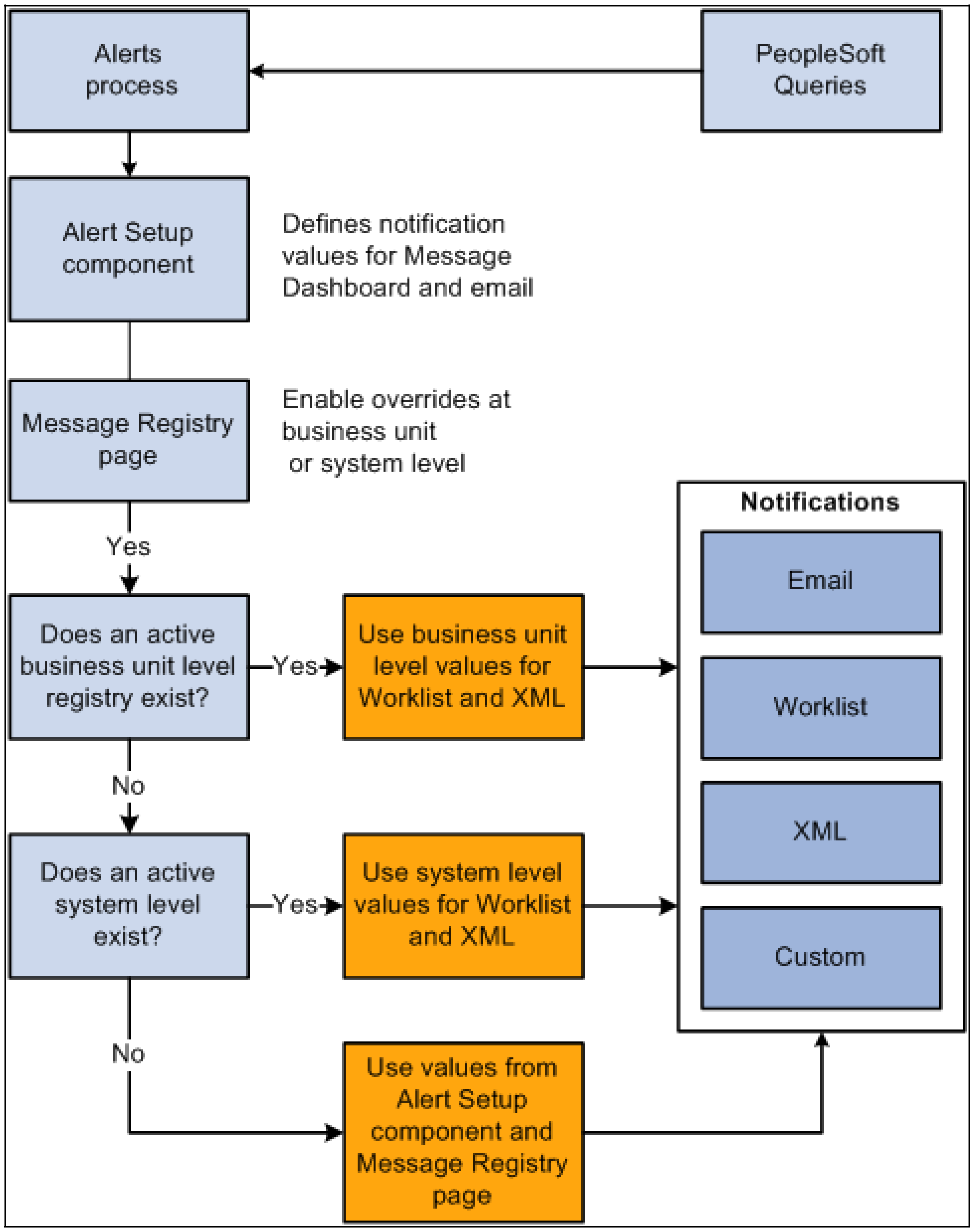
If there are no entries on the BU-Level Notifications page for this process name and process category, then the PeopleSoft system checks to see if the values have been overridden at the system level using the System-Level Notifications page. The system-level values override the values provided by the PeopleSoft process.

The messages are posted to the Notification Dashboard and other locations based on the values used. If the notification is sent using an email or a worklist entry, a link is provided to take the individual to the message detail page of the Notification Dashboard.

Note. Integration Broker must be configured and active for the Notification and Events framework to function.

The Process Flow of Alert Messages Generated by a PeopleSoft Query

This diagram illustrates the process flow of notifications produced by a PeopleSoft Query in combination with the Alert feature. The Alert process is run for one or more PeopleSoft Queries. If issues are found, then message object data is generated using the values included in the Alert Setup component, the BU-Level Notifications page, or the System-Level Notifications page. The appropriate person is notified using the Notification Dashboard page, an email, a worklist entry, or an XML message.


Alerts process
PeopleSoft Queries
Alert Setup component
Defines notification values for Message Dashboard and email
Message Registry page
Enable overrides at business unit or system level
Yes
Does an active business unit level registry exist?
Yes
Use business unit level values for Worklist and XML
No
Does an active system level exist?
Yes
Use system level values for Worklist and XML
No
Use values from Alert Setup component and Message Registry page
Notifications
Email
Worklist
XML
Custom

Generating Alert messages from a PeopleSoft Query

One or more PeopleSoft Queries are run using the Alerts process page.

Values are collected from the Alert Setup component, the BU-Level Notifications page, or the System-Level Notifications page based on process name, process category, and query name. These values include:

- Where to log the message (any combination of the Notification Dashboard, PeopleSoft Worklist, email, and XML notification).
- The message set and message number.
- User roles, user IDs, email addresses, or PeopleSoft nodes needed to route the message to the correct location.
- Links to the PeopleSoft page where the transaction can be viewed or corrected.

You must use the Notification Registry page to enable the overrides for the process name and process category.

The Alert Setup component defines values for the Notification Dashboard and email notifications. The BU-Level Notifications page or the System-Level Notifications page define values for the PeopleSoft Worklist and XML notifications. If there are no entries on the BU-Level Notifications page for the process name and the process category, then the PeopleSoft system checks to see if the values have been defined at the system level using the System-Level Notifications page.

The message is posted to the Notification Dashboard and other locations based on the values used. If the notification is sent using an email or a worklist entry, a link is provided to take the individual to the message detail page of the Notification Dashboard.

Chapter 3

Setting Up Notifications

This chapter provides an overview of Notification setup and discusses how to:

- Set up the notification registry and notification override pages.
- Set up notifications using XML messages.
- Use the pre-defined notifications.
- Modify PeopleSoft processes to create notifications.

Understanding Notification Setup

The Notification setup steps are:

1. Use the Notification Registry page to identify the notifications to be used and to enable any overrides at the system or business unit levels. In addition, the Notification Registry page can create custom actions or allow a pre-processing user exit.
2. Use the System-Level Notifications page to override information from the message that determines who should be notified (user, user role, email address, or node) and how they should be notified (worklist, email, or XML notification). You can also disable worklist, email, or XML notifications. These overrides are applied for the process name and process category combination to the entire PeopleSoft system.
3. Use the BU-Level Notifications page to override information from the message that determines who should be notified (user, user role, email address, or node) and how they should be notified (worklist, email, or XML notification). You can also disable worklist, email, or XML notifications for a specific business unit. These overrides are applied for the process name, process category, and business unit combination. Values at the business unit level override any system-level overrides.
4. Setup and activate the EOEN Notify Message to enable the PeopleSoft Integration Broker to send a message to a third-party system using a XML-formatted messages.

Additional steps are necessary if you are modifying a PeopleSoft process to generate new messages or are using PeopleSoft Query to monitor PeopleSoft tables.

Understanding System and Business Unit Level Overrides

A notification message includes values used to build the message, such as:

- Where to log the message (any combination of the Notification Dashboard, PeopleSoft Worklist, email, and XML notification).

- A message set and message number. (optional)
- User roles, user IDs, email addresses, or PeopleSoft nodes needed to route the notification message to the correct location.
- Links to the PeopleSoft page where the transaction can be viewed or corrected.

The values are obtained from the PeopleSoft process generating the notification or in the case of Alerts from the Alert Setup component, BU-Level Notifications page, and System-Level Notifications page.

For notifications that are not Alerts, you can override the values from the initial PeopleSoft process in order to accommodate a temporary change or permanently provide more specific values using:

- System-Level Notifications page.
- BU-Level Notifications page.

System-level notifications are defined by the process name and the category and enable you to override the notification for the entire system. For example, if Bob receives all of the notifications for the OM_PEGGING / RECEIPT process and he goes on an extended vacation, you can set up an override at the system level to route all of these specific messages to Steve.

Business unit-level notifications are defined by the process name, the category, and the business unit and enable you to override the notification at the business unit level. For example, if the notifications for the OM_PEGGING / RECEIPT process need to be specific to the inventory warehouse, you can set up an override at the business unit level to route all of the messages for business unit US001 to Steve and to route all of the messages for business unit US008 to Ralph.

When you set up system and business unit level notification overrides, you determine who should be notified when a notification occurs and how they should be notified. As with most default hierarchies in the PeopleSoft system, the more specific information overrides the more general information. In this case, if values are defined at both levels for the same process name and category, then the values defined at the business unit level (BU-Level Notifications page) override the values defined at the system-level (System-Level Notification page).

Setting Up the Notification Registry and Override Pages

This section discusses how to:

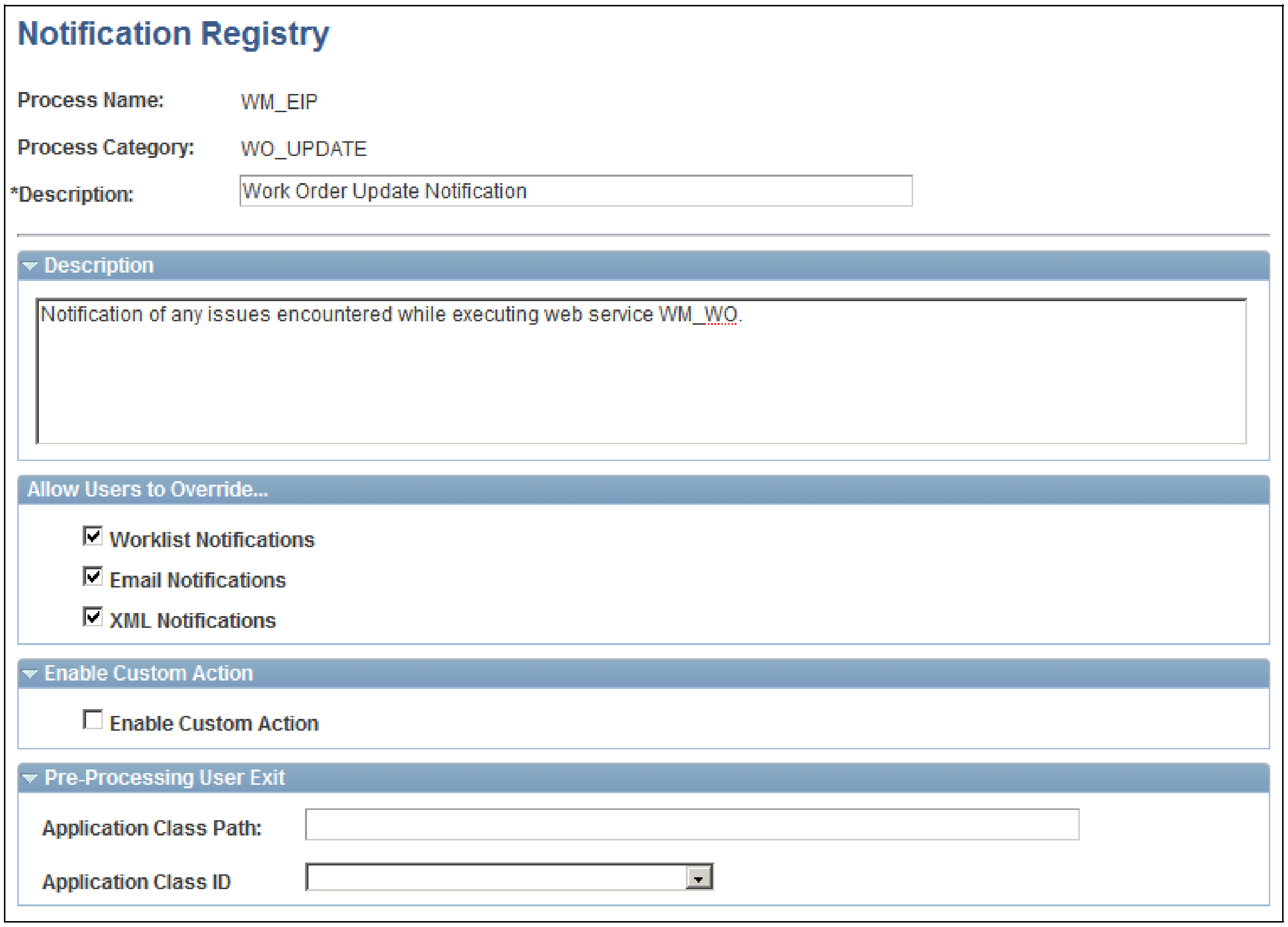
1. Setup the Notification Registry page.
2. Setup the System-Level Notifications Override page.
3. Setup the BU-Level Notifications Override page.

Pages Used to Set Up the Notification Registry and Registry Notification Pages

Page Name	***Definition Name***	***Navigation***	***Usage***
Notification Registry	EOEN_NTF_REGISTRY	Enterprise Components, Events & Notifications, Notifications Framework, Notifications Registry	Set up the notification registry to identify the notifications to be used and to enable any overrides at the system or business unit levels. In addition, the Notification Registry page can create custom actions or allow a pre-processing user exit.
System-Level Notifications Override	EOEN_NTF_REG_SYS	Enterprise Components, Events & Notifications, Notifications Framework, Notifications System Overrides	Set up system-level notifications, which define overrides at the system level for each process name and process category.
BU-Level Notifications Override	EOEN_NTF_REG_BU	Enterprise Components, Events & Notifications, Notifications Framework, Notifications BU Overrides	Set up business unit level notifications, which define overrides at the business unit level for each process name, process category, and business unit.

Setting Up the Notification Registry

Access the Notification Registry page (Enterprise Components, Events & Notifications, Notifications Framework, Notifications Registry).



Notification Registry

Process Name: WM_EIP

Process Category: WO_UPDATE

*Description: Work Order Update Notification

Description

Notification of any issues encountered while executing web service WM_WO.

Allow Users to Override...

- [✓] Worklist Notifications
- [✓] Email Notifications
- [✓] XML Notifications

Enable Custom Action

- [ ] Enable Custom Action

Pre-Processing User Exit

Application Class Path:

Application Class ID

Notification Registry page

The notification registry contains all delivered processes and categories. If you have customized your system with additional processes and categories, you must add them to the Notification Registry page.

Process Name	Enter or view the process using notifications. This field can contain an application batch or online process. For Alert notifications, a user-defined value is entered in this field.
Process Category	Enter or view the category. The category is a logical partition of the Process Name. For PeopleSoft processes, the category names are defined when the process is developed. For example, when the Reservation process runs, multiple messages are related to item ID. All of these item ID-type messages are grouped into one category so that one user can be notified when a message occurs. For each delivered process, the PeopleSoft system has categories that are delivered with the PeopleSoft product.
Description	Enter or view the extended description for this notification message by process name and category combination.

Allow Users to Override

Select the appropriate check boxes Worklist Notifications, Email Notifications, and XML Notifications to enable overrides at the system and business unit levels for this notification message. If you do not allow overrides then the notification method and destinations contained in the notification will always be used. If there are no notification methods or destinations in the notification then the notification dashboard will be the only way to view the notification

Enable Custom Action

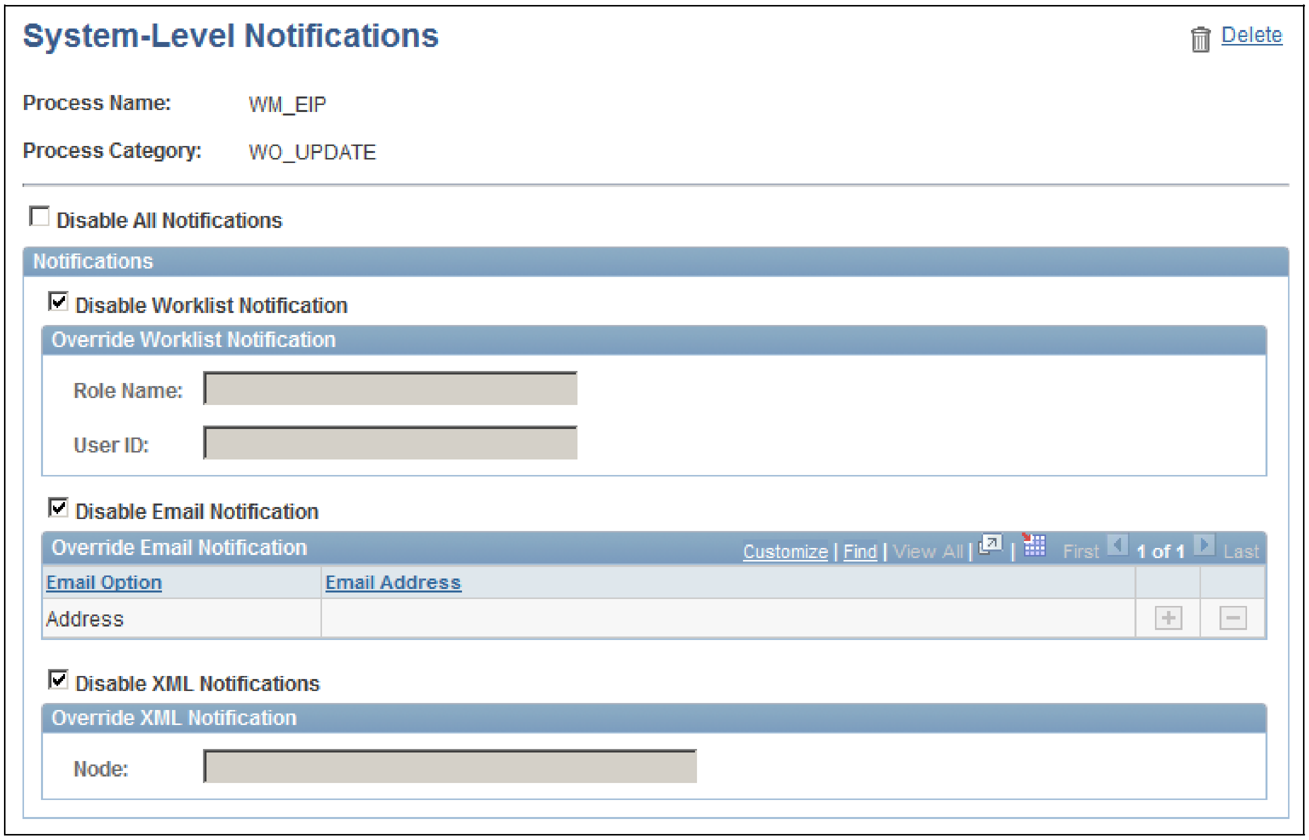
For PeopleSoft processes, select Enable Custom Action, to enable a customized action. Use this option if you have written your own application class for a customized action. Enter the Application Class Path and Application Class ID fields, which are extensions of the EOEN_MVC:MODEL:EOEN_PostProcessingUserExit class.

Pre-Processing User Exit

For PeopleSoft processes, enter the Application Class Path and Application Class ID fields. A preprocessing user exit tells the system to run another process before it determines who to notify. For example, a purchase order approval process is created that is based on the purchase order value. Purchase orders under 500.00 USD are approved by the department manager, and purchase orders over 500.01 USD are approved by the department director. The preprocessing user exit tells the system to run the purchase order approval process before determining who to notify. The preprocessing user exits are extensions of the EOEN_MVC:MODEL:EOEN__PreProcessingUserExit class.

Setting Up Registry Notifications at the System Level

Access the System-Level Notifications page (Enterprise Components, Events & Notifications, Notifications Framework, Notifications System Overrides).



System-Level Notifications

Delete

Process Name: WM_EIP

Process Category: WO_UPDATE

☐ Disable All Notifications

Notifications

☑ Disable Worklist Notification

Override Worklist Notification

Role Name:

User ID:

☑ Disable Email Notification

Override Email Notification — Customize | Find | View All | First 1 of 1 Last

Email Option	Email Address
Address	

☑ Disable XML Notifications

Override XML Notification

Node:

System-Level Notifications page

The Override Notification sections of this page are determined by the options selected on the Notification Registry page.

Disable All Notifications Select to disable all notifications for this message.

Note. Notifications will still be posted to the Notification Dashboard.

Disable Worklist Notification Select to disable only worklist notifications for this message.

Role and **User ID** Select a role or a user ID for the worklist notification. This determines the person to notify for this message.

Note. If the Worklist Notifications check box is selected on the Notification Registry page and the Role field or the User ID field are blank on the System-Level Notifications page and the BU-Level Notifications page, then the worklist notification for the original message object (either User ID or Role) is used.

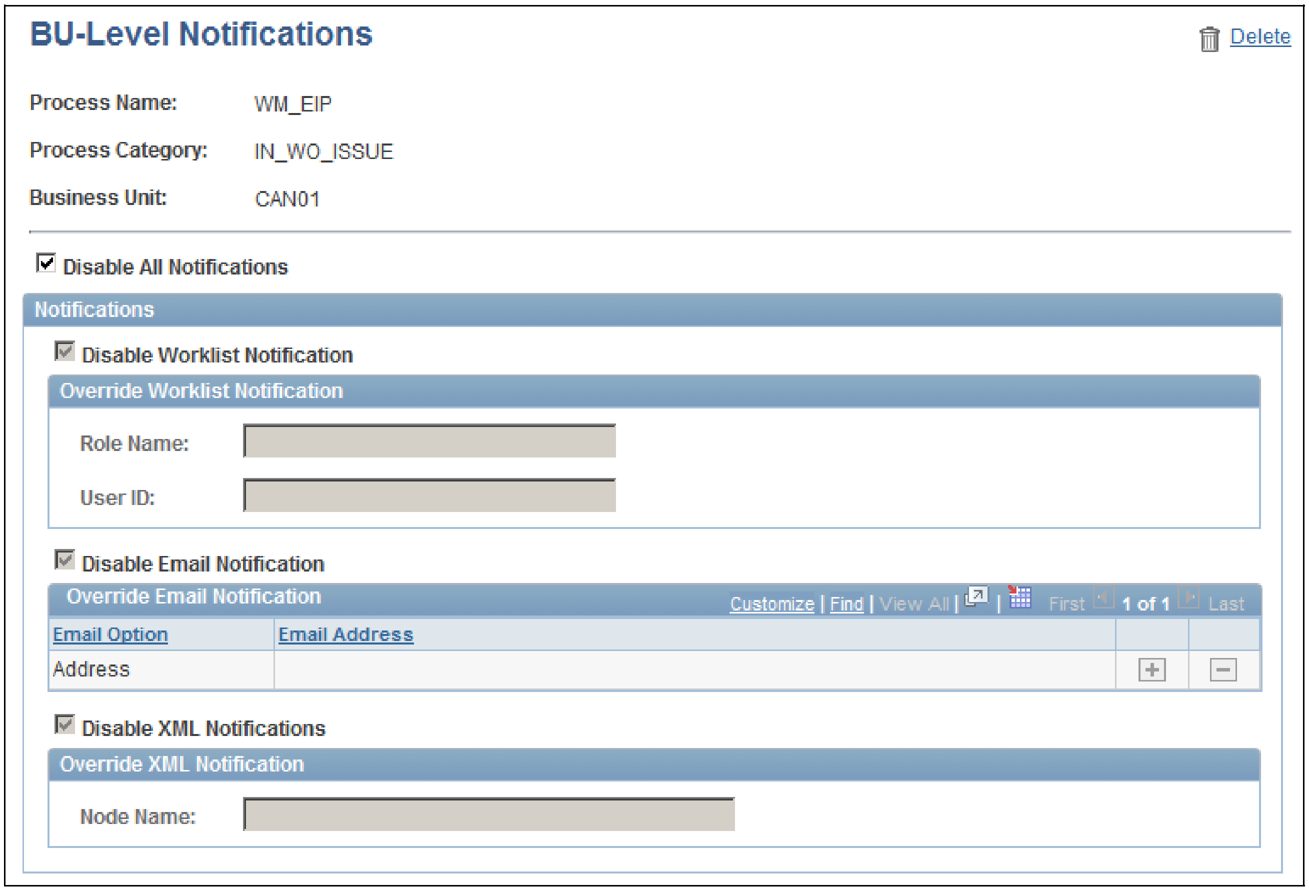
Disable Email Notification Select to disable only email notifications for this message.

Email Option	Select *Address* or *UserID* for the email notification. Depending on which option you select, the field to the right will change to *Email Address* or *User ID.* You may add multiple rows if needed. **Note.** If the Email Notifications check box is selected on the Notification Registry page and you do not enter an Email Address on the System-Level Notifications page or the BU-Level Notifications page, then the Email Address notification for the original message object is used.
Disable XML Notification	Select to disable only XML notifications for this message.
Node	Select the node name for the XML notification. **Note.** If the XML Notifications check box is selected on the Notification Registry page and you do not enter a Node on the System-Level Notifications page or the BU-Level Notifications page, then the XML notification (node) for the original message object is used.

Note. The delivered notifications contained in some PeopleSoft processes are controlled by the owner of the transaction. For example, notification messages for pegging purchase orders goes to the buyer on the purchase order. Therefore, overriding notifications at the business unit level or system level is not enabled.

Setting Up Registry Notifications at the Business Unit Level

Access the BU-Level Notifications page (Enterprise Components, Events & Notifications, Notifications Framework, Notifications BU Overrides).


BU-Level Notifications

Delete

Process Name: WM_EIP

Process Category: IN_WO_ISSUE

Business Unit: CAN01

☑ Disable All Notifications

Notifications

☑ Disable Worklist Notification

Override Worklist Notification

Role Name:

User ID:

☑ Disable Email Notification

Override Email Notification

Email Option	Email Address
Address	

☑ Disable XML Notifications

Override XML Notification

Node Name:

BU-Level Notifications Override page

The BU-Level Notifications page operates exactly like the System-Level Notifications page except that it enables you to specify a business unit as a key for notifications. Values defined at the business unit level (BU-Level Notifications page) override the same values defined at the system-level (System-Level Notification page).

Setting Up Notifications Using XML Messages

The Notify Message enables the PeopleSoft Integration Broker to send a message to a third-party system using a XML-formatted messages. The notification framework can publish an outbound message using the Notify Message (EOEN_NOTIFY_MSG) service operation to notify an external system that an error or warning has occurred in the PeopleSoft system.

Setting Up the Notify Message

PeopleSoft delivers the service operation EOEN_NOTIFY_MSG with a default status of *Inactive.* You must activate each service operation before attempting to send or receive data from a third-party source or another PeopleSoft system, such as CRM.

Using the Pre-Defined Notifications

Several PeopleSoft processes are delivered with the ability to produce notification messages. These notifications are displayed on the Notification Registry page.

Use the Notification Registry page, BU-Level Notifications page, and System-Level Notifications page to customize who should be notified (user, user role, email address, or node) and how they should be notified (worklist, email, XML notification, or custom action).

Some PeopleSoft applications deliver processes with pre-defined coding to generate notification messages, refer to the specific application documentation for a list of delivered processes and categories in the Notification Registry.

Modifying PeopleSoft Processes to Create Notifications

This section discusses how to modify PeopleSoft processes to generate notifications. Not all PeopleSoft processes are delivered with the ability to produce notifications. Your IT department can alter a batch and online process to generate notifications based on your needs.

Once you have added a new process name and process category to the Notification Registry page, you can customize a PeopleSoft process to send notifications.

Follow these steps to add to the notification registry:

1. Add your process name and category to the Notification Registry page.
   a. Add the description and long text description, which describes the functionality of the message that is logged into the registry so that you can assign the appropriate user to the notification classification.
   b. Disable any notification feature that your application does not allow you to override at the system and business unit levels.

      These notifications are handled within the object interface directly.
2. Create a message context record to pass all transactional data through the message framework.
   a. All context records must include the EOEN_LOG_KEY subrecord.
   b. Any additional transactional fields can be added to the context record.
3. Implement the EOEN interface object within your application process by using the EOEN_MVC:EOEN_MODEL.EOEN_INTERFACE class.

Chapter 4

Setting Up Events

This chapter provides an overview of event setup and event handlers and discusses how to:

- Define and register event handlers.
- Create event handlers.
- Test event handlers and events.
- Raise events.
- Monitor events.
- View event summaries.
- View help for Event Manager pages.

Understanding Event Setup

To implement Event Manager into your application processing:

1. Identify the business events that drive the execution of your business logic and, if necessary, define a new event.

   In many cases, the event is already defined in the local event registry that is accessible through the Event Registry component. If the event is not yet defined, you must define a new business event. To define a new event:

   a. Create an Application Package in PeopleTools Application Designer.

      This Application Package is a container for your Handler application classes. It needs to contain a sub-package named *Handlers* with individual application classes under it for each Handler you wish to register.

   b. Register the event to the Event Manager framework through the Event Registry component.

2. Write the event handlers in PeopleTools Application Designer to execute the business logic that is specific to the event.

   To develop an event handler, create an Application Class with a method named ProcessEvent() that includes the Event Manager interface code that is provided in this chapter. Event handlers can be for one or more business events.

3. Register the event handlers to the event through the Event Registry component.

4. Test your event and its registered event handlers.

   To test event handlers and events:

   a. Test the event handler in standalone mode through the Handler Tester component.

      Note. This test method does not require Integration Broker to be running.

   b. Test the event independent of the business process that raises the event through the Event Tester component.

      Note. This test method requires Integration Broker to be up and running.

   c. Test the event in the context of the business process by executing the business process that raises the event through the component or PeopleTools Application Engine process.

      The Event Manager framework automatically executes the registered event handlers for an event when you raise the business events.

Understanding Event Handlers

Business events are the functions in a business environment, such as adding a person, changing an employee's compensation rate, terminating an assignment, promoting an employee, and so on. After the events are created, you register events in the Event Registry component. Event handlers are modules of business logic that react to instances of particular events to which they are registered. You must register event handlers to events so that the Event Manager framework executes the event handlers whenever the event is raised.

You can assign multiple event handlers to a single business event, and you can assign a single event handler to react to multiple events. When defining events in the Event Registry component, you can assign event handlers to function in either synchronous mode or asynchronous mode with respect to the process that raises the associated event. When a process raises a registered event, the Event Manager framework determines whether the event handlers that are registered to that event are synchronous or asynchronous. If an event handler is synchronous, the Event Manager framework executes that event handler inline as part of the process that raises the event. If the event handler is asynchronous, the Event Manager framework executes the event handler as a detached process that is out of the critical path of the mainline process. You can register both synchronous and asynchronous event handlers concurrently to the same event. The Event Manager framework executes the registered event handlers in whichever mode you configure them to operate, always executing synchronous event handlers first. When defining event handlers for registry in multiple events, you should include some code in the event handler to examine the Name property of the passed-in event object so that you know which type of event caused the event handler to be executed.

The Event Manager framework is based upon a simple notify and respond model. For the notification side, the Event Manager framework broadcasts that event has occurred, along with the data that caused the creation of the event instance. For the respond side, the individual applications link business logic into the Event Manager framework to respond to the business event. By linking PeopleTools Application Classes to the event in the Event Registry component, the Event Manager framework knows when to execute the registered event handlers for the event when the event is raised.

An event handler that fails during execution causes a disruption to the execution of any other event handlers for that event or the mainline process. The Event Manager framework therefore provides many options for monitoring and troubleshooting event handlers and events. You can monitor event instances and the executions and exceptions of the event handlers through the Event Monitor component or the Event Summary component. You can also test event handlers and events outside of the mainline business process through the Handler Tester component and the Event Tester component.

Local and Remote Nodes

The Event Manger framework enables you to manage events that have influence over more than one database instance. You can use the Event Manager framework to process business events that occur on either a local or remote database (also called a node). A remote node can execute only event handlers that are implemented locally and registered to events in its local event registry. For example, an event that is raised in an HCM database might be important to a separate PeopleSoft Enterprise Learning Management (ELM) database or a PeopleSoft Financials database.

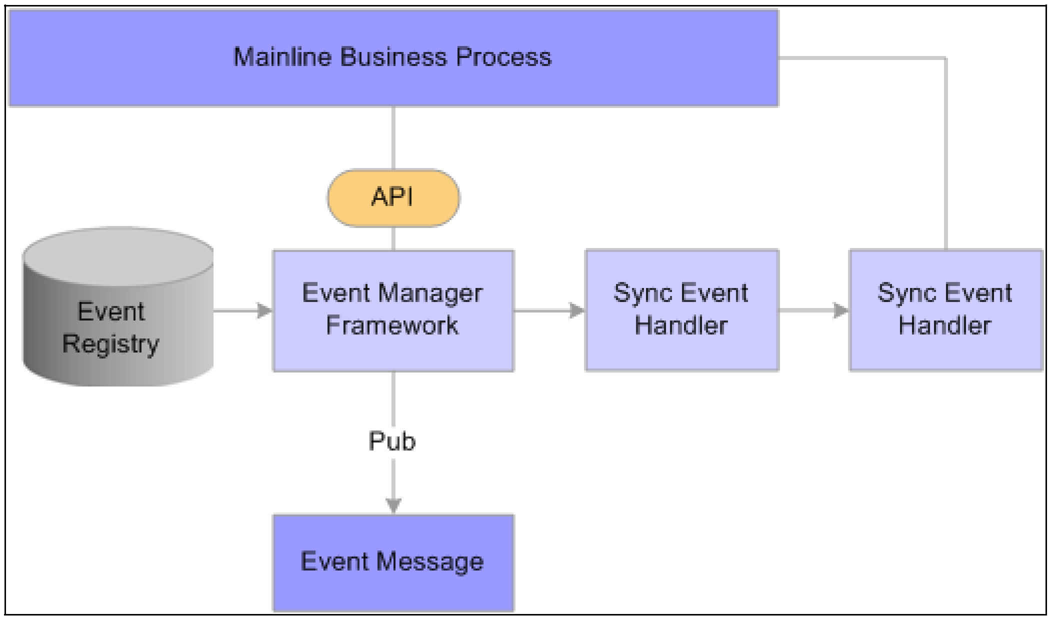
The Event Manager Framework is currently not integrated with the Service Oriented Architecture (SOA) distributed registry. Instead, each database has its own local event registry. Within a local event registry, you can define events that you can raise by local process as well as events of interest to applications on the local database that are raised by remote databases. To support the processing of an event in a remote database, remote databases must implement the Event Manager framework. Also, each local event registry must contain entries for the remote events of interest along with their associated local event handlers.

From the perspective of the Event Manager framework, virtually no difference exists between the way that asynchronous handlers are supported for remote events and local events. You register event handlers to events in the event registry, configuring the Event Manager framework to execute event handlers for events that are raised in the local database, a remote database, or both types of databases. Thus, a single event handler can respond to instances of the same event that are raised on the local database as well as a remote database.

Event handlers for remote events are by definition asynchronous. Thus, on the local database the Event Manager framework executes these remote events only after completing the execution of synchronous handlers that you have registered for the event.

Example: Local Event with Synchronous Event Handlers

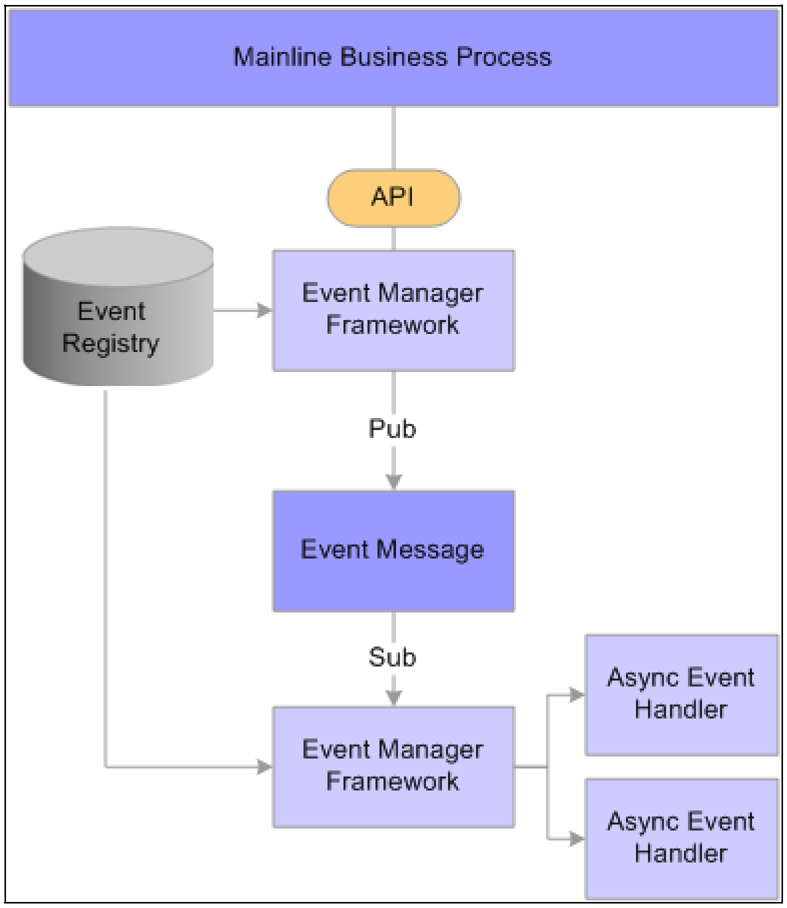
This example illustrates the processing of a mainline business process on a local database using an event with two registered synchronous event handlers. In this scenario, the mainline business process calls the application program interface (API) for the Event Manager framework to raise the event. The framework executes inline the two event handlers in order, publishes the generic PeopleTools Integration Broker event message, and then returns processing control back to the mainline business process.



Local event with synchronous handlers

Example: Local Event with Asynchronous Event Handlers

This example illustrates the processing of a mainline business process on a local database using an event with two registered asynchronous event handlers. In this scenario, the mainline process calls the API for the Event Manager framework to raise the event. The framework immediately publishes the generic PeopleTools Integration Broker event message, and returns processing control back to the mainline process without delay. A normal PeopleTools Integration Broker local subscription picks up the generic event message and invokes the framework to execute the asynchronous event handlers that are registered to the event.



Local event with asynchronous handlers

Example: Local Event with Both Synchronous and Asynchronous Event Handlers

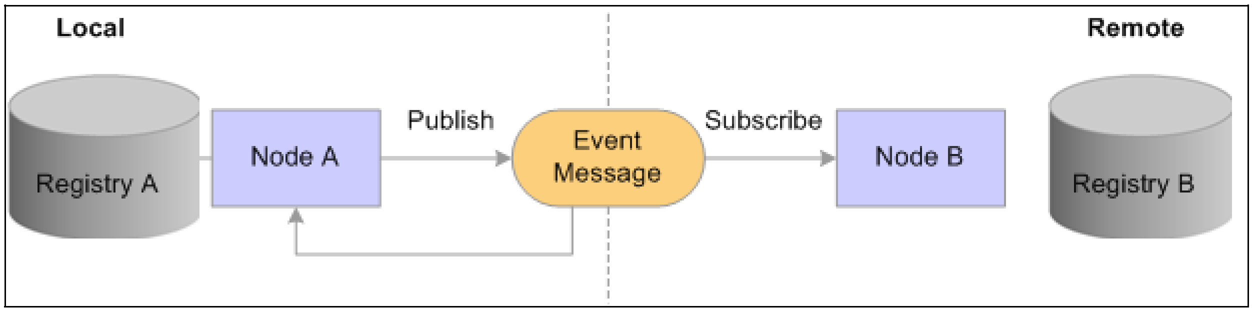
This example illustrates the processing of a mainline business process on a local database using an event with two registered synchronous handlers and two registered asynchronous event handlers. In this scenario, the mainline process waits while the Event Manager framework executes the synchronous event handlers. The Event Manager framework executes the asynchronous event handler as detached processes launched by the local subscription to the generic PeopleTools Integration Broker event message.

Example: Remote Event

This example illustrates event processing that spans across local and remote databases.

In Registry A, the event has both registered synchronous and asynchronous event handlers. The event handler that is registered to this event is set for a local source. When the event in raised in Registry A, the Event Manager framework logs the event (if it is configured for logging), publishes the generic event message, executes synchronous event handlers that are registered to the event, and then subscribes locally to the generic event message to execute asynchronous event handlers that are registered locally to the event.

Registry B has the same event active in its local event registry. The event handler that is registered to this event is set for a remote source. Because Registry B subscribes remotely to the same event message that Registry A publishes, Registry B logs the occurrence of the event and then executes the registered asynchronous handler.



Remote Event

Defining and Registering Events

To register business events and their event handlers, use the Event Registry (EOEN_EVENT_REG) component.

This section provides an overview of new events and discusses how to define and register events.

Understanding New Events

Occasionally, you need to create new events that you can raise in the system for use by the Event Manager framework. As stated in the chapter overview, the first step is to identify the business events that drive the execution of your business logic and, if necessary, define a new event. Recall that to define a new event you must:

- Create a new application package for a new event in PeopleTools Application Designer.
- Register the new event to the Event Manager framework through the Event Registry component.

If you want to register a new event handler to an existing business event, you can bypass this step.

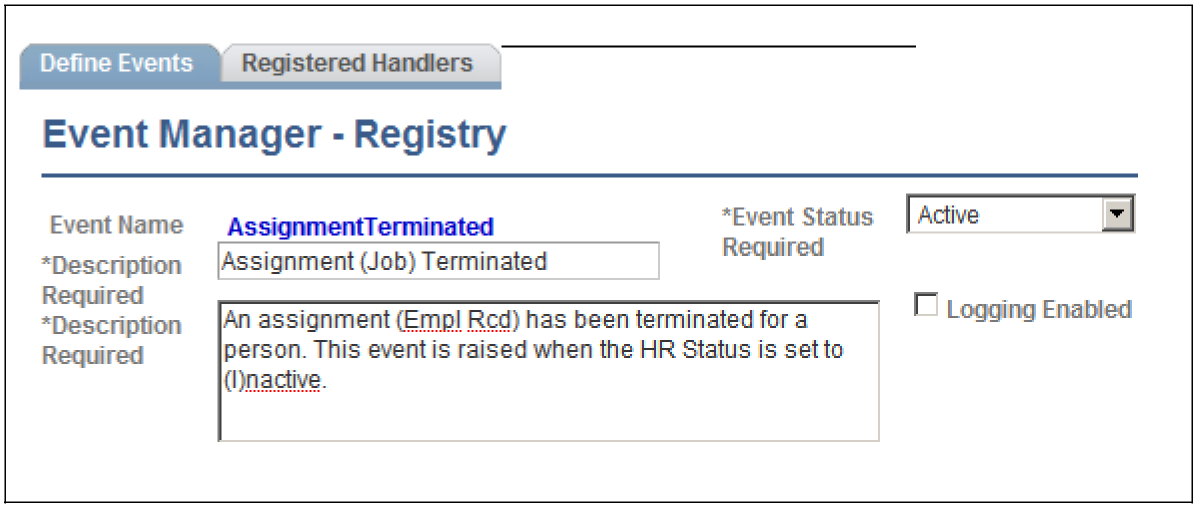
Pages Used to Define and Register Events

Page Name	***Definition Name***	***Navigation***	***Usage***
Define Events	EOEN_EVENT_REG	Enterprise Components, Events & Notifications, Event Framework, Event Registry	Register the business events that you want to be able to raise within the Event Manager framework.

Page Name	***Definition Name***	***Navigation***	***Usage***
Registered Handlers	EOEN_EVENT_REG2	Enterprise Components, Events & Notifications, Event Framework, Event Registry Select the Registered Handlers tab	Register event handlers to the selected business event. The Event Manager framework executes event handlers according to the rules that you specify on this page whenever it receives notification that the specified type of event has been raised.

Registering Events to the Event Manager Framework

Access the Define Events page (Enterprise Components, Events & Notifications, Event Framework, Event Registry).



Event Manager - Registry

Event Name AssignmentTerminated

*Description Required: Assignment (Job) Terminated

*Description Required: An assignment (Empl Rcd) has been terminated for a person. This event is raised when the HR Status is set to (I)nactive.

*Event Status Required: Active

☐ Logging Enabled

Define Events page

Event Name The system displays the name of the business event that you want to be able to raise within the Event Manager framework. If you are adding a new business event, enter the name of the event when accessing the component through the Add mode.

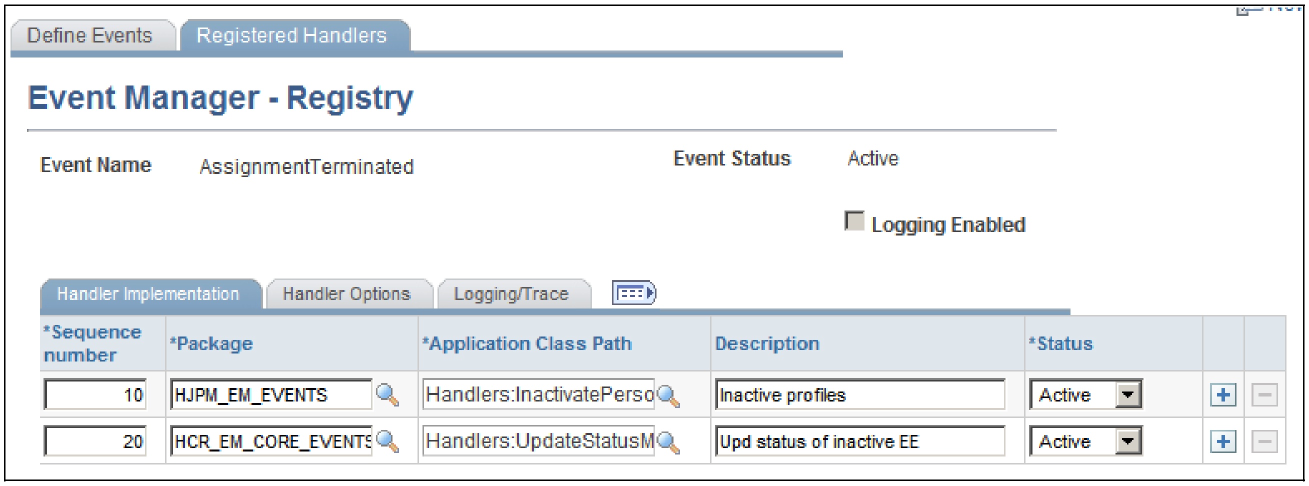
Event Status Select whether the status of the event is *Active* or *Inactive*.The system only raises events in the local database that are active. Note that the system ignores inactive events that are raised in remote databases.

Description and (Long) **Description** Enter a short and long description of the business event. The system uses the short description in search pages. You should provide enough detail of the business event in the long description so that anyone accessing the event knows what it represents.

Logging Enabled	Select to have the system write a row to a log table for each instance of this event. The system writes to the log each time that this event is raised in the local database. For events that are raised in remote databases, the system writes to the local log each time it receives a generic event message for this event.

Registering Event Handlers

Access the Registered Handlers page (Enterprise Components, Events & Notifications, Event Framework, Event Registry).



Event Manager - Registry

Event Name AssignmentTerminated

Event Status Active

Logging Enabled

*Sequence number	*Package	*Application Class Path	Description	*Status
10	HJPM_EM_EVENTS	Handlers:InactivatePerso	Inactive profiles	Active
20	HCR_EM_CORE_EVENTS	Handlers:UpdateStatusM	Upd status of inactive EE	Active

Registered Handlers page, Handler Implementation tab

The number of different events to which you can register a particular event handler is not restricted.

Handler Implementation

Sequence number	Enter a number to indicate the sequence in which the system executes the registered handlers when this event is raised. The execution sequence is important when more than one event handler is registered to an event.

Note. The sequence of event handler execution is relative to other event handlers that are operating in the same mode, either synchronous or asynchronous.

Sys Data (system data)	If this check box is selected, then PeopleSoft Human Resources delivers the event handler registration for this event as system data. You should not modify this system data unless PeopleSoft Human Resources instructs otherwise.
Package	Select the root application package that contains the application class for the event handler. The application class can reside within any application package; however, the application class must reside under a sub-package named *Handlers*.

Application Class Path	Select the application class that implements the event handler. The system displays as possible values all application classes that reside within the selected package as part of the *Handlers* subpackage.
Description	For information purposes, enter a short description of what the event handler does.
Status	Select whether the status of the event handler is *Active* or *Inactive.* The system executes event handlers that are in *Active* status only when this event is raised.

Handler Options

Event Source	Select the source from which you want the system to execute the event handler for this event. The Event Manager framework supports both local and remote events. Possible values are: • *Local & Remote:* Select to have the system execute the event handler for instances of this event that are raised in either the local database or remote databases. • *Local Only:* Select to have the system execute the event handler for instances of this event that are raised in the local database only. • *Remote Only:* Select to have the system execute the event handler for instances of this event that are raised in remote databases only.
Handler Mode	Select the mode by which you want the system to execute the event handler for this event. This setting applies only to events that are raised in the local database. Possible values are: • *Asynchronous:* Select to have the system execute the event handler as a detached or background process. The process that raises the event *does not* wait for all registered asynchronous event handlers to execute. • *Synchronous:* Select to have the system execute the event handler inline with the business process that raises the event. The process that raises the event *does* wait for all registered synchronous event handlers to execute. **Note.** The Event Manager framework always executes synchronous handlers before asynchronous handlers, regardless of the values that you specify in the Order column.

Logging/Trace

Logging	Select how you want the system to log execution of event handlers. This option applies only if you select the Logging Enabled check box for the selected event on the Define Events page. Possible values are: • *All:* Select to log every execution of this event handler. This option is available only if you select the Event Logging check box for the event of the Define Events page. • *Errors:* Select if you want to log execution of this event handler only when the execution fails. If you select this option for the event handler but do not enable logging for the event, then if the event handler fails, the system creates an event log entry along with the event handler log entry. • *None:* Select if you do not want to log execution of this event handler.
Trace	Select to include trace/debug entries in the execution log for this event handler. Use this option for debugging the event handler. This option is available only if you select to log all exceptions of the event handler by selecting the *All* value in the Logging field.

Creating Event Handlers

An event handler is the application-specific code that responds to a particular type of event. You must register event handlers to events so that when an event is raised, the Event Manager framework executes the appropriate event handlers.

Requirements for Creating Event Handlers

To be compatible with the Event Manager framework, when creating event handlers, you must implement all business logic for the event handlers to abide by the following set of standards and code requirements:

- You must implement the entry point to the event handler in a method named ProcessEvent() within an application class.

  The Event Manager framework executes the ProcessEvent()method whenever the associated event is raised. Within the ProcessEvent() method, you are free to reference or execute code in other methods, classes, or function libraries. This application class can be an extension of any other application class if necessary.

- You can create the event handler in any new or existing application package within your application, but you must create the event handler under the *Handlers* subpackage.

- The application class and the application package that it resides in must be owned by the application, not the Event Manager framework.

- You must import the baseEvent class definition from the Event Manager framework application package into the application class that implements the event handler.

  The ProcessEvent() method in the application class of the event handler receives as an input parameter an instance of this baseEvent class object.

- If you want to be able to communicate detailed errors, warnings, or any other messages back to the Event Manager framework, then you must populate the HandlerStatus property with appropriate entries.

  The HandlerStatus property is a complex property that is part of the baseEvent object that the Event Manager framework makes available to the event handler. The structure of this property is defined by EOEN_EVENT_MANAGER:Base:baseEvent. The property contains both an overall status indicator and an exception array. The Event Manager framework examines the contents of this property after executing this class and takes appropriate action.

  The HandlerStatus complex type also carries a method that facilitates appending an entry to the ExceptionArray structure. This method is named AddException() and takes as an argument the structure defined by EOEN_EVENT_MANAGER:Base:Types:ExceptionType.

  Note. If event logging is enabled, the Event Manager framework automatically creates an exception entry that lists the event data that was made available to the event handler. The system displays the event data in the pages of the Event Monitor component. If event logging is not enabled in the registry an exception entry containing the data for the event is created only if the Handler ExecutionStatus is set to *F* (Fail). This is done to allow re-processing of the event from the Event Monitor.

  The structure within the HandlerStatus property is:

  - ExecutionStatus (string – values "S"uccess, "F"ail)

  - ExceptionArray (array):

    EXCEPTION_TYPE (string – values "E"xception, "K"eys, "T"race)

    MESSAGE_SET_NBR (number)

    MESSAGE_NBR (number)

    MSG_SEVERITY (string – values "M"essage, "W"arning, "E"rror)

    MESSAGE_TEXT (string)

    EXPLAIN_TEXT (string)

Details of the baseEvent Class

When the Event Manager framework invokes the baseEvent class, it passes the baseEvent class into the ProcessEvent() method of the application class of the event handler. This enables you to use the properties and methods of the base class to obtain more information about the event instance to direct internal processing of the event.

This table lists public properties of the event object that are of interest to application developers who are implementing the Event Manager framework:

Public Properties	***Description***
EventName	A string that is populated with the name (type) of the event instance. Examples are "AssignmentTerminated", "PersonAdded", and so on. If you are registering the same event handler to multiple events but these events have slightly different logic depending upon the exact event that is being processed, you can use this property to control your processing.
EventID	A numeric that is populated with the unique serial number that identifies this event instance for an originating node. Generally, event handlers do not have any need for this property. The event handler uses this property primarily inside the framework for audit and monitoring purposes.
ContextRecord	The context record associated with the event.
HasContextRecord	A Boolean set to True if the raised event has a context record.
IsLocal	A Boolean set to True if the event was raised by an activity in the local database. (The term *local* is relative to the domain in which the event handler code is executing).
EventNode	A string that is populated with the node name of the database in which the activity that raised the event occurred.
HandlerTraceEnabled	A Boolean that indicates whether the user wants the current event handler to create trace and debug information in the event handler log.
EventLoggingEnabled	A Boolean that indicates whether the user wants the current event to create event logging information in the event log. Must be set to true if Event Handler logging is desired.
HandlerLoggingSwitch	A Boolean that indicates whether the user wants the current event handler to create event handler log information in the event log.

Event Handler Skeleton Code

You can use the following skeleton code as a starting point when building an event handler class:

```
import EOEN_EVENT_MANAGER:Base:baseEvent;
import EOEN_EVENT_MANAGER:Base:Types:ExceptionType;

class MyHandlerClass
   /** Dummy Constructor. */
   method MyHandlerClass();
   /** Process Event. */
   method ProcessEvent(&inEvent As EOEN_EVENT_MANAGER:Base:baseEvent);
end-class;

method MyHandlerClass
end-method;

method ProcessEvent
   /+ &inEvent as EOEN_EVENT_MANAGER:Base:baseEvent +/

   Local Record &recContext;

/* the following lines are required only if you want to pass detail information
    about Errors, Warnings  or other messages back to the Event Framework */
  Local EOEN_EVENT_MANAGER:Base:Types:ExceptionType &myExceptionType = ⇒
create EOEN_EVENT_MANAGER:Base:Types:ExceptionType();


   If Not &inEvent.HasContextRecord Then
      &myExceptionType.MESSAGE_SET_NBR = 18137;
      &myExceptionType.MESSAGE_NBR = 6509;
      &myExceptionType.MSG_SEVERITY = "E";
      &myExceptionType.MESSAGE_TEXT = MsgGetText(18137, 6509, ⇒
"no message found", &inEvent.EventID, &inEvent.EventNode);
      &myExceptionType.EXPLAIN_TEXT = "";
      &inEvent.HandlerStatus.AddException(&myExceptionType);

   Else
⇒
/*================================================================== */⇒
⇒
/* ----------------   Business Logic goes here ------------------*/⇒

/*================================================================== */⇒
   End-If;
end-method;
```

Addressing Errors and Warnings

During execution, an event handler might encounter errors, warnings, or other issues that a user needs to be made aware of. In most cases, you want to pass this information back to the Event Manager framework so that users can view the information through the Monitor Events component. Although you do not have to pass this information back to the Event Manager framework, the more information that you do pass back, the easier it is for the administrator to resolve the problem.

The Event Manager framework uses a simple exception-array scheme to receive status information that is passed back to it from an event handler and subsequently displays that information in the Monitor Events component. The structure of the exception array is the same as the structure of a message in the message catalog. It contains a message set number, message number, severity level, message text, and explanation text in each row of the array. The message set number and message number are significant only if the information that you want to pass back to the Event Manager framework is described in a Message Catalog entry. If not, you can leave these two properties blank and just populate the message text, explanation text, and severity properties.

As mentioned previously, the event object that is passed into your event handler contains a complex property that is used to communicate the status of your event handler. At a minimum, you should populate the overall-status simple property with either (S)uccess or (F)ail prior to quitting your ProcessEvent() method. If you do not populate the overall status property, the Event Manager framework assumes that the event handler executed successfully. PeopleSoft recommends that you pass back information through the exception array of the complex property, although this is optional. For example, if the event object is passed into your handler as &inEvent, you populate the overall status property as:

```
&inEvent.HandlerStatus.ExecutionStatus = "S";⇒
     /* Handler terminated successfully */
```

You never have to deal directly with the exception array that is embedded in the event object that the event handler receives. Instead, whenever you want to add an item to this array, you call the AddException() method, thus passing in an object that the system has already populated with exception information. This object that you pass in is an instance of the ExceptionType TypeClass. This class has the following properties:

- Message Set Number
- Message Number
- Message Severity
- Message Text
- Explain (Long) Text

The following example shows how to add a message to the exception array for a message that does not come from the Message catalog, assuming that the event object is passed into your event handler as &inEvent:

```
&myExceptionType = create EOEN_EVENT_MANAGER:Base:Types:ExceptionType();
&myExceptionEntry.MESSAGE_SET_NBR = 0;
&myExceptionEntry.MESSAGE_NBR = 0;
&myExceptionEntry.MSG_SEVERITY = "M";
&myExceptionEntry.MESSAGE_TEXT = "Hello World!";
&myExceptionEntry.EXPLAIN_TEXT = "";

&inEvent.HandlerStatus.AddException(&myExceptionType);
```

For a Message Catalog entry, you must include the message set number and the message number. You must also retrieve the text and explain text from the Message Catalog because substitution tokens might be involved that the Event Monitor framework cannot pass or resolve.

Note. If you use the AddException() method to create an exception array entry with the MSG_SEVERITY property set to (E)rror, then the system automatically sets the overall execution status indicator for the event handler to (F)ail. This means that you do not need to separately set the overall status property.

Note. The Event Manager framework executes all event handlers that you register to an event. If a particular event handler fails, the Event Manager framework still executes the remaining event handlers that are registered to the event. The Event Manager framework can catch exceptions that an event handler raises rather than terminating before it has finished executing all of the registered event handlers. However, the Event Manger framework does not contain any exception processing or error handling subsystems, and because of this you must ensure that event handler code does not raise an exception that the Event Manager framework cannot catch. One such exception is the PeopleCode error statement. Therefore, you must never raise an exception for an error statement in your event handler. Otherwise, the Event Manager framework stops and does not execute any remaining event handlers that are registered to the event.

Component Interface and SOA Service Exceptions

If you have PeopleSoft Human Resources two additional exception methods are available: Component Interfaces (CIs) and Service Oriented Architecture (SOA) services. The Event Manager framework therefore provides assistance to facilitate passing exceptions from these sources back to the Event Manager framework and ultimately into the Event Monitor component.

The system displays exceptions that are raised by CIs or SOA services on the Handler Exceptions page of the Event Monitor component, much like any other exceptions that are written to the log. For severity information, the system prefaces the message with "[CI] " or "[SOA]", respectively.

The structure of the event-handler exception log is similar to the exception structure within PeopleTools for CIs and within the SOA framework for SOA services. Essentially, you must pass these exceptions back to the Event Manager framework when they arise. Two methods are available within the Event Manager framework that enable you to do this easily. One method is:

```
.HandlerStatus.AddCIException(&arrExceptions as array of Record, &Severity as⇒
 string)
```

This method copies all exceptions from a standard CI exception array to the exception log of the event handler according to the specified severity of the exception. Severity codes are (E)rror, (W)arning, and (M)essage.

You can specify multiple severity codes by concatenation. Thus, if you want to copy both errors and warnings, then specify a severity code of EW. A blank severity code causes the copying of all exceptions regardless of severity.

The structure of the exception array that is passed into this method is identical to the exception array that the FUNCLIB_CI version of the standard HCM CI wrapper uses. This is an array of Record.CI_EXCEPTIONS. If you're using the FUNCLIB_CI wrapper, you can pass the exception array that you use with those functions into this method.

If you are calling a CI directly, then you need to first ensure that you have instructed PeopleTools to redirect CI exceptions from the screen to the %Session.PSMessages object. Then, you need to copy the items from PSMessages into an array of Record.CI_EXCEPTIONS so that these exceptions can be passed into this method.

Another method is:

```
.HandlerStatus.AddSOAException(&inException As HMCR_TYPES:ExceptionTypes:Exception⇒
Type_v1_0:ExceptionType)
```

This method copies all exceptions from a standard SOA exception type structure to the exception log of the event handler.

When invoking SOA services, the system always executes them in a try/catch block. Typically, the event handler executes the services and catches certain defined exceptions that arise. A catch-all exception always catches any exceptions that have not been specifically called out.

SOA exceptions differ slightly from CI exceptions in that when invoking an SOA service, you manage only one (if any) exception. The exception that you pass into this method is the exception structure that you catch in your try/catch block.

The following example shows an SOA service invocation passing exceptions back to the Event Manager:

```
     *      try
     *        &MyService.MyServiceManager.LocateService(&ServiceName);
     *        &MyService.DoService();
     *          catch <NamedException1> &e1
     *                &myEvent.HandlerStatus.AddSOAException(&e1.ServiceException);
     *                --- other exception processing ---
     *          catch <NamedException2> &e2
     *                &myEvent. HandlerStatus.AddSOAException(&e2.Service⇒
Exception);
     *                --- other exception processing ---
     *          . . .
     *          catch HMCR_FRAMEWORK:ServiceFramework:baseClasses:baseException⇒
 &e0
     *                &myEvent. HandlerStatus.AddSOAException(&e0.Service⇒
Exception);
     *                --- other exception processing ---
     *      end-try
```

In this code, the items appearing as <NamedExceptionX> represent the application-specific exceptions that are registered in the SOA registry. These are exceptions that are known to be thrown by the indicated SOA service. The class named HMCR_FRAMEWORK:ServiceFramework:baseClasses:baseException is for the catch-all exception, defining the shape of the generic SOA exception.

Providing Trace/Debug Information

The Registered Handlers page of the Event Registry component provides a Trace check box to indicate whether to write trace/debug information to the execution log of the event handler. This functionality is available only when you set the Logging field for the event handler to the *All* value.

When you select the Trace check box, the event handler produces trace/debug in the handler execution log. The amount of trace/debug information that the system writes to the log is determined by the developer of the event handler. Generally, you must provide enough information so that anyone examining the event handler log can trace the path of the execution that the event handler has taken.

The system makes the selected value of this check box available to the event handler using the HandlerTraceEnabled Boolean property of the event. When this property is True, the event handler creates trace/debug entries in the exceptions structure. PeopleSoft software provides the following method to make the creation of these entries easier:

```
&myEvent.HandlerStatus.AddTraceEntry(&Heading, &Detail);
```

In the preceding code, &Heading is the string that appears as the first line of the entry, and &Detail is a long string containing whatever detailed trace/debug information you want to create.

The system displays the trace/debug information in the Event Monitor component.

No fixed rules are associated with the trace/debug entries that the event handler generates. This is a judgement call. You should provide whatever entries you think are useful to debug or troubleshoot the event handler code.

Create trace/debug entries only when the .HandlerTraceEnabled property is True. Thus, the code looks similar to:

```
   If &myEvent.HandlerTraceEnabled Then
      &myEvent.HandlerStatus.AddTraceEntry("Method ABC()", "No data returned for:⇒
 " | &Emplid);
   End-If;
```

Testing Event Handlers and Events

This section provides an overview of event testing and discusses how to:

- Test an event handler.
- Test an event.

Understanding Event Testing

Generally in a production environment, you raise events by making a change to application data through a component or by running a PeopleTools Application Engine process. When the event is raised, the Event Manager framework executes the event handlers registered with the event as defined in the Event Registry component. When testing events, you should begin at the lowest level, which is the event handler, and then work your way up to the highest level, which is the business process that raises the event. Thus, to test events:

1. Test the event handler in standalone mode through the Handler Tester component.
2. Test the event independent of the business process that raises the event through the Event Tester component.
3. Test the event in the context of the business process by executing the business process that raises the event through the component or PeopleTools Application Engine process.

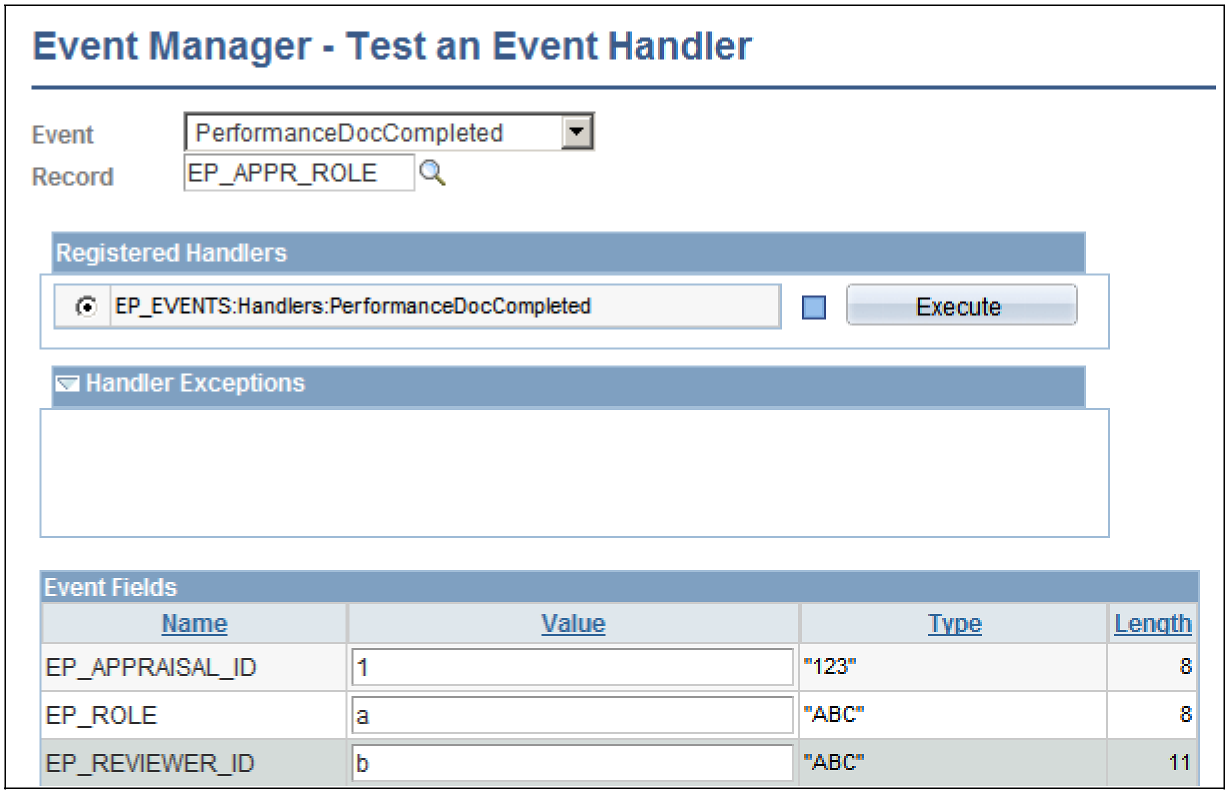
Pages Used to Test Event Handlers and Events

Page Name	***Definition Name***	***Navigation***	***Usage***
Event Manager - Test an Event Handler	EOEN_HNDLR_TEST	Enterprise Components, Events & Notifications, Event Framework, Handler Tester	Execute a single event handler as a standalone process without raising an event. Use this feature for testing and debugging event handlers.

Page Name	***Definition Name***	***Navigation***	***Usage***
Event Manager - Raise an Ad-Hoc Test Event	EOEN_EVENT_TEST	Enterprise Components, Events & Notifications, Event Framework, Event Tester	Raise a selected business event that you have defined on an ad hoc basis without invoking the business process that normally raises the event. This enables you to test events without having to perform the business operation that normally raises the event. Execute a single event handler as a standalone process without raising an event. Use this feature for testing and debugging event handlers.

Testing an Event Handler

Access the Event Manager - Test an Event Handler page (Enterprise Components, Events & Notifications, Event Framework, Handler Tester).



Event Manager - Test an Event Handler

Event	PerformanceDocCompleted
Record	EP_APPR_ROLE

Registered Handlers

● EP_EVENTS:Handlers:PerformanceDocCompleted

Event Fields

Name	Value	Type	Length
EP_APPRAISAL_ID	1	"123"	8
EP_ROLE	a	"ABC"	8
EP_REVIEWER_ID	b	"ABC"	11

Event Manager - Test an Event Handler page

Event

Select the event to which the event handler that you want to execute is registered. Event handlers must always execute within the context of an event.

Record

When you select an event, the system also allows you to select a record to provide data for the event. Typically, this is the record in the database from which a data change normally raises the event. however you can select a different record. The record must have the same keys as the record that normally raises the event. After you select a record the fields from the record are populated on the page with fields to enter values, expected types and lengths. Enter your sample data for your Event Test in the value fields and click the Execute button.

The system displays in the Registered Handlers group box the event handlers that are registered with the selected event in the local database through the Event Registry component.

Registered Handlers

This group box displays the event handlers that you have registered with the selected event in the Event Registry component.

Execute

Click to execute the selected event handler. Before you can execute an event handler, you must select the event, record, and event handler and enter the event data that you want to pass to the event handler. You can execute only one event handler at a time.

The system executes the event handler immediately regardless of any settings in the Event Registry component. When testing an event handler through this page, the system does not raise an event, publish the generic PeopleTools Integration Broker message, or generate log entries.

The system displays the status of the event handler execution to the left of this button and displays any exceptions that the event handler creates in the Handler Exceptions group box directly on this page.

🟦	Indicates that you have not yet executed the event handler.
✓	Indicates successful execution of the event handler.
✗	Indicates that the system executed the event handler with a fatal error.

Handler Exceptions

Use this group box to view results of the event handler execution. The system displays event keys, exceptions, and trace information that is associated with the execution of the event handler.

🔑	The system displays the event data that the system used during the execution of the event handler. At a minimum, the system displays the event data in the Handler Exceptions group box after execution of the event handler.
ℹ	The system displays the exception information that the system generated during execution of the event handler.

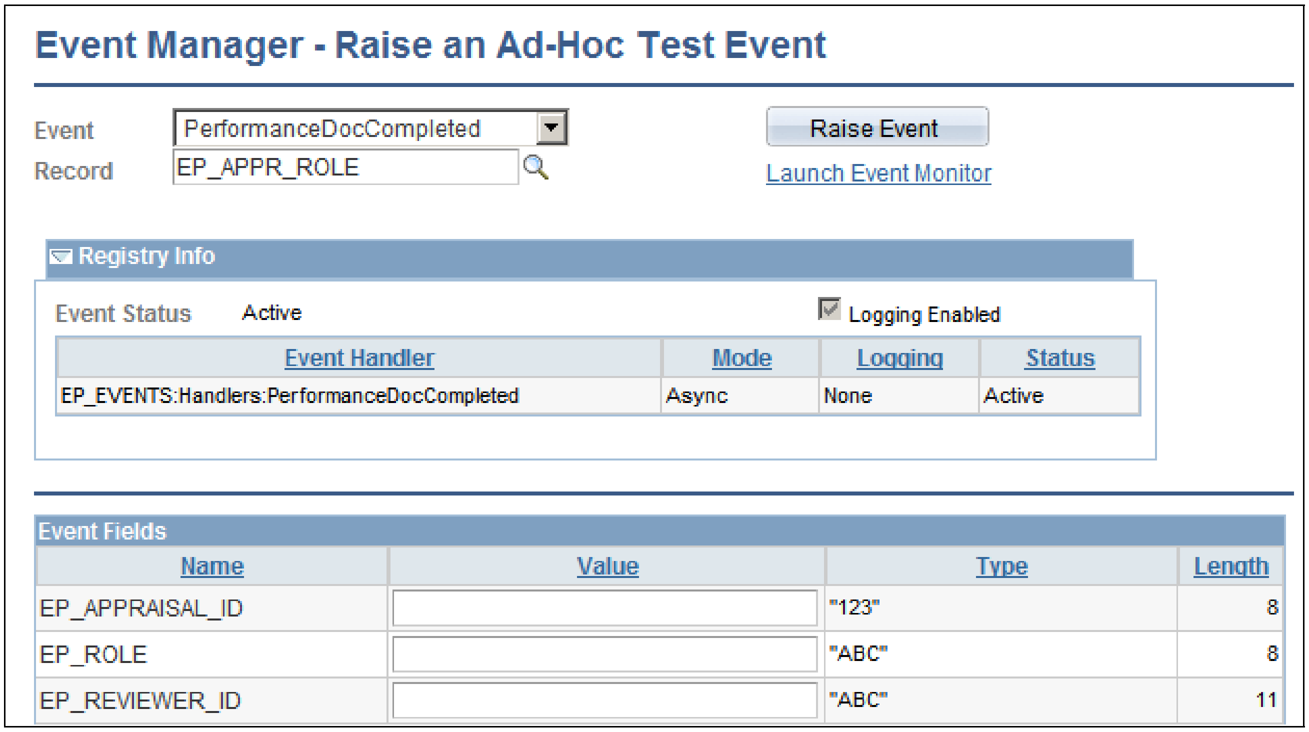
⚠	The warnings that the system generated during execution of the event handler.
✖	The errors that the system generated during execution of the event handler.
📝	The trace information that the system generated during execution of the event handler.

Event Fields

Use this group box to enter values for each of the fields that are part of the selected event and record. These are the values that the system passes into the event handler during the test. The system does not perform any validation on the data that you enter in the Value fields. Be sure to enter the correct format for data values and use uppercase or lowercase values for character data as appropriate. The system makes this data available to the event handlers as the Context Record for the Event.

Testing an Event

Access the Event Manager - Raise an Ad-Hoc Test Event page (Enterprise Components, Events & Notifications, Event Framework, Event Tester).



Event Manager - Raise an Ad-Hoc Test Event

Event: PerformanceDocCompleted

Record: EP_APPR_ROLE

Registry Info

Event Status: Active

Logging Enabled: ✓

Event Handler	Mode	Logging	Status
EP_EVENTS:Handlers:PerformanceDocCompleted	Async	None	Active

Event Fields

Name	Value	Type	Length
EP_APPRAISAL_ID		"123"	8
EP_ROLE		"ABC"	8
EP_REVIEWER_ID		"ABC"	11

Event Manager - Raise an Ad-Hoc Test Event page

Event	Select the event that you want to raise. You can only raise events that are active in the local event registry. If you are unsure of the status of an event, select the event in the Event field and click the Event Info button.

Record	When you select an event, the system allows you to select a record to provide data for this event. Typically, this is the record in the database from which a data change normally raises the event. The system displays the Event Fields grid so that you can enter data to pass to the event.
ℹ	Click the Event Info button to view select information from the Event Registry component about the selected event and its registered event handlers. The system displays the Registry Info group box, which contains the event details.
Raise Event	Click this button to have the Event Manager framework raise the event with the specified event data. The system processes the event and executes all active event handlers that you have registered to the event exactly as if you have raised it from the normal business process such as through a component or PeopleTools Application Engine process. The system displays a message below the Record field indicating whether you have successfully raised the event. This message also includes the event ID on the local node and, for multichannel events, the event channel.
Launch Event Monitor	After you raise the event, click this link to view the execution status of the raised event. The system launches the Event Monitor component in a new browser window and automatically displays information for this particular event instance.

Registry Info

Use this group box to view select information from the Event Registry component about the selected event and its registered handlers. The system displays this group box only if you select an event and click the Event Info button. To change any of this data, you must access the Event Registry component for the selected event.

Event Fields

Use this group box to enter values for each of the event fields that are part of the event that you are raising. The system does not perform any validation on the data that you enter in the Value fields. Be sure to enter the correct format for data values and use uppercase or lowercase values for character data as appropriate. The system makes these field values available to the event handlers as the context record for the event.

Raising Events

After you determine that an occurrence of the business event exists that the event in the Event Manager framework represents, you can raise the event in the Event Manager framework.

The Event Manager framework provides an API that creates and raises events. The API code is:

```
import EOEN_MVC:EOEN_MODEL:EOENInterface;
Local EOEN_MVC:EOEN_MODEL:EOENInterface &myEvent;
         Local Record &recContext;

         &myEvent = create EOEN_MVC:EOEN_MODEL:EOENInterface("", 0);

         &recContext = &rsContextRec(1).GetRecord(Record.CONTEXT_REC);
         &myEvent.AddContextRecord(&recContext);
         &myEvent.RaiseEvent("EventName");
```

Pass the name of the event to the Event Interface, which returns an instance of the event object. Add the context record and then execute the RaiseEvent method on the event object.

Important! Instantiating an event object does not automatically raise the event. You must invoke the event's RaiseEvent method to raise the event and initiate subsequent Event Manager framework processing.

In the following example, assume that we have created an event called AssignmentMilitaryRankChanged. This event represents the business event of assigning or updating the military rank of a person's assignment (Job record) instance. Whenever the rank attribute of the assignment (Job record row) changes, we want to raise the event. Here is the pseudocode to add to the JOB_MIL.MIL_WORN_RANK:SavePostChange PeopleCode event for the Job component:

```
import EOEN_MVC:EOEN_MODEL:EOENInterface;

/**********************************************************/
/* FUNCTION:    GetPriorRank                              */
/* Inputs:      &rsJOB As Rowset, &CurrentRow as number   */
/* Returns:     none                                      */
/*                                                        */
/* Get Prior Worn Rank                                    */
/**********************************************************/
Function GetPriorRank(&rsJOB As Rowset, &CurrentRow As integer,⇒
 &PRIOR_ROW As integer, &PRIOR_RANK As string);

   Local date &PRIORDT = Date3(1900, 1, 1);
   Local integer &PRIORSEQ = 0;
   Local date &JOB_Effdt = &rsJOB(&CurrentRow).JOB.EFFDT.Value;
   Local integer &JOB_Effseq = &rsJOB(&CurrentRow).JOB.EFFSEQ.Value;
   Local integer &i;

   &PRIOR_ROW = 0;
   &PRIOR_RANK = " ";
   For &i = 1 To &rsJOB.RowCount;
      Local date &FETCH_EFFDT = &rsJOB(&i).JOB.EFFDT.Value;
      Local integer &FETCH_EFFSEQ = &rsJOB(&i).JOB.EFFSEQ.Value;
      If (&FETCH_EFFDT < &JOB_Effdt Or
            (&FETCH_EFFDT = &JOB_Effdt And
               &FETCH_EFFSEQ < &JOB_Effseq)) And
            (&FETCH_EFFDT > &PRIORDT Or
               (&FETCH_EFFDT = &PRIORDT And
                  &FETCH_EFFSEQ > &PRIORSEQ)) Then
         &PRIORDT = &FETCH_EFFDT;
         &PRIORSEQ = &FETCH_EFFSEQ;
         &PRIOR_ROW = &i;
      End-If;
   End-For;
   If &PRIOR_ROW > 0 Then
      &PRIOR_RANK = &rsJOB(&PRIOR_ROW).GetRowset(Scroll.JOB_MIL)⇒
(1).JOB_MIL.MIL_WORN_RANK.Value;
   End-If;
End-Function;

/*****************************************************************************/
/* FUNCTION:    RaiseMilitaryRankEvent                                       */
/* Inputs:      JOB_MIL rowset, Current row                                  */
/* Returns:     none                                                         */
/*                                                                           */
/* Check if MilitaryRankEvent event should be raised                         */
/* if a change has been done:                                                */
/*    - Worn rank changed                                                    */
/*    - Row deleted and the worn rank is different from the previous one     */
/*    - Row inserted and the worn rank is different from the previous one    */
/* the event will be raised                                                  */
/*****************************************************************************/
Function RaiseMilitaryRankEvent(&rsJOB As Rowset)
   If INSTALLATION.MILITARY = "Y" Then
      Local integer &PRIOR_ROW;
      Local string &PRIOR_RANK;
      Local boolean &RaiseEvent = False;
      Local integer &i;

      For &i = 1 To &rsJOB.RowCount
         /* Get current Job row and Worn Rank value */
         Local Row &rowJOB = &rsJOB.GetRow(&i);
         Local Rowset &rsJOB_MIL = &rowJOB.GetRowset(Scroll.JOB_MIL);
         /*************************/
         /* Rank has been updated */
         /*************************/
         If &rsJOB_MIL(1).JOB_MIL.MIL_WORN_RANK.IsChanged Or
               (&rowJOB.JOB.EFFDT.IsChanged And
                  Not &rowJOB.IsNew) Then
            If All(&rsJOB_MIL(1).JOB_MIL.MIL_WORN_RANK.Value) Then
               &RaiseEvent = True;
               Break;
            End-If;

         Else
            /***************/
            /* New JOB row */
            /***************/
            If &rowJOB.IsNew Then
               /* Get previous Rank */
               GetPriorRank(&rsJOB, &i, &PRIOR_ROW, &PRIOR_RANK);
               If &PRIOR_ROW > 0 Then
                  If &rsJOB_MIL(1).JOB_MIL.MIL_WORN_RANK.Value <> &PRIOR_RANK And
                        All(&rsJOB_MIL(1).JOB_MIL.MIL_WORN_RANK.Value) Then
                     &RaiseEvent = True;
                     Break;
                  End-If;
               Else
                  /* No previous Rank */
                  If All(&rsJOB_MIL(1).JOB_MIL.MIL_WORN_RANK.Value) Then
                     &RaiseEvent = True;
                     Break;
                  End-If;
               End-If;
            Else
               /****************************/
               /* JOB row has been deleted */
               /****************************/
               If &rowJOB.IsDeleted Then
                  /* Get previous Rank */
                  GetPriorRank(&rsJOB, &i, &PRIOR_ROW, &PRIOR_RANK);
                  If &PRIOR_ROW > 0 Then
                     If &rsJOB_MIL(1).JOB_MIL.MIL_WORN_RANK.Value <> &PRIOR_RANK And
                           All(&rsJOB_MIL(1).JOB_MIL.MIL_WORN_RANK.Value) Then
                        &RaiseEvent = True;
                        Break;
                     End-If;
                  Else
                     /* No previous Rank */
                     If All(&rsJOB_MIL(1).JOB_MIL.MIL_WORN_RANK.Value) Then
                        &RaiseEvent = True;
                        Break;
                     End-If;
                  End-If;
               End-If;
            End-If;
         End-If;
      End-For;
      If &RaiseEvent Then

         Local EOEN_MVC:EOEN_MODEL:EOENInterface &myEvent;
         Local Record &recContext;

         &myEvent = create EOEN_MVC:EOEN_MODEL:EOENInterface("", 0);

         &recContext = &rsJOB_MIL(1).GetRecord(Record.JOB_MIL);
         &myEvent.AddContextRecord(&recContext);
```

```
         &myEvent.RaiseEvent("AssignmentMilitaryRankChanged");
      End-If;
   End-If;
End-Function;

If GetLevel0()(1).GetRowset(Scroll.JOB).GetRow(CurrentRowNumber(1)).RowNumber = 1 Then
   RaiseMilitaryRankEvent(GetLevel0()(1).GetRowset(Scroll.JOB));
End-If;
```

The lines in the preceding code that deal with the action of raising the event are:

```
Local EOEN_MVC:EOEN_MODEL:EOENInterface &myEvent;
         Local Record &recContext;

         &myEvent = create EOEN_MVC:EOEN_MODEL:EOENInterface("", 0);

         &recContext = &rsJOB_MIL(1).GetRecord(Record.JOB_MIL);
         &myEvent.AddContextRecord(&recContext);
         &myEvent.RaiseEvent("AssignmentMilitaryRankChanged");
```

Important! During the course of raising an event, the framework causes certain tracking and logging information to be stored in the database. Because of this, events can be raised only from PeopleCode events that normally allow database updates. More specifically, events can be raised only from SavePreChange, SavePostChange, and Workflow PeopleCode events.

Monitoring Events

This section discusses how to:

- Review event instances.
- Review event handler executions.
- Review event handler exceptions.

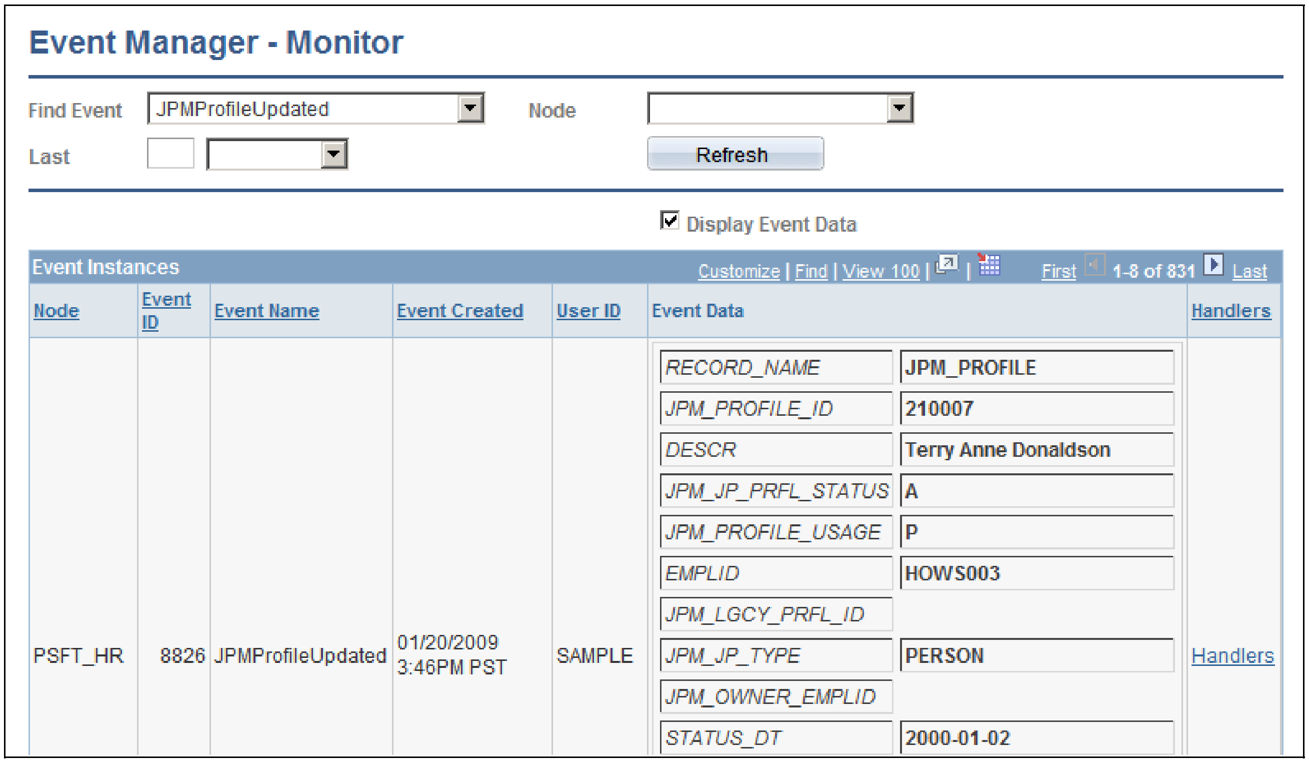
Pages Used to Monitor Events

Page Name	*Definition Name*	*Navigation*	*Usage*
Event Manager - Monitor	EOEN_EVENT_MON	Enterprise Components, Events & Notifications, Event Framework, Event Monitor	Review the status of raised or processed business events and their associated event handler executions.
Event Monitor - Handlers	EOEN_EVENT_MON2	Enterprise Components, Events & Notifications, Event Framework, Event Monitor Click the Handlers link for an event instance on the Event Manager - Monitor page.	Review the details of registered event-handler executions on the local node for a selected event instance.

Page Name	***Definition Name***	***Navigation***	***Usage***
Event Monitor - Handler Exceptions	EOEN_EVENT_MON3	Enterprise Components, Events & Notifications, Event Framework, Event Monitor Click the Handlers link for an event instance on the Event Manager - Monitor page. Click the Display Exceptions button on the Event Monitor - Handlers page.	View the data, exceptions, and trace information that the selected event handler logged during execution and passed back to the Event Manager framework.

Reviewing Event Instances

Access the Event Manager - Monitor page (Enterprise Components, Events & Notifications, Event Framework, Event Monitor).


Event Manager - Monitor

Find Event: JPMProfileUpdated
Node:
Last:
Refresh

☑ Display Event Data

Event Instances

Node	Event ID	Event Name	Event Created	User ID	Event Data		Handlers
PSFT_HR	8826	JPMProfileUpdated	01/20/2009 3:46PM PST	SAMPLE	RECORD_NAME	JPM_PROFILE	Handlers
					JPM_PROFILE_ID	210007	
					DESCR	Terry Anne Donaldson	
					JPM_JP_PRFL_STATUS	A	
					JPM_PROFILE_USAGE	P	
					EMPLID	HOWS003	
					JPM_LGCY_PRFL_ID		
					JPM_JP_TYPE	PERSON	
					JPM_OWNER_EMPLID		
					STATUS_DT	2000-01-02	

Event Manager - Monitor page

Search Criteria

Enter the search criteria for retrieving event instances. At a minimum, you must select at least one criterion to retrieve results. Use the remaining fields to narrow your search results.

Find Event	Select the event from the local Event Registry component for which you want to review event instances.
Last	Specify the time period in which the event was raised. For remote events, this is the time period in which the event appeared in the local node. In the first box, specify a numeric quantity. Then select a time qualifier: *Days*,*Hours*, or *Minutes*.
Node	Select a node name to limit the search results to event instances of raised events within a specific node.
Refresh	Click to refresh the list of event instances after changing any of the search criteria.
Display Event Data	Select to display the event data for each event instance. The system displays the Event Data column in the Event Instances grid.

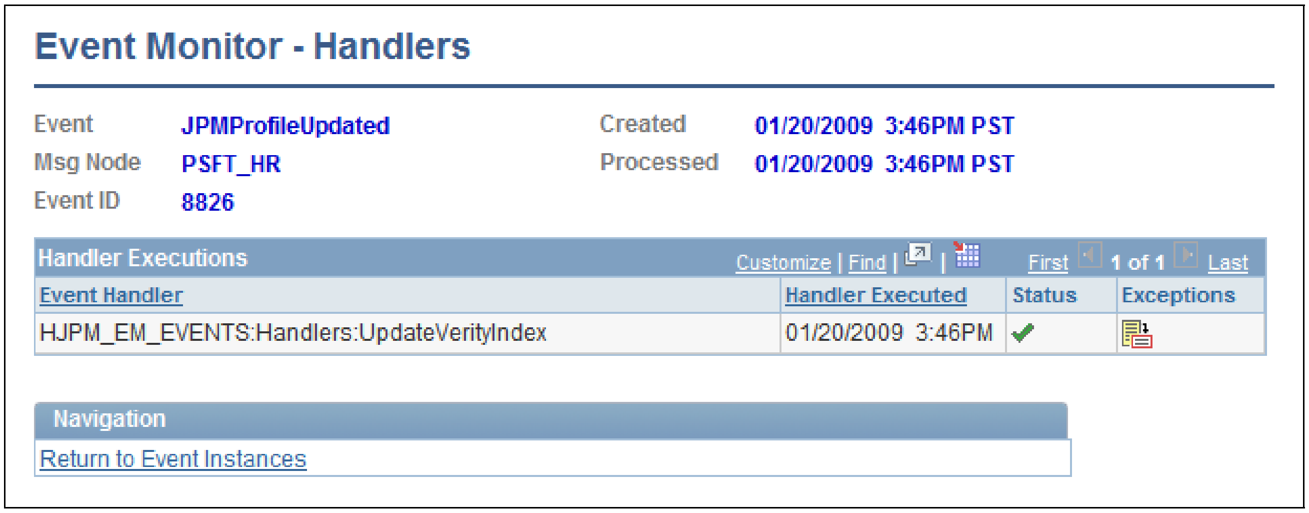
Event Instances

Review the logged event instances that meet your search criteria. The system displays instances for only the events and event handlers that you have configured in the local Event Registry component to be logged. You set the logging option for an event by selecting the Logging Enabled check box for an event on the Define Events page. Note that the process that raises the event waits for all registered synchronous event handlers to execute.

Node	The system displays the node from which the event instance was raised.
Event ID	The identification code of the event. The combination of this value and the node value uniquely identify the event instance on the originating node.
Event Name	The event name.
Event Created	The date and time that the event was raised in the source database (node). The system displays the time zone for this field because events could be raised in a node that operates in a time zone that differs from the local node.
User ID	The ID of the user that raised the event. This field is applicable only to event instances that are raised in the local database.
Event Data	The data for the event instance. To view the data, you must select the Display Event Data check box.
Handlers	Click this link for a specific event instance to access the Event Monitor - Handlers page, where you can view information about the registered event handlers that were executed on the local node for the selected event instance.

Reviewing Event Handler Executions

Access the Event Monitor - Handlers page (click the Handlers link for an event instance on the Event Manager - Monitor page).



Event Monitor - Handlers

Event	JPMProfileUpdated	Created	01/20/2009 3:46PM PST
Msg Node	PSFT_HR	Processed	01/20/2009 3:46PM PST
Event ID	8826		

Handler Executions — Customize | Find | First 1 of 1 Last

Event Handler	Handler Executed	Status	Exceptions
HJPM_EM_EVENTS:Handlers:UpdateVerityIndex	01/20/2009 3:46PM	✓	

Navigation

Return to Event Instances

Event Monitor - Handlers page

The system displays information for only the event handlers that you have configured to log executions. To log event handler executions, you must set the Logging option for a specific event handler and event combination on the Registered Handlers page.

The upper portion of the page displays information about the selected event instance that caused the event handler execution as well as the date and time that the local node began processing the event. For local events, the Processed date and time is generally the same as the date and time that the event was created. For remote events, the Processed date and time indicates when the local node received notification of the remote event from PeopleTools Integration Broker.

Handler Execution

This grid lists the event handlers that the system executed on the local database in response to the selected event instance.

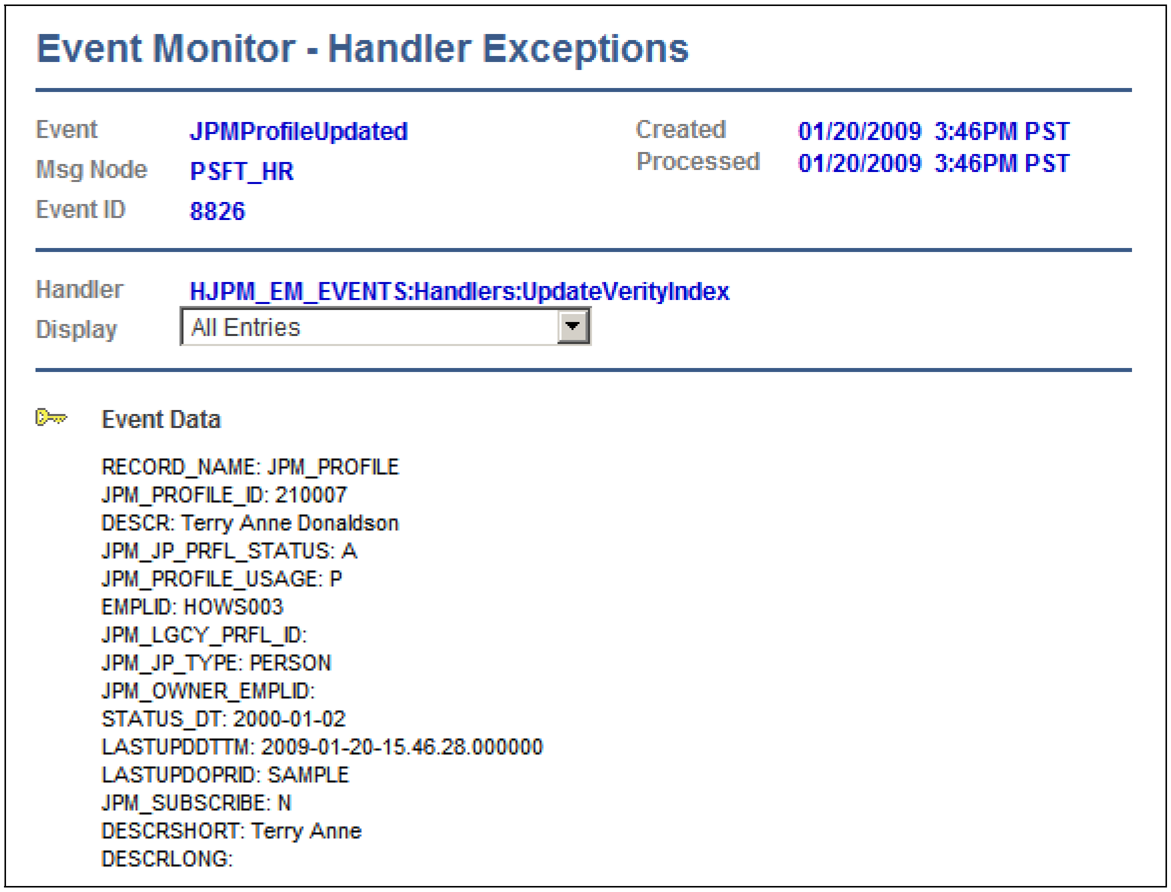
If no event handler execution information is available, the system displays an informative message that indicates the possible reasons why no information is available. Possible reasons are that no event handlers are registered for the selected event, no event handlers are configured to log execution for the selected event, no synchronous event handlers are registered for the selected event and PeopleTools Integration Broker has not yet executed any asynchronous event handlers, or no event handlers executed for the event.

Event Handler	The system displays the full path of the application class that implements the event handler.
Handler Executed	The system displays the date and time that the system executed the event handler.
🔑	The system displays the event instance data.
ℹ	The system displays information about the event handler exceptions.
⚠	The system displays the event handler exception warnings.
✖	The system displays the event handler exception errors.

The system displays event-handler trace information. To view trace information, you must select the Trace check box on the Registered Handlers page for this event handler and event combination.

Reviewing Event Handler Exceptions

Access the Event Monitor - Handler Exceptions page (click the Display Exceptions button on the Event Monitor - Handlers page).



Event Monitor - Handler Exceptions

Event	JPMProfileUpdated	Created	01/20/2009 3:46PM PST
Msg Node	PSFT_HR	Processed	01/20/2009 3:46PM PST
Event ID	8826		

Handler HJPM_EM_EVENTS:Handlers:UpdateVerityIndex

Display All Entries

Event Data

```
RECORD_NAME: JPM_PROFILE
JPM_PROFILE_ID: 210007
DESCR: Terry Anne Donaldson
JPM_JP_PRFL_STATUS: A
JPM_PROFILE_USAGE: P
EMPLID: HOWS003
JPM_LGCY_PRFL_ID:
JPM_JP_TYPE: PERSON
JPM_OWNER_EMPLID:
STATUS_DT: 2000-01-02
LASTUPDDTTM: 2009-01-20-15.46.28.000000
LASTUPDOPRID: SAMPLE
JPM_SUBSCRIBE: N
DESCRSHORT: Terry Anne
DESCRLONG:
```

Event Monitor - Handler Exceptions page

The upper portion of the page displays information about the selected event instance that caused the event handler execution as well as the date and time that the local node began processing the event.

Events raised in a remote database are visible in the Event Monitor component only after the local node receives and processes the asynchronous event message, and only if you have enabled logging for the event when registering the event in the local database through the Event Registry component.

Note. If you inactivate the PeopleTools Integration Broker Service Operation for the generic event message, then the local Event Manager framework does not process any remote events and thus no remote events will be visible in the Event Monitor component.

Handler	The system displays the full path of the application class that implements the event handler.
Display	Select a value to filter the type of information that displays on the page. Possible values are *All Entries*,*Exceptions*,*Event Data*, and *Trace Entries.* The system displays the information that is produced by the event handler in the order in which the event handler produced the information.
🔑	The system displays the event instance data.
ℹ	The system displays information about the event handler exceptions.
⚠	The system displays the event handler exception warnings.
✖	The system displays the event handler exception errors.
📝	The system displays event-handler trace information. To view trace information, you must select the Trace check box on the Registered Handlers page for this event handler and event combination.

Viewing Event Summaries

This section discusses how to:

- View a summary of event instances.
- View failed event handlers.

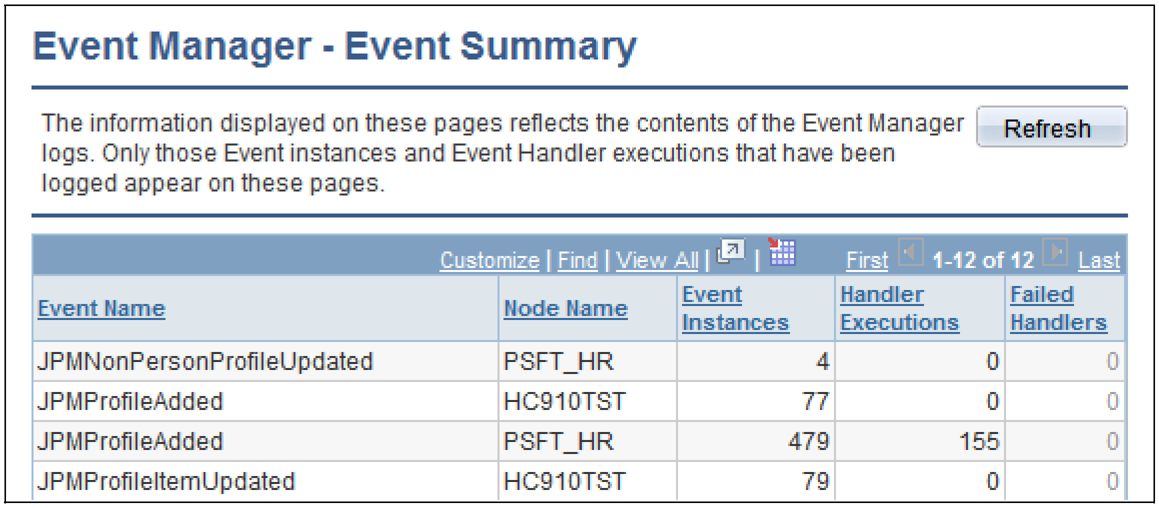
Pages Used View Event Summaries

Page Name	***Definition Name***	***Navigation***	***Usage***
Event Manager - Event Summary	EOEN_EVENT_SUM	Enterprise Components, Events & Notifications, Event Framework, Event Summary	View a summary of all the logged event instances in the local database. This is useful for quickly identifying problems, such as researching the cause of failed event-handler executions to determine a resolution.
Event Manager - Failed Handlers	EOEN_EVENT_SUM2	Click the Failed Handlers link for an event handler on the Event Manager - Event Summary page.	View a list failed event handlers across all instances of a selected event and node combination.

Page Name	***Definition Name***	***Navigation***	***Usage***
Event Monitor - Handler Exceptions	EOEN_EVENT_MON3	Click the Details button for a failed event handler on the Event Manager - Failed Handlers page.	View the exceptions that failed event handlers produced.

Viewing a Summary of Event Instances

Access the Event Manager - Event Summary page (Enterprise Components, Events & Notifications, Event Framework, Event Summary).


Event Manager - Event Summary

The information displayed on these pages reflects the contents of the Event Manager logs. Only those Event instances and Event Handler executions that have been logged appear on these pages.

Event Name	Node Name	Event Instances	Handler Executions	Failed Handlers
JPMNonPersonProfileUpdated	PSFT_HR	4	0	0
JPMProfileAdded	HC910TST	77	0	0
JPMProfileAdded	PSFT_HR	479	155	0
JPMProfileItemUpdated	HC910TST	79	0	0

Event Manager - Event Summary page

Refresh Click to refresh the statistics that the system displays on this page. The option to refresh is useful in production environments in which the raising and processing of events occurs frequently.

Summary Grid

Use this group box to review statistics for every event that has logged instances in the local database. The system displays event instances for only the events that are configured for logging through the Event Registry component in the local database. The system groups the logged event instances by event and node combination.

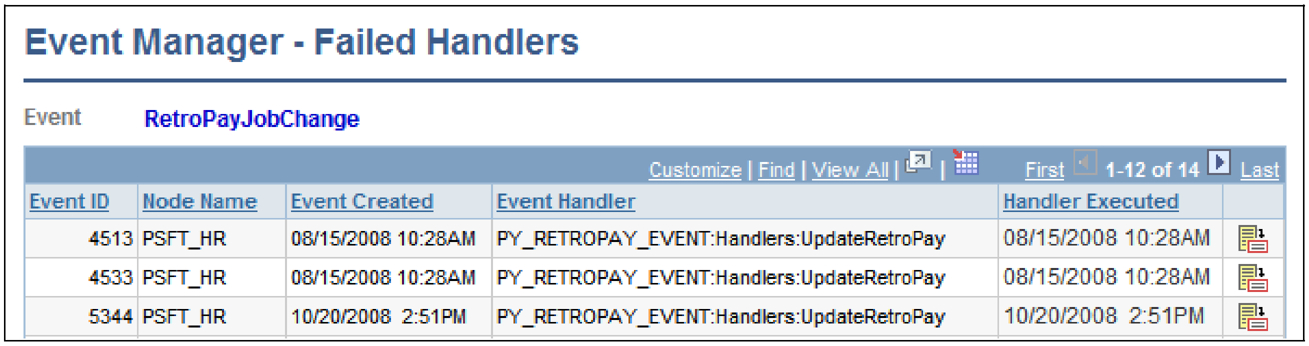
Event Name The system displays the name of the event for which the system logged event instances for the event and node combination.

Node Name The name of the originating node from which the system raised the event instances for the event and node combination.

Event Instances	The number of event instances that the system logged for the event and node combination. This equates to the number of times that the system raised the event.
Handler Executions	The total number of event handlers that the system has executed for the corresponding event and node combination.
Failed Handlers	The total number of event handlers that the system executed for the corresponding event and node combination that returned a failed status. The system displays this number as a link. Click the link to access the Event Manager - Failed Handlers page, where you can view a list of failed event handlers across all instances of the event and node combination. From that page you can view the exceptions that the failed event handlers produced.

Viewing Failed Event Handlers

Access the Event Manager - Failed Handlers page (click the Failed Handlers link for an event handler on the Event Manager - Event Summary page).


Event Manager - Failed Handlers

Event RetroPayJobChange

Customize | Find | View All | First 1-12 of 14 Last

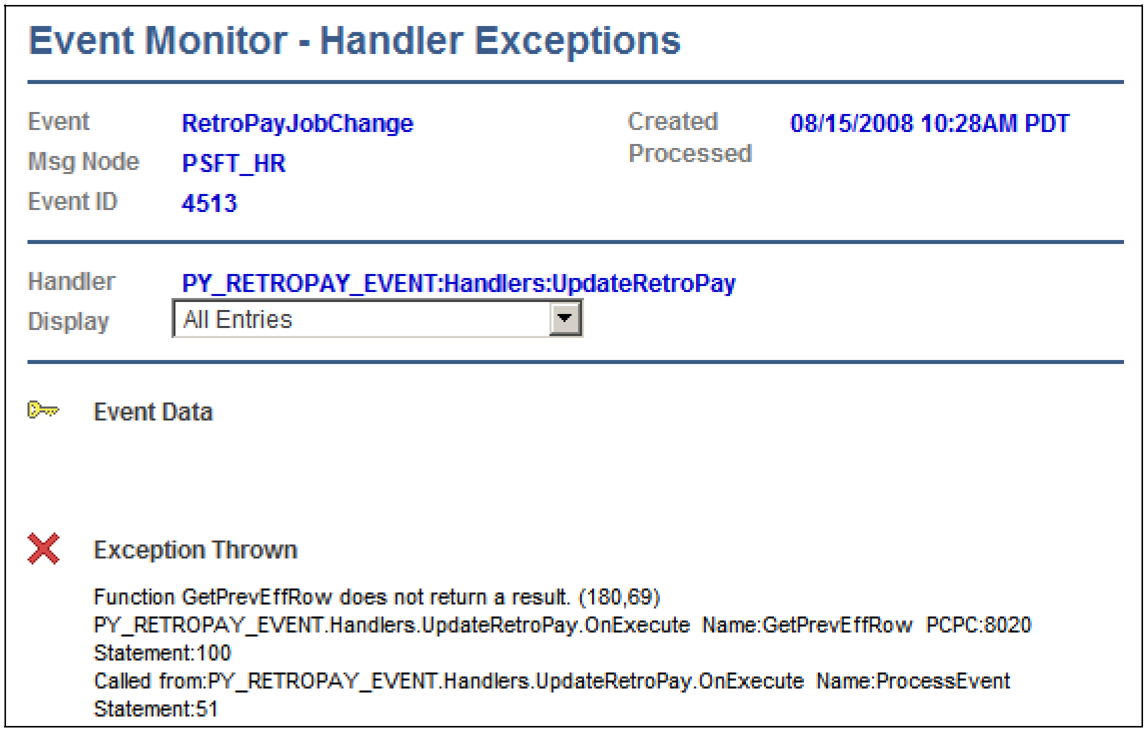
Event ID	Node Name	Event Created	Event Handler	Handler Executed	
4513	PSFT_HR	08/15/2008 10:28AM	PY_RETROPAY_EVENT:Handlers:UpdateRetroPay	08/15/2008 10:28AM	
4533	PSFT_HR	08/15/2008 10:28AM	PY_RETROPAY_EVENT:Handlers:UpdateRetroPay	08/15/2008 10:28AM	
5344	PSFT_HR	10/20/2008 2:51PM	PY_RETROPAY_EVENT:Handlers:UpdateRetroPay	10/20/2008 2:51PM	

Event Manager - Failed Handlers page

The system lists information about each failed event handler. Click the Details button to access the Event Monitor - Handler Exceptions page, where you can view the exceptions that the failed event handlers produced.

Viewing Exceptions for Failed Handlers

Access the Event Manager - Handler Exceptions page (click the Details link for an event handler on the Event Manager - Failed Handlers page).



Event Monitor - Handler Exceptions

Event	RetroPayJobChange	Created Processed	08/15/2008 10:28AM PDT
Msg Node	PSFT_HR		
Event ID	4513		

Handler PY_RETROPAY_EVENT:Handlers:UpdateRetroPay

Display All Entries

Exception Thrown

```
Function GetPrevEffRow does not return a result. (180,69)
PY_RETROPAY_EVENT.Handlers.UpdateRetroPay.OnExecute  Name:GetPrevEffRow  PCPC:8020
Statement:100
Called from:PY_RETROPAY_EVENT.Handlers.UpdateRetroPay.OnExecute  Name:ProcessEvent
Statement:51
```

Event Monitor-Handler Exceptions page

Details for the exception are displayed.

Chapter 5

Creating Alerts

This chapter discusses how to:

- Define PeopleSoft Queries in the Alert Setup Component
- Run the Alerts Process

Understanding PeopleSoft Queries Within the Alert Framework

PeopleSoft Query can be used in combination with the Events and Notifications Framework feature to generate alert messages based on data within one or more PeopleSoft tables.

Using PeopleSoft Query, you can scan one or more PeopleSoft tables and then generate an alert message for any problems or stalled transactions. The steps to set up alert notifications based on a PeopleSoft Query are:

1. Write the PeopleSoft Query.
2. Use the Notification Registry page to add the new notification. Add the Process Name and Process Category that identifies this specific query/notification. Add a description of the new notification.
3. Use the Alert Setup component to define the values to be included in the alert message. The message object data is defined by the PeopleSoft Query name, the Process Name, and the Process Category. The Alert Setup component contains the values used to build the notifications to the Notification Dashboard and email.
4. Enter the Notification Registry page again to enable overrides. You must select the override check boxes. The values used to build the notifications to the PeopleSoft Worklist and XML messages are defined at the System-Level Notifications page or the BU-Level Notifications page.
5. Use the System-Level Notifications page to define values used to build the notifications to the PeopleSoft Worklist and XML messages. These values determine who should be notified (user role and node) and how they should be notified (worklist and XML notification). These values are applied for the process name and process category combination to the entire PeopleSoft system. If you enter values for email notification, the entries on the Alert Setup component override these values.
6. Use the BU-Level Notifications page to define values used to build the notifications to the PeopleSoft Worklist and XML messages. These values determine who should be notified (user role and node) and how they should be notified (worklist and XML notification). These overrides are applied for the process name, process category, and business unit combination. Values at the business unit level override any system-level overrides. If you enter values for email notification, the entries on the Alert Setup component override these values.
7. For XML notifications, setup and activate the EOEN Notify Message to enable the PeopleSoft Integration Broker to send a message to a third-party system using a XML-formatted messages.

8. Setup the Alerts process run control to query the PeopleSoft tables on a regular schedule.

Defining PeopleSoft Queries in the Alert Setup Component

When using PeopleSoft Query with the alert framework feature you must setup the Alert Setup component. The values defined on the Alert Setup component are used to build the notifications sent to the Notification Dashboard and to email. The values are defined by the PeopleSoft Query name, the Process Name, and the Process Category. The values include:

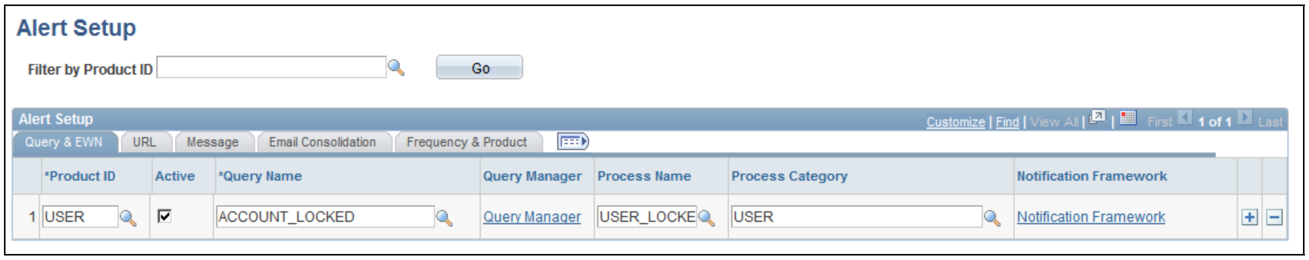
- The message set and message number.
- User IDs and email addresses needed to route the alert message to the correct location.
- Links to the PeopleSoft page where the problem can be viewed or resolved.
- Consolidation of email notifications.
- Frequency and content of notifications.

Pages Used to Define PeopleSoft Queries in the Alert Setup Component

Page Name	***Definition Name***	***Navigation***	***Usage***
Alert Setup	EOEN_ALERT_SETUP	Enterprise Components, Events and Notifications, Alerts, Alert Setup	Enter the alert framework notification methods and recipients of the alert messages when using one or more PeopleSoft Queries with the alert framework feature.
Alert Setup	EOEN_ALERT_SETUP	Select the Messages tab on the Alert Setup page.	Enter search keys for the destination page defined in the link from the alert message.
Alert Setup-Message Parameters	EOEN_ALERT_SETUP	Select the Message tab on the Alert Setup page. Click the Message Parameters link.	Enter the sequence numbers and field names (parameters) for the message set and message number included with the alert notification.

Setting Up the Query & EWN Tab of the Alert Setup Page

Access the Alert Setup-Query & EWN page (Enterprise Components, Events and Notifications, Alerts, Alert Setup).



Alert Setup

Filter by Product ID

Go

Alert Setup

Customize | Find | View All | First 1 of 1 Last

Query & EWN | URL | Message | Email Consolidation | Frequency & Product

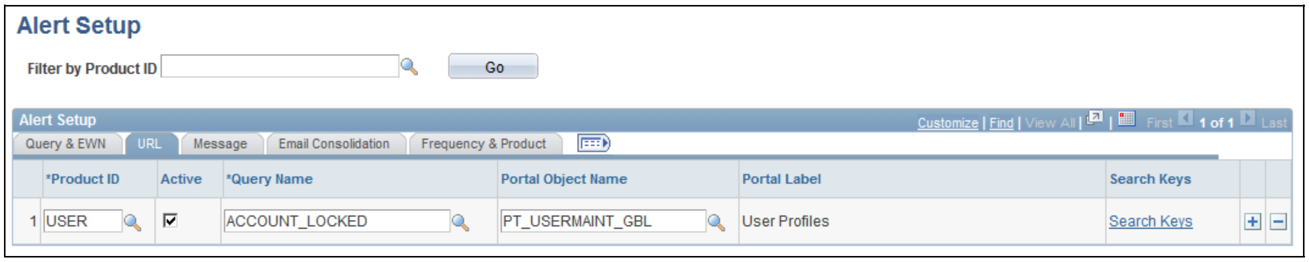
	*Product ID	Active	*Query Name	Query Manager	Process Name	Process Category	Notification Framework
1	USER	✓	ACCOUNT_LOCKED	Query Manager	USER_LOCKE	USER	Notification Framework

Alert Setup-Query&EWN page

Active	Select this check box to activate this PeopleSoft query within the notification feature.
Query Name	(required) Specify a query that is defined in the PeopleSoft Query Manager. Insert additional rows to this component to define additional queries to be used for notification.
Query Manager	Select this link to go to the PeopleSoft Query Manager where you can view a PeopleSoft query.
Process Name	(required) Enter or view the process for the notifications. The notification is sent to the user based on the setup for the Process Name and Process Category combination.
Process Category	(required) Enter or view the category for the notifications. The category is a logical partition of the Process Name. The notification is sent to the user based on the setup for the Process Name and Process Category combination. The Process Category field is blanked out when the Process Name is changed.
Notification Framework	Select this link to access the Notification framework setup pages. If business unit-level setup exists for the Process Name and Process Category combination, then the BU-Level Notifications page is displayed; otherwise the System-Level Notifications page is displayed. If the Process Name is blank or both Process Name and Process Category fields are blank then the Notification Registry page is displayed. See Chapter 3, "Setting Up Notifications," page 11.

Setting Up the URL Tab of the Alert Setup Page

Access the Alert Setup-URL page (Enterprise Components, Events & Notifications, Alerts, Alert Setup, select the URL tab):


Alert Setup

Filter by Product ID

*Product ID	Active	*Query Name	Portal Object Name	Portal Label	Search Keys
USER	✓	ACCOUNT_LOCKED	PT_USERMAINT_GBL	User Profiles	Search Keys

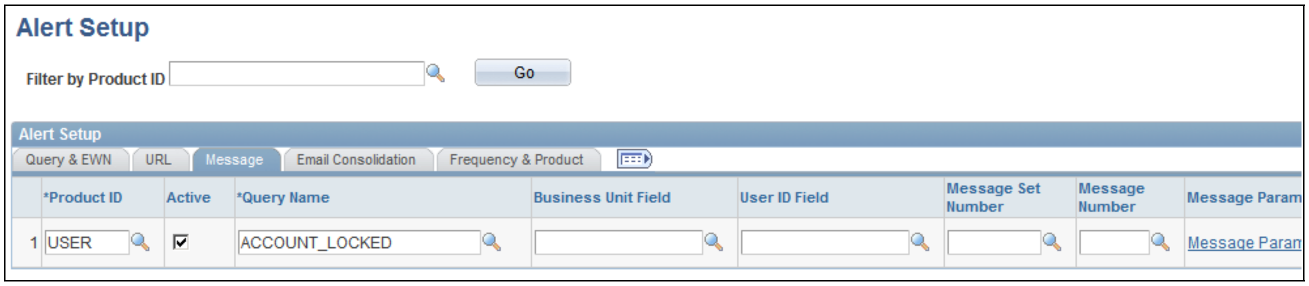
Alert Setup-URL page

Use the Alert Setup-URL page to provide a link from the notification message to the PeopleSoft page where you can resolve issue.

Query Name	Identifies the query for this notification.
Portal Object Name	Enter a portal object name to define the URL linking the notification message to the transaction page where the user can resolve the issue discovered by the query. This URL is placed on the notification message on the Notification Dashboard page as well as in the email to the user.
Search Keys	Select this link to access the Alert Setup-Search Keys page where you can enter search keys or parameters for the transaction page. An error message is displayed if the Portal Object Name field is blank when trying to launch this secondary page.

Setting Up the Message Tab of the Alert Setup Page

Access the Alert Setup- Message page (Enterprise Components, Events & Notifications, Alerts, Alert Setup, select the Message tab):


Alert Setup

Filter by Product ID

*Product ID	Active	*Query Name	Business Unit Field	User ID Field	Message Set Number	Message Number	Message Param
USER	✓	ACCOUNT_LOCKED					Message Param

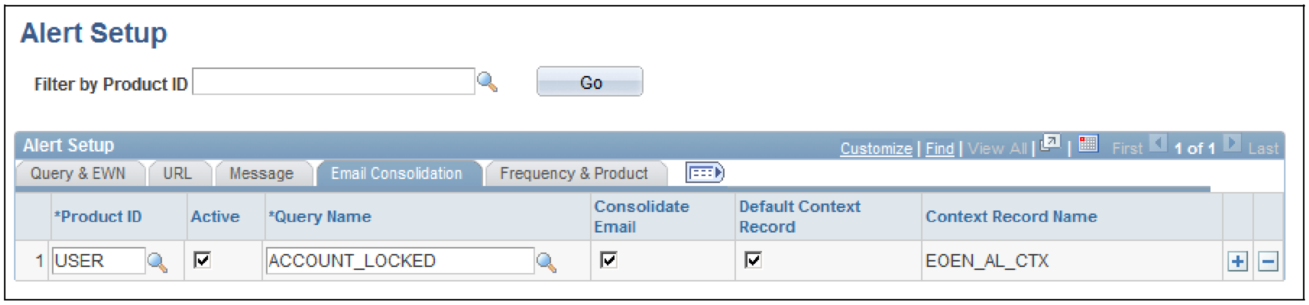
Alert Setup-Message page

Business Unit Field	Specify which field from the query should be used to map the business unit level setup of the notifications. A warning message is issued if this field is blank and the BU-Level Notifications page for this process name/process category is populated. The prompt list for this field is the output fields of the query.

User ID Field	For email notifications, specify which field from the PeopleSoft query has the user ID. If a user ID is entered in this field and you have enabled email notifications, then the email address is derived from the user's profile. The hierarchy for determining which email address to use is as follows: 1. The User ID from the Alert Setup- Message page. If none is found, then: 2. TheUser ID from the BU-Level Notifications page. If none is found, then: 3. The User ID from the System-Level Notifications page.
Message Set Number and **Message Number**	For email notifications, specify which message defined in the PeopleSoft Message Catalog is used in the notification message.
Message Parameters	Select the link to access the Alert Setup-Message Parameters page where you can enter the sequence numbers and field names (parameters) for the message. If the Message Set Number field or the Message Number field is blank then you cannot access the Alert Setup-Message Parameters page and an error message is displayed.

Setting Up the Email Consolidation Tab of the Alert Setup Page

Access the Alert Setup-Email Consolidation page (Enterprise Components, Events & Notifications, Alerts, Alert Setup, select the Email Consolidation tab):



	*Product ID	Active	*Query Name	Consolidate Email	Default Context Record	Context Record Name
1	USER	✓	ACCOUNT_LOCKED	✓	✓	EOEN_AL_CTX

Alert Setup-Email Consolidation page

Consolidate Email	Select this check box to consolidate email notifications from this PeopleSoft query. When this check box is selected, each email address receive one consolidated email notification per query. If this check box is not selected, then each email address receives one email per transaction or problem found.

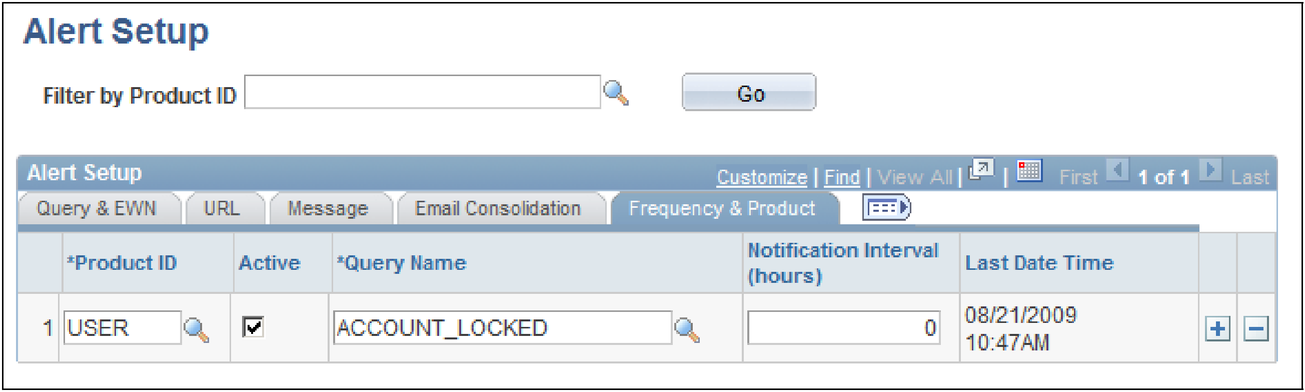
Default Context Record Select this check box to use the default context record in the notifications. A context record holds transaction-related information. This information can be accessed in the Notification Dashboard page. You can choose to use the default context record or add a user-defined context record.

If this check box is selected the Context Record field becomes display-only and is populated with the default context record name, *EOEN_AL_CTX.*

If this check box is un-selected, the Context Record field becomes available for input and you can choose a user-defined context record. EOEN_LOG_KEY and EOEN_AL_CTX_SBR subrecords are required for user-defined context records. The prompt list for the Context Record field is the records that have EOEN_LOG_KEY and EOEN_AL_CTX_SBR subrecords.

Setting Up the Frequency & Product Tab of the Alert Setup Page

Access the Alert Setup- Frequency & Product page (Enterprise Components, Events & Notifications, Alerts, Alert Setup, select the Frequency & Product tab):



	*Product ID	Active	*Query Name	Notification Interval (hours)	Last Date Time
1	USER	✓	ACCOUNT_LOCKED	0	08/21/2009 10:47AM

Alert Setup-Frequency & Product page

Notification Interval (hours) Specify the number of hours between notifications. The Alerts process checks the time interval and the last run date/time stamp to determine whether the notification should be sent.

For example without the notification interval feature if the Alert process is run every hour using a query to check for unpaid invoices and the user cannot resolve the issue within one hour, then a repeat notification is issued for the same unpaid invoices. These repeat notifications on the same problems can crowd up a user's email account. If we add a notification interval of 4 hours and the Alert process is run every hour, then the Alert process checks the last run date/time stamp and only sends the notification if 4 hours have past since the last notification.

Note. Interval does not limit dashboard, worklist and XML notifications. These will always be created irregardless of the interval setting.

Product ID Identify the PeopleSoft application of the query. This field enables the Alerts process page to run a group of queries for a particular PeopleSoft product.

Running the Alerts Process

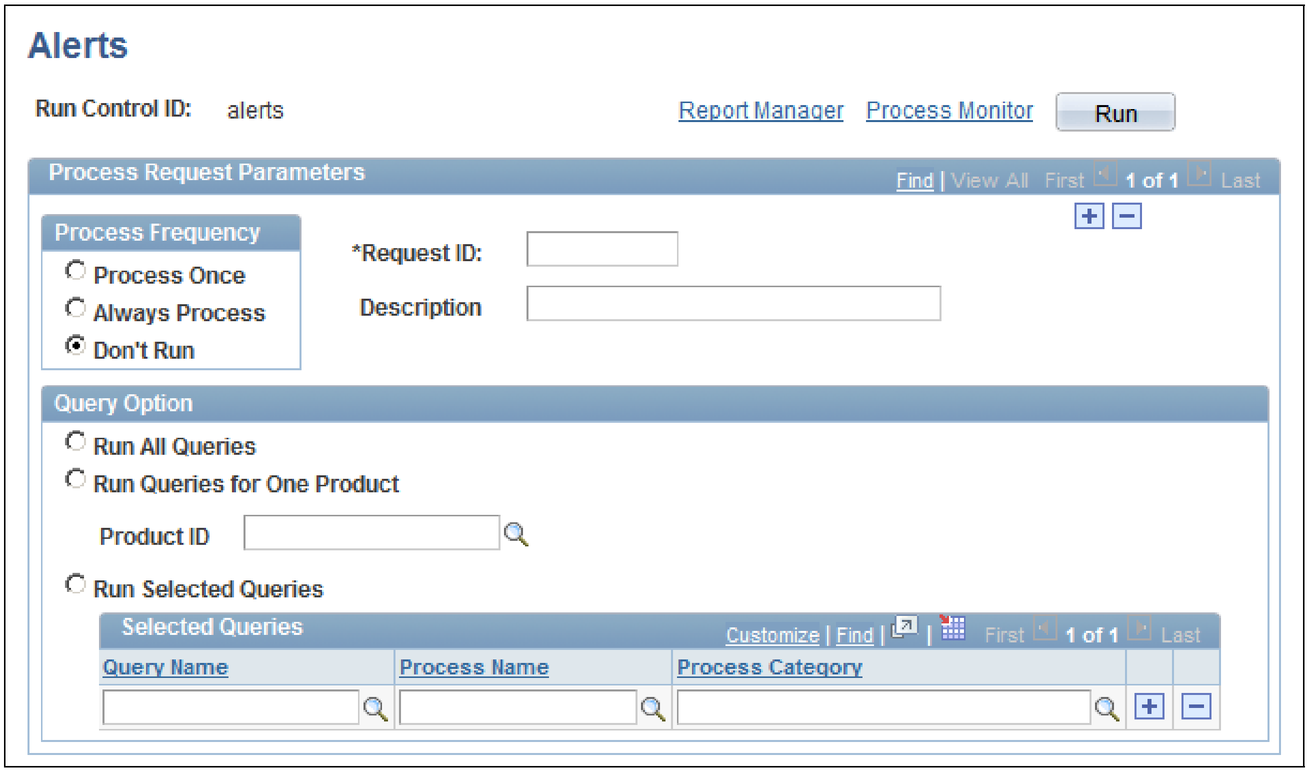
This section discusses how to setup the Alerts process page to query the PeopleSoft tables on a regular schedule and generate any necessary alert messages.

Page Used to Run the Alerts Process

Page Name	***Definition Name***	***Navigation***	***Usage***
Alerts	RUN_EOEN_ALERT	Enterprise Components, Events & notifications, Alerts, Run Alerts	Enter the run control parameters for the EOEN_ALERT process. This process generates the alert messages from the PeopleSoft queries.

Setting Up the Alerts Process Page

Access the Alerts process page (Enterprise Components, Events & notifications, Alerts, Run Alerts):



Alerts

Run Control ID: alerts

Report Manager | Process Monitor | Run

Process Request Parameters — Find | View All First 1 of 1 Last

Process Frequency
- Process Once
- Always Process
- [✓ Don't Run]

*Request ID:

Description

Query Option
- Run All Queries
- Run Queries for One Product

Product ID

- Run Selected Queries

Selected Queries — Customize | Find | First 1 of 1 Last

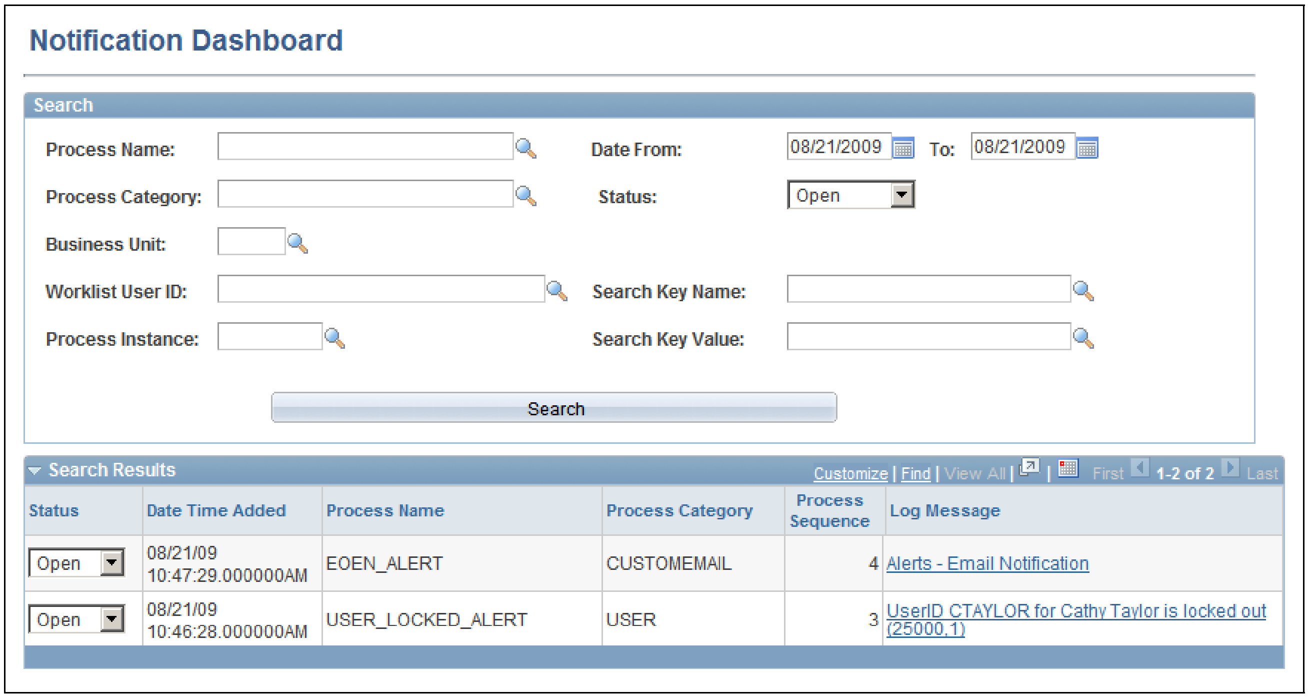
Query Name	Process Name	Process Category

Alerts run control page

Use the EOEN_ALERT process to run the PeopleSoft queries and generate the alert messages based on these queries.

Query Option	Select the PeopleSoft queries to be run. The options are: • *Run All Queries:* Select to run all PeopleSoft queries defined for the alert notification. When you select this button, the PeopleSoft Product ID field and the Selected Queries group box are disabled. • *Run Queries for One Product:* Select to run the PeopleSoft queries defined for one PeopleSoft application. Selecting this button, enables a PeopleSoft Product ID field to the right where you can select the PeopleSoft applications defined on the Alert Setup component on the Frequency & Product tab. • *Run Selected Queries:* Select to run the PeopleSoft queries defined in the Selected Queries group box at the bottom of this page.
Query Name	Identifies one or more PeopleSoft queries linked to the alert notification. This column is used by the Alerts process if the Run Selected Queries button is selected.
Process Name	Enter the process defined for the PeopleSoft query on the Alert Setup component.
Process Category	Enter the category defined for the PeopleSoft query on the Alert Setup component. The category is a logical partition of the Process Name.

You can view alerts in the Notification Dashboard (select Enterprise Components, Events & Notifications, Notification Framework, Notification Dashboard)



Notification Dashboard

Search

Process Name:		Date From:	08/21/2009 To: 08/21/2009
Process Category:		Status:	Open
Business Unit:			
Worklist User ID:		Search Key Name:	
Process Instance:		Search Key Value:	

Search Results Customize | Find | View All | First 1-2 of 2 Last

Status	Date Time Added	Process Name	Process Category	Process Sequence	Log Message
Open	08/21/09 10:47:29.000000AM	EOEN_ALERT	CUSTOMEMAIL	4	Alerts - Email Notification
Open	08/21/09 10:46:28.000000AM	USER_LOCKED_ALERT	USER	3	UserID CTAYLOR for Cathy Taylor is locked out (25000,1)

Notification Dashboard showing Alerts

Chapter 6

Using the Notification Dashboard

This chapter discusses how to:

- Search for a message.
- View message details.

Searching and Viewing the Notification Dashboard

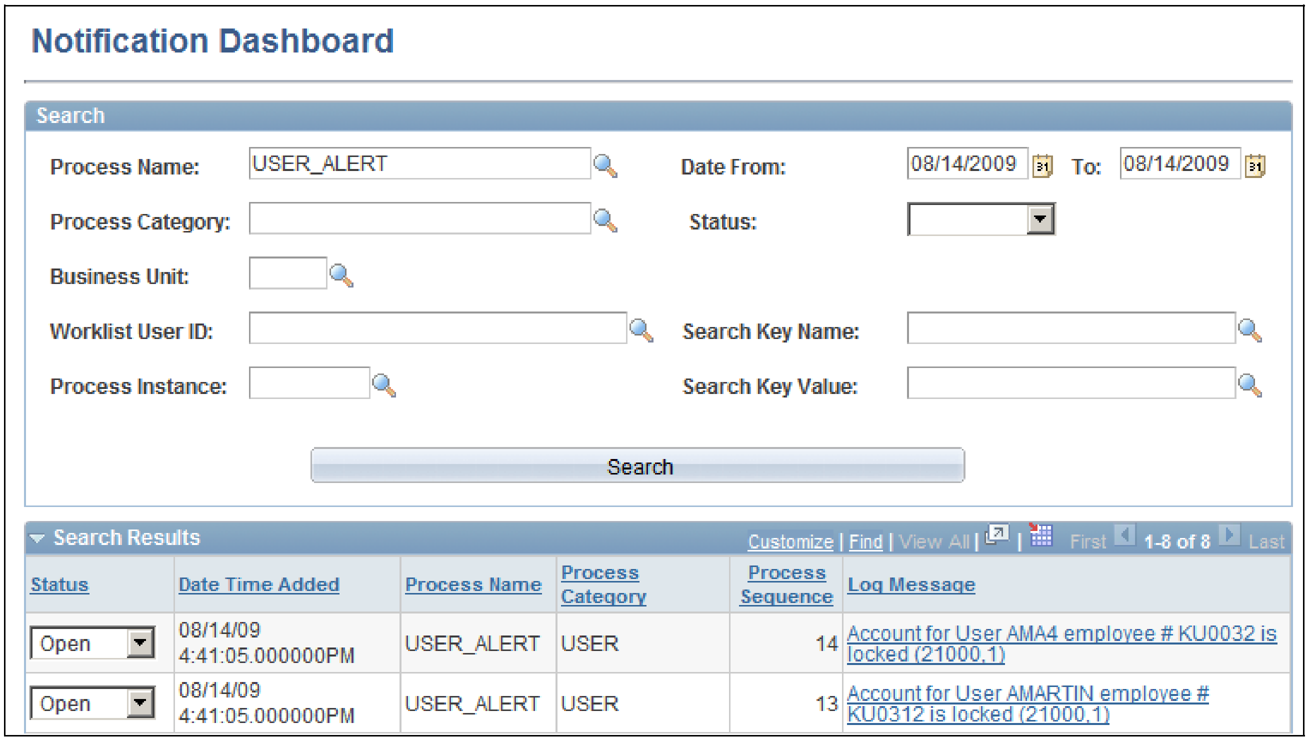
The Notification Dashboard is a central location to view and resolve errors, warnings, and notifications generated by the Events & Notification framework.

Pages Used to Search and View the Notification Dashboard

Page Name	*Definition Name*	*Navigation*	*Usage*
Notification Dashboard	EOEN_NTF_DSH_SRCH	Enterprise Components, Events and Notifications, Notifications Framework, Notifications Dashboard	Search for notification messages that were created from a PeopleSoft process or a PeopleSoft Query.
Message Details	EOEN_NTF_DASH_DTL	Enterprise Components, Events and Notifications, Notifications Framework, Notifications Dashboard	View the details of a notification message that was displayed on the Notification Dashboard page.

Searching for a Message

Access the Notification Dashboard page (Enterprise Components, Events and Notifications, Notifications Framework, Notifications Dashboard).


Notification Dashboard

Search

Process Name:	USER_ALERT	Date From:	08/14/2009 To: 08/14/2009
Process Category:		Status:	
Business Unit:			
Worklist User ID:		Search Key Name:	
Process Instance:		Search Key Value:	

Search Results

Status	Date Time Added	Process Name	Process Category	Process Sequence	Log Message
Open	08/14/09 4:41:05.000000PM	USER_ALERT	USER	14	Account for User AMA4 employee # KU0032 is locked (21000,1)
Open	08/14/09 4:41:05.000000PM	USER_ALERT	USER	13	Account for User AMARTIN employee # KU0312 is locked (21000,1)

Notification Dashboard search page

The Notification Dashboard page is used to search for notification messages based on selected search criteria. Select the link in the Log Message column to access the Message Detail page where you can see further details and access the specific PeopleSoft page for correction of the issue.

Selection Criteria group box

The Selection Criteria group box displays the search criteria that you can enter to narrow the display of notification messages.

Process Name	Enter the process that generated the notification messages that you wish to view. The process name can be a PeopleSoft process or a user-defined name for a PeopleSoft query.
From Date and **To**	Enter the date range for the notification messages that you wish to view. The date of the notification message is the date that the PeopleSoft process was run.
Process Category	Enter the process category that generated the notification messages that you wish to view. The category is a logical partition of the Process Name. The notification messages are sent to the Notification Dashboard based on the Process Name and Process Category combination.

Status	Enter the status of the notification messages that you wish to view. The options are: • *Open:* Indicates that the notification message is still unresolved and has not been set to the closed status. • *Closed:* Indicates that the notification message has been resolved and set to the closed status using the Status field in the Search Results group box of this page or the Message Detail page.
Business Unit	Enter a PeopleSoft business unit to narrow your search results to notification messages related to one business unit.
Worklist User ID	Enter a PeopleSoft User ID to limit the search results to the notification messages that would appear on a specific PeopleSoft Worklist.
Process Instance	Enter the process instance number of a single PeopleSoft process. This enables you to view just one specific process run. You can locate the instance number in the Process Monitor component.
Search Key Name	Enter search keys or parameters for the transaction or problem in the notification message.
Search Key Value	Enter the value within the Search Key Name field.
Search	Click this button to initiate the search based on your search criteria. The results display in the Search Results group box.

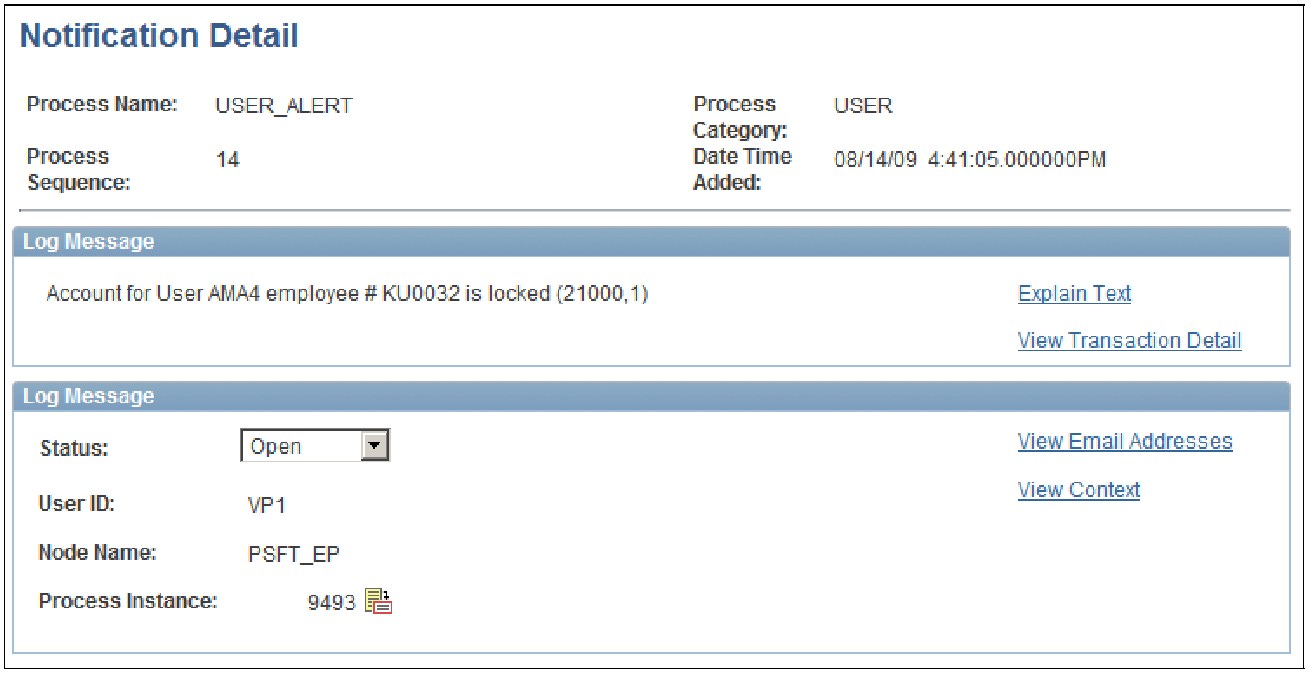
Search Results group box

The Search Results group box displays all notification messages that meet the search criteria after you click the Search button.

Status	Displays the current status of the notification message. Change to *Closed* when the issue is resolved.
Log Message	Displays the message of the notification message. Click this link to access the Message Details page where you can view additional information about this specific notification message.

Viewing Notification Details

Access the Message Detail page (Enterprise Components, Events and Notifications, Notifications Framework, Notifications Dashboard. Click the Message Details link).


Notification Detail

Process Name:	USER_ALERT	Process Category:	USER
Process Sequence:	14	Date Time Added:	08/14/09 4:41:05.000000PM

Log Message

Account for User AMA4 employee # KU0032 is locked (21000,1)

Explain Text

View Transaction Detail

Log Message

Status:	Open
User ID:	VP1
Node Name:	PSFT_EP
Process Instance:	9493

View Email Addresses

View Context

Notification Detail page

The Message Detail page provides the user with valuable information for responding to the message. The message includes valuable links.

View Transaction Detail	Select this link to access the specific PeopleSoft page for correction of the message. For PeopleSoft processes, this link is dependent on the employee servlet setup on the URL Maintenance page. For PeopleSoft queries, this link is dependent on the Alert Setup-URL page.
Status	Displays the current status of the message. Change to *Closed* when the issue is resolved.
View Email Address	Select this link to view the email addresses where the alert was sent.
View Context	Select this link to view key data from the message; including, process name, process category, process instance number, and the record name.

Chapter 7

Using PeopleSoft Worklists to View Notification Messages

This chapter discusses using the PeopleSoft Worklist to view notification messages.

Using the PeopleSoft Worklist to View Notification Messages

Worklists are prioritized lists of the work items that a person (or group of people) has to do. A worklist is a standard PeopleTools grid. Therefore, the user can use grid personalization features to order and sort columns. When work is routed to a PeopleSoft user, it is put in the user's worklist. To work on an item, the user selects it from the worklist and is presented with the appropriate page to begin work. Users accessing worklists through a browser see worklist entries prioritized in a predefined order (set on a properties page). For example, worklists for accounts receivable clerks can be sorted by days overdue, amount overdue, or credit class.

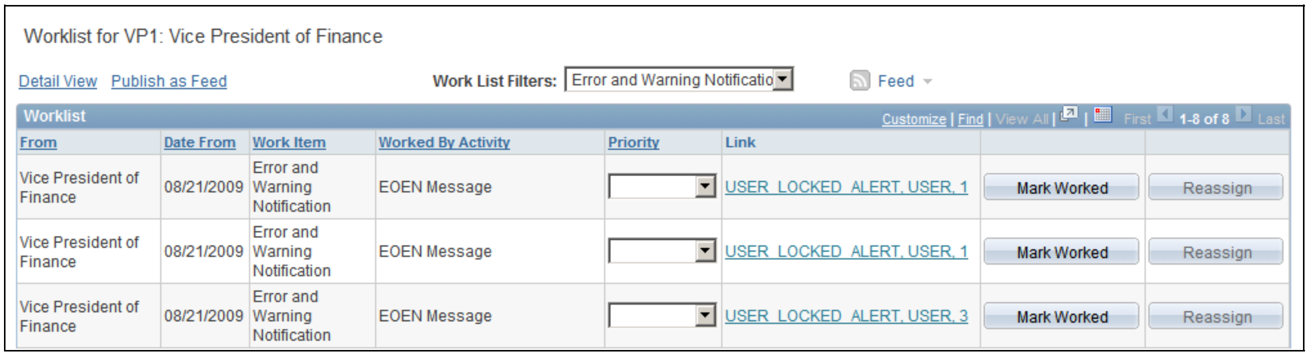
For more information about using PeopleSoft Worklists, see the *Enterprise PeopleTools 8.50 PeopleBook: Workflow Technology*.

Pages Used to View Notification Messages on a PeopleSoft Worklist

Page Name	***Definition Name***	***Navigation***	***Usage***
Worklist	WORKLIST	Worklist, Worklist	View, assign priority to, reassign, and process worklist items.

Using a PeopleSoft Worklist with Notification Messages

Access the Worklist page (Worklist, Worklist):


Worklist for VP1: Vice President of Finance
Detail View
Publish as Feed
Work List Filters: Error and Warning Notificatio
Feed
Worklist
Customize | Find | View All |
First 1-8 of 8 Last

From	Date From	Work Item	Worked By Activity	Priority	Link		
Vice President of Finance	08/21/2009	Error and Warning Notification	EOEN Message		USER_LOCKED_ALERT, USER, 1	Mark Worked	Reassign
Vice President of Finance	08/21/2009	Error and Warning Notification	EOEN Message		USER_LOCKED_ALERT, USER, 1	Mark Worked	Reassign
Vice President of Finance	08/21/2009	Error and Warning Notification	EOEN Message		USER_LOCKED_ALERT, USER, 3	Mark Worked	Reassign

Worklist

Work List Filters	Select *Error and Warning Notification* in the field to view just your notification messages.
From	Displays the individual who triggered the work item.
Date From	Displays the date when the work item was triggered.
Link	Click to access the Message Detail page of the Notification Dashboard.
Mark Worked	Click this button to mark an item as worked if you have performed the necessary actions.

Index