Oracle® Approvals Management

Implementation Guide

Release 12.1

Part No. E13516-13

May 2018

ORACLE

Oracle Approvals Management Implementation Guide, Release 12.1

Part No. E13516-13

Primary Author: Gowri Arur

Contributing Author: V Suryanarayana Murthy Boggavara

Contents

3 Attributes

4 Conditions

5 Actions

Send Us Your Comments

Oracle Approvals Management Implementation Guide, Release 12.1

Part No. E13516-13

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document. Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

Note: Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Oracle E-Business Suite Release Online Documentation CD available on My Oracle Support and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: appsdoc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

Intended Audience

Welcome to Release 12.1 of the *Oracle Approvals Management Implementation Guide.*

This guide assumes you have a working knowledge of the following:

- The principles and customary practices of your business area.
- Oracle HRMS.

  If you have never used Oracle HRMS, Oracle suggests you attend one or more of the Oracle HRMS training classes available through Oracle University
- Oracle Self-Service Web Applications.
- The Oracle Applications graphical user interface.

  To learn more about the Oracle Applications graphical user interface, read the *Oracle E-Business Suite User's Guide*.

See Related Information Sources on page x for more Oracle E-Business Suite product information.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.

com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Structure

Related Information Sources

Oracle HRMS shares business and setup information with other Oracle Applications products. Therefore, you may want to refer to other user guides when you set up and use Oracle HRMS.

You can read the guides online by choosing Library from the expandable menu on your HTML help window, by reading from the Oracle Applications Document Library CD included in your media pack, or by using a Web browser with a URL that your system administrator provides.

If you require printed guides, you can purchase them from the Oracle store at http://oraclestore.oracle.com.

Guides Related to All Products

Oracle E-Business Suite User's Guide

This guide explains how to navigate, enter data, query, and run reports using the user interface (UI) of Oracle E-Business Suite. This guide also includes information on setting user profiles, as well as running and reviewing concurrent requests.

Guides Related to This Product

Oracle Daily Business Intelligence for HRMS User Guide

This guide describes the dashboards and reports available for HR Line Managers, Chief HR Officer, Budget Managers, and Benefits Managers using Daily Business Intelligence for HRMS. It includes information on using parameters, how DBI for HRMS derives values, and how to troubleshoot dashboards and reports.

Oracle Daily Business Intelligence for HRMS Implementation Guide

This guide provides basic setup procedures for implementing and maintaining HRMS-related dashboards.

Oracle Daily Business Intelligence Implementation Guide

This guide describes the common concepts for Daily Business Intelligence. It describes the product architecture and provides information on the common dimensions, security considerations, and data summarization flow. It includes a consolidated setup checklist by page and provides detailed information on how to set up, maintain, and troubleshoot Daily Business Intelligence pages and reports for the following functional areas: Financials, Interaction Center, iStore, Marketing, Product Lifecycle Management, Projects, Procurement, Sales, Service, Service Contracts, and Supply Chain.

Oracle Daily Business Intelligence User Guide

This guide describes the common concepts for Daily Business Intelligence. It describes the product architecture and provides information on the common dimensions, security considerations, and data summarization flow. It includes a consolidated setup checklist by page and provides detailed information on how to set up, maintain, and troubleshoot Daily Business Intelligence pages and reports for the following functional areas: Financials, Interaction Center, iStore, Marketing, Product Lifecycle Management, Projects, Procurement, Sales, Service, Service Contracts, and Supply Chain.

Oracle Application Framework Personalization Guide

Learn about the capabilities of the OA Framework technologies.

Oracle Human Resources Management Systems Enterprise and Workforce Management Guide

Learn how to use Oracle HRMS to represent your enterprise. This includes setting up your organization hierarchy, recording details about jobs and positions within your enterprise, defining person types to represent your workforce, and also how to manage your budgets and costs.

Oracle Human Resources Management Systems Workforce Sourcing, Deployment, and Talent Management Guide

Learn how to use Oracle HRMS to represent your workforce. This includes recruiting new workers, developing their careers, managing contingent workers, and reporting on your workforce.

Oracle Human Resources Management Systems Payroll Processing Management Guide

Learn about wage attachments, taxes and social insurance, the payroll run, and other processes.

Oracle Human Resources Management Systems Compensation and Benefits Management Guide

Learn how to use Oracle HRMS to manage your total compensation package. For example, read how to administer salaries and benefits, set up automated grade/step progression, and allocate salary budgets. You can also learn about setting up earnings and deductions for payroll processing, managing leave and absences, and reporting on compensation across your enterprise.

Oracle Human Resources Management Systems Configuring, Reporting, and System

Administration Guide

Learn about extending and configuring Oracle HRMS, managing security, auditing, information access, and letter generation.

Oracle Human Resources Management Systems Implementation Guide

Learn about the setup procedures you need to carry out in order to implement Oracle HRMS successfully in your enterprise.

Oracle Human Resources Management Systems FastFormula User Guide

Learn about the different uses of Oracle FastFormula, and understand the rules and techniques you should employ when defining and amending formulas for use with Oracle applications.

Oracle Self-Service Human Resources Deploy Self-Service Capability Guide

Set up and use self-service human resources (SSHR) functions for managers, HR Professionals, and employees.

Oracle Performance Management Implementation and User Guide

Learn how to set up and use performance management functions. This includes setting objectives, defining performance management plans, managing appraisals, and administering questionnaires.

Oracle Succession Planning Implementation and User Guide

Learn how to set up and use Succession Planning functions. This includes identifying succession-planning requirements, using talent profile, suitability analyzer, and performance matrices.

Oracle Human Resources Management Systems Deploy Strategic Reporting (HRMSi)

Implement and administer Oracle Human Resources Management Systems Intelligence (HRMSi) in your environment.

Oracle Human Resources Management Systems Strategic Reporting (HRMSi) User Guide

Learn about the workforce intelligence reports included in the HRMSi product, including Daily Business Intelligence reports, Discoverer workbooks, and Performance Management Framework reports.

Oracle Human Resources Management Systems Approvals Management Implementation Guide

Use Oracle Approvals Management (AME) to define the approval rules that determine the approval processes for Oracle applications.

Oracle Human Resources Management Systems Window Navigation and Reports Guide

This guide lists the default navigation paths for all windows and the default reports and processes as they are supplied in Oracle HRMS.

Oracle iRecruitment Implementation and User Guide

Set up and use Oracle *i*Recruitment to manage all of your enterprise's recruitment needs.

Oracle Learning Management User Guide

Use Oracle Learning Management to accomplish your online and offline learning goals.

Oracle Learning Management Implementation Guide

Implement Oracle Learning Management to accommodate your specific business practices.

Oracle Time and Labor Implementation and User Guide

Learn how to capture work patterns, such as shift hours, so that this information can be used by other applications, such as General Ledger.

Oracle Labor Distribution User Guide

Learn how to maintain employee labor distribution schedules, distribute pay amounts, encumber (commit) labor expenses, distribute labor costs, adjust posted labor distribution, route distribution adjustment for approval, and manage error recovery processes. You also learn how to set up effort reporting for Office of Management and Budget (OMB) compliance.

Other Implementation Documentation

Oracle Workflow Administrator's Guide

This guide explains how to complete the setup steps necessary for any product that includes workflow-enabled processes. It also describes how to manage workflow processes and business events using Oracle Applications Manager, how to monitor the progress of runtime workflow processes, and how to administer notifications sent to workflow users.

Oracle Workflow Developer's Guide

This guide explains how to define new workflow business processes and customize existing Oracle E-Business Suite-embedded workflow processes. It also describes how to define and customize business events and event subscriptions.

Oracle Workflow User's Guide

This guide describes how users can view and respond to workflow notifications and monitor the progress of their workflow processes.

Oracle Workflow API Reference

This guide describes the APIs provided for developers and administrators to access Oracle Workflow.

Oracle E-Business Suite Flexfields Guide

This guide provides flexfields planning, setup, and reference information for the Oracle E-Business Suite implementation team, as well as for users responsible for the ongoing

maintenance of Oracle E-Business Suite product data. This guide also provides information on creating custom reports on flexfields data.

Oracle eTechnical Reference Manuals

Each eTechnical Reference Manual (eTRM) contains database diagrams and a detailed description of database tables, forms, reports, and programs for a specific Oracle Applications product. This information helps you convert data from your existing applications, integrate Oracle Applications data with non-Oracle applications, and write custom reports for Oracle Applications products. Oracle eTRM is available on My Oracle Support.

Integration Repository

The Oracle Integration Repository is a compilation of information about the service endpoints exposed by the Oracle E-Business Suite of applications. It provides a complete catalog of Oracle E-Business Suite's business service interfaces. The tool lets users easily discover and deploy the appropriate business service interface for integration with any system, application, or business partner.

The Oracle Integration Repository is shipped as part of the Oracle E-Business Suite. As your instance is patched, the repository is automatically updated with content appropriate for the precise revisions of interfaces in your environment.

Do Not Use Database Tools to Modify Oracle E-Business Suite Data

Oracle STRONGLY RECOMMENDS that you never use SQL*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle E-Business Suite data unless otherwise instructed.

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL*Plus to modify Oracle E-Business Suite data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle E-Business Suite tables are interrelated, any change you make using an Oracle E-Business Suite form can update many tables at once. But when you modify Oracle E-Business Suite data using anything other than Oracle E-Business Suite, you may change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle E-Business Suite.

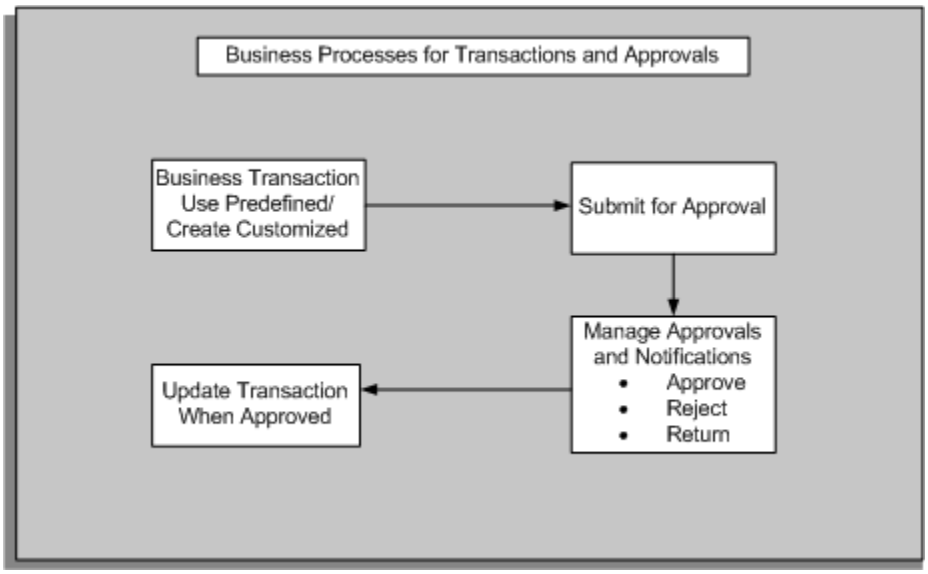
When you use Oracle E-Business Suite to modify your data, Oracle E-Business Suite automatically checks that your changes are valid. Oracle E-Business Suite also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who has changed your information because SQL*Plus and other database tools do not keep a record of changes.

1

Introduction to Oracle Approvals Management

Overview of Oracle Approvals Management

The purpose of Oracle Approvals Management (AME) is to define approval rules that determine the approval processes for Oracle applications. The following graphic illustrates the typical approval process used in an organization.



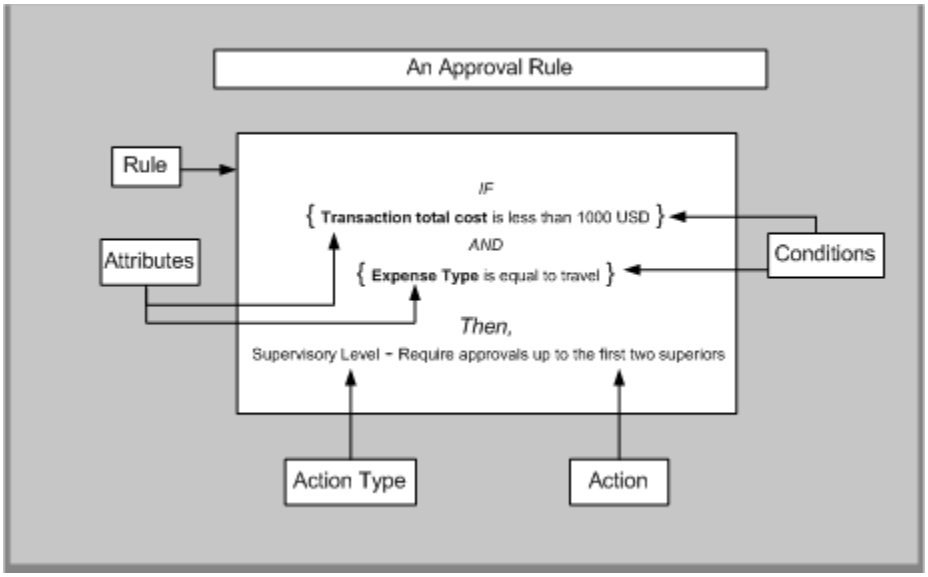
Approval Rules

An approval rule is a business rule that helps determine a transaction's approval process. Rules are constructed from *conditions* and *actions*. For example an approval rule can be as follows:

If the transaction's total cost is less than 1,000 USD, and the transaction is for travel

expenses, then get approvals from the immediate supervisor of the person submitting the transaction.

The following graphic identifies the components of the approval rule:



The approval rule's *if* part consists of zero or more conditions, and its *then* part consists of one or more actions. A condition consists of a business variable (in AME, an attribute) and a set of attribute values, any one of which makes the condition true. An action tells AME to modify a transaction's approval process in some fashion. The conditions in the sample rule in the graphic refer to two attributes: the transaction's total cost, and the transaction's purpose. The sample rule's action tells AME to add the requestor's supervisor to the transaction's approver list.

AME enables you to define rules that express a wide variety of approval rules. For example, rules that:

- Require subject-matter-expert approval
- Require managerial approval
- Create exceptions for rules requiring managerial approval
- Substitute one approver for another in special cases
- Revoke a manager's signing authority in special cases
- Grant a manager extra signing authority in special cases
- Generate a production that assigns a value to a variable name such as the value *digital certificate* to the variable name *eSignature*.

- Send for-your-information notifications.

You can prioritize the approval rules. This enables you to apply rules of sufficient priority to any given transaction.

Transaction Types

An application that uses AME to govern its transactions' approval processes is termed an integrating application. An integrating application may divide its transactions into several categories where each category requires a distinct set of approval rules. Each set of rules is called a transaction type. Different transaction types can use the same attribute name to represent values that are calculated in different ways or fetched from different places. This allows several transaction types to share approval rules (thereby implementing a uniform approval policy across multiple transaction types). A rule use occurs when a transaction type uses a particular rule for a given time period, optionally at a given priority level.

Transaction Deviations

At the time of transaction approval, deviations such as addition of adhoc approvers or deletion of approvers can occur. It is possible to record these transaction deviations by setting the Record Approval Deviations configuration variable to Yes at transaction type level, and running the Approvals Deviations Report after the approval of a transaction. The report displays transaction deviations to the generated approver list.

AME generates an approver list based on the rules setup according to the business requirements. At the time of transaction approval, deviations such as addition of adhoc approvers or deletion of approvers can occur. Any such change to the generated approver list is a deviation. It is possible to record these transaction deviations by setting the Record Approval Deviations configuration variable to Yes at the corresponding transaction type level, and running the Approvals Deviations Report after the approval of a transaction. You can track the deviations by running the Approvals Deviations Report. This report displays the deviations between specified dates.

See: Oracle Approvals Management, page 1-3

See: Running the Approvals Deviation Report, page 2-27

Oracle Approvals Management

Oracle Approvals Management (AME) is a self-service Web application that enables you to define business rules governing the process for approving transactions in Oracle applications that have integrated AME.

What are the advantages of using Oracle Approvals Management?

Oracle Approvals Management enables you as a business user to specify the approval rules for an application without having to write code or customize the application. Once

you define the rules for an application, that application communicates directly with AME to manage the approvals for the application's transactions.

Are all the AME features mentioned in this guide available within integrating applications such as iExpenses or SSHR?

You should review the product documentation to see if a particular feature such as parallel approvers is available in the integrating application you are interested in. AME delivers new features in each release; therefore it is possible that there will be a time lag between delivery and implementation by any given development team.

What kind of approval hierarchies are supported?

You can define approvals by job, supervisor hierarchy, positions, or by lists of individuals created either at the time you set up the approval rule or generated dynamically when the rule is invoked. You can link different approval methods together, resulting in an extremely flexible approval process.

Can you use the same rules for different applications?

Yes. You can define rules to be specific to one application or shared between different applications.

How can you ensure that the rules you create are valid?

AME has built-in testing features that enable you to confirm the behavior of new or edited business rules before live execution.

How is a transaction in progress affected by changes in your organization?

As AME recalculates the chain of approvals after each approval, a transaction is assured to be approved under the latest conditions, regardless of organizational changes, changes to transaction values, rule changes, or currency conversions.

What is a deviation?

A change to the standard approver list generated by AME for approval of transactions is a deviation. AME generates a standard approver list based on the rules set up for transaction types, either by the Approvals Management Administrator or the Approvals Management Business Analyst. Any change to this standard approver list is a deviation.

Can I capture transaction deviations generated to an approval list?

Yes, you can capture deviations generated to the standard approver list by running the Approvals Deviations report.

How can I track the deviations?

You must set the configuration variable Record Approval Deviations at transaction type level to Yes to track deviations.

Approval Processes

Approval Process

Oracle Approvals Management (AME) provides a parallel approval process. The approval parallelization shortens the time that a transaction's approval process requires. The approval process time is reduced as AME imposes a hierarchical (tree) structure on the transaction's approver list, and enables each part of the tree at a given level to progress through its notification-approval cycle in parallel. See: Parallel Approval Process, page 2-8 for details.

A transaction's approval process can have two components:

- List of approvers
- Set of productions

Approver Lists

A transaction's approver list has a hierarchical structure. The transaction's approver list may contain several items' approver lists. Each item's approver list may have three sub-lists. Of these sub-lists, the authority sub-list can have a chain of authority generated by one or more action types. Each approver group or chain of authority can contain multiple approvers.

Items

AME can generate an approver list for a transaction's header, and a separate approver list for each item in the transaction. A transaction type can define multiple item classes. For example, a transaction type might generate separate approver lists for each transaction's header, line items, and cost centers. All transaction types include a header item class, which always has one item (the transaction's header). All other item classes are optional.

Sub-Lists

An item's approver list may contain three sub-lists:

- Pre-chain of authority
- Authority
- Post-chain of authority

The pre- and post-chain sub-lists contain zero or more approver groups; the authority sub-list contains zero or more chains of authority.

Action Types

An action type is a set of actions having a common purpose. Each sub-list can contain approver groups or chains of authority generated by several action types. For example, actions in the absolute-job-level action type all generate chains of authority by ascending the HR supervisory hierarchy until they reach a manager with a particular job level. The actions differ according to the job levels they require.

Approver Groups

An approver group is a collection of approvers that you define. Typically, approver groups contain subject-matter experts.

Chains of Authority

A chain of authority ascends a hierarchy of approvers that are normally defined in applications other than AME, for example HRMS (supervisor position hierarchies). The start point of the chain, and how far it ascends the hierarchy, usually varies between transactions. You can also treat an approver group as a chain of authority. In this case AME ignores the approver group's group-specific properties. Generally, chains of authority contain managers.

Approver groups and chains of authority behave differently in certain circumstances. For example, when one approver forwards a notification requesting approval to another approver. Otherwise, approver groups and chains of authority behave similarly.

Approvers

An approver has the following two properties:

- **Approver Types**

  An approver type is any Workflow Directory Services originating system that defines entities, which can receive Workflow notifications requesting an approval. For example, the HR application defines its set of employees as a Directory Services originating system, so an HR employee can be an approver.

- **Approver Categories**

  AME can generate approvers belonging to either of two approver categories: action and informational (for-your-information or FYI) approvers. Action approvers must approve a transaction. FYI approvers merely receive a notification describing the transaction. The exact content of all notifications depends on the application that generates the notification.

Productions

In AME, a production assigns a value to a variable name. For example, AME might generate a production that assigns the value *digital certificate* to the variable name *eSignature*. AME does not interpret the productions it generates. In fact, AME does not even define any standard production variable names. Rather, it leaves these tasks to integrating applications and their transaction types.

AME generates the following kinds of productions:

- Transaction-level productions that are variable name or value pairs associated with a whole transaction.
- Approver-level productions are associated with specific approvers within a transaction's approver list.
- Item-level productions are associated with each item in the transaction.

What Happens at Run Time

Once you have defined a set of rules for a transaction type, and the application associated with the transaction type is configured to use AME, the application communicates directly with AME to manage the transaction type's approval processes. Typically, the application communicates with AME when a transaction is initiated in the application, and then each time an approver responds to the application's request for approval of the transaction, until all approvers have approved the transaction. AME records each approval, and recalculates the approver list for a transaction each time an approver responds to a request for approval of the transaction.

AME recalculates the approver list each time an approver responds. This enables AME to account for several possible circumstances that can affect a transaction's approver list:

- An attribute value changes, thereby affecting which conditions are true and so which rules apply to the transaction.
- A condition or rule is added, changed, or deleted, again affecting which rules apply to the transaction.
- A change occurs in the organizational hierarchy used by the transaction type's set of rules, thereby changing the membership of the applicable chain of authority.
- Currency exchange rates change, thereby affecting which conditions on currency attributes are true and so which rules apply to the transaction.

By accounting for such changes, AME guarantees that transactions are always approved according to the most current business data possible.

Approval Process Execution

Approval-Process Execution

Integrating applications can communicate with AME in many different ways. For example, an integrating application can ask AME for a transaction's entire approver list, for the rules satisfied, or for the set of approvers the application should notify next.

The Standard Algorithm

Typically, an integrating application follows a simple procedure for managing a transaction's approval process:

1. Ask AME for a transaction's entire approver list.
2. Display the approver list to the requestor, optionally prompting them to suppress or add approvers.
3. Communicate any approver suppressions or additions to AME.
4. Ask AME whether the transaction's approval process is complete, and if not, what approvers (if any) to notify.
5. If AME indicates that no further approvals are required, stop.
6. Notify any approvers identified by AME in step 5.
7. Wait until an approver responds to a notification.
8. Communicate the response to AME.
9. Go to step 4.

Approver-Notification Order

The order in which AME presents approvers for notification at step 4 of the standard algorithm depends on a variety of ordering modes and order numbers that together determine a unique ordering of the approvers in a transaction's approver list.

An *ordering mode* tells AME how to order the collections of approvers at a given level of the hierarchy constituting a transaction's approver list. For example, the sub-list ordering mode basically tells AME whether to notify pre-approvers at the same time as authority approvers, all other things being equal. AME typically uses ordering modes, before a transaction is submitted to AME for approval, where the number of things to be ordered is unknown. For example the approvers generated by a particular action type maybe notified sequentially or in parallel.

Order numbers establish a fixed ordering of a collection of approvers at a given level of the hierarchy constituting a transaction's approver list, for example, the approvers in an approver group are assigned order numbers. Order numbers are not necessarily unique. Thus several approvers in an approver group can have the same order number. AME typically uses order numbers where you know the number of things to be ordered before a transaction is submitted to AME for approval.

2

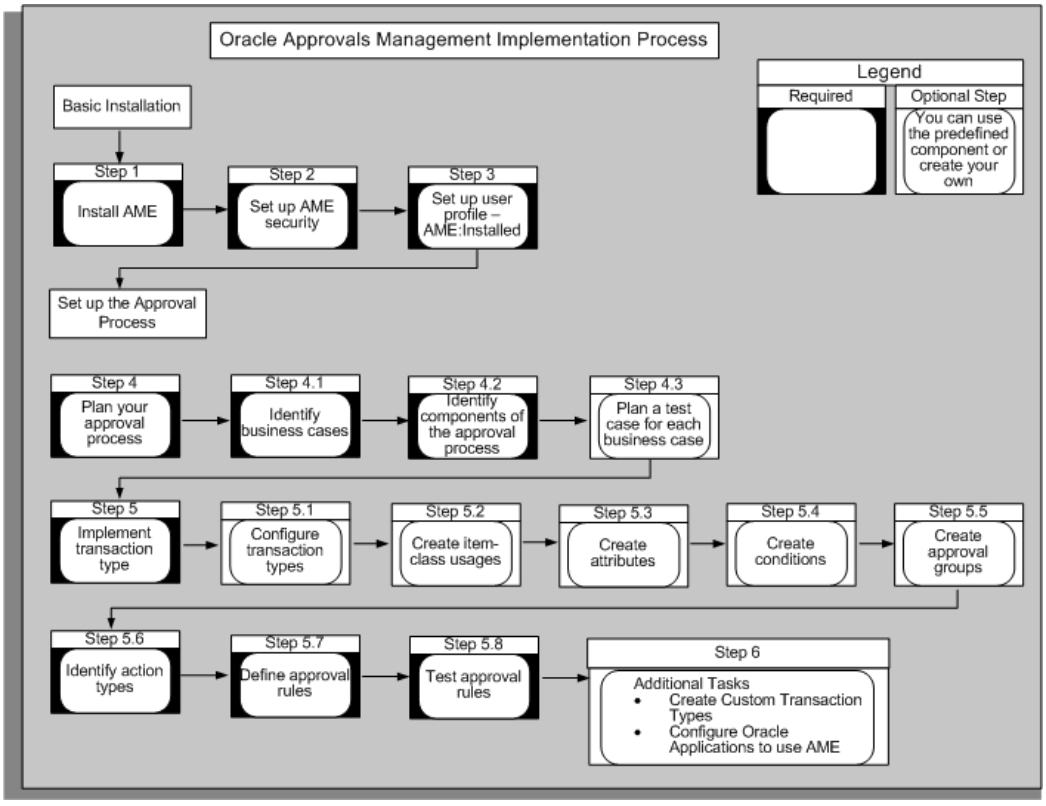
Implementing Oracle Approvals Management

Implementing Oracle Approvals Management

Overview

AME provides you the flexibility to create the approval processes your organization needs without writing programming code. AME's power and flexibility require that you attend carefully to detail during implementation, to ensure your approvals processes function as planned.

The following graphic illustrates the AME implementation process. Depending on your organization's requirements, you may be able to skip some of the implementation steps. Please ensure you understand them all, even if you expect to skip some of them.



Planning your Organization's Approval Processes

Before you begin using an AME transaction type, you should document the approvals processes that the transaction type must implement. A transaction type's requirements document should specify the following:

- A set of business cases

- How to represent the business cases in AME by defining a set of configuration-variable values, item class use, mandatory-attribute values, approver groups, and rule use sufficient to express each business case.
- A representative test case for each business case.

Business Cases

A transaction type's requirements document should account for four types of business cases:

- Transactions having similarly structured approval processes.
- Transactions whose default approver lists include repeated approvers.
- Transactions whose approver's forward approval requests in unusual ways.
- Transactions whose approval processes should have similar parallelization.

Approvals Cases

The first type of business case accounts for general approvals requirements. Such cases should specify informally what kinds of approvals are required for a set of transactions having attribute values that your organization deems similar. For example, a business case might say,

Example

All expense reports for travel having totals above $1,000 and not exceeding $5,000 should be approved by the submitter's two immediate supervisors.

Approvals business cases typically translate directly into a combination of rule use, values for the rulePriorityModes configuration variable, and mandatory-attribute use. If a business case requires approvals from a group of approvers, rather than a chain of authority, the translation can involve defining approver groups as well.

Repeated-Approvers Case

A repeated-approvers case indicates how AME should behave when the approval rules require that an approver appear several times in a single transaction's approver list. (AME can suppress repeated approvers in specified cases.) This business case translates directly to a single transaction-type-specific value for the repeatedApprovers configuration variable.

Special Forwarding Cases

A special case of forwarding occurs, for example, when an approver forwards to someone preceding them in the approver list, or to a subordinate not already in the list. There are eight special cases that translate into eight transaction-type-specific values for the forwardingBehaviors configuration variable.

Parallelization Cases

A parallelization case indicates how AME should treat approvers in the same subset of approvers, at the same level in the hierarchy of approvers that constitutes an approver list. There are six levels:

- Item class
- Item
- Sublist
- Action type
- Group or chain
- Approver

Your requirements document should specify whether approvers in the same subset at each level should be ordered serially or in parallel. For example, it might say that, other things being equal, approvers generated by the same action type should be ordered in parallel.

Representation of Business Cases in AME

There are often several ways to translate your approvals business cases into a transaction type. For example, you can sometimes use fewer or simpler rules by using rule priorities. In general, translating your approvals business cases into a transaction type involves setting values for several transaction-type-specific configuration variables, creating item class use, defining use for several mandatory attributes, defining approver groups, and then creating rules and rule use.

Configuration Variables

The configuration variables you should consider when implementing approvals business cases are:

- adminApprover
- allowAllApproverTypes
- allowAllItemClassRules
- allowFyiNotifications
- productionFunctionality
- purgeFrequency
- repeatedApprovers

- distributedEnvironment
- currencyConversionWindow
- rulePriorityModes
- forwardingBehaviors

Item Class Use

You may wish to use item classes other than those that came with a given transaction type, when implementing your business cases. There are two possible reasons to do so. First, you may wish to define attributes that have a distinct value for each item in the new item class. This would let you define conditions on these attributes, so your rules could account for per-item variations. Second, you may want AME to generate an approver list for each item in the item class.

Mandatory Attributes

The mandatory attributes related to implementing approvals business cases are:

- ALLOW_DELETING_RULE_GENERATED_APPROVERS
- ALLOW_REQUESTOR_APPROVAL
- AT_LEAST_ONE_RULE_MUST_APPLY
- EFFECTIVE_RULE_DATE
- EVALUATE_PRIORITIES_PER_ITEM
- REJECTION_RESPONSE
- REPEAT_SUBSTITUTIONS
- USE_RESTRICTIVE_ITEM_EVALUATION
- USE_WORKFLOW
- WORKFLOW_ITEM_KEY
- WORKFLOW_ITEM_TYPE

Approver Groups

You must create or include approver groups in your transaction type. If a business case requires approvers from a group of approvers that does not exist as a chain of authority in an approver hierarchy supported by AME, then you must define an approver group containing the approvers. When you define the approver group, AME automatically creates the related approval-group actions in all of the action types that come with AME. It can be convenient to list in your requirements document the approver groups

required by the business cases, for ease of reference during implementation. The list must contain for each group a name, description, and membership list. The members must be ordered, but their order numbers do not need to be unique. For example, if you assign all of a group's members the order number one, the group will typically be notified in parallel.

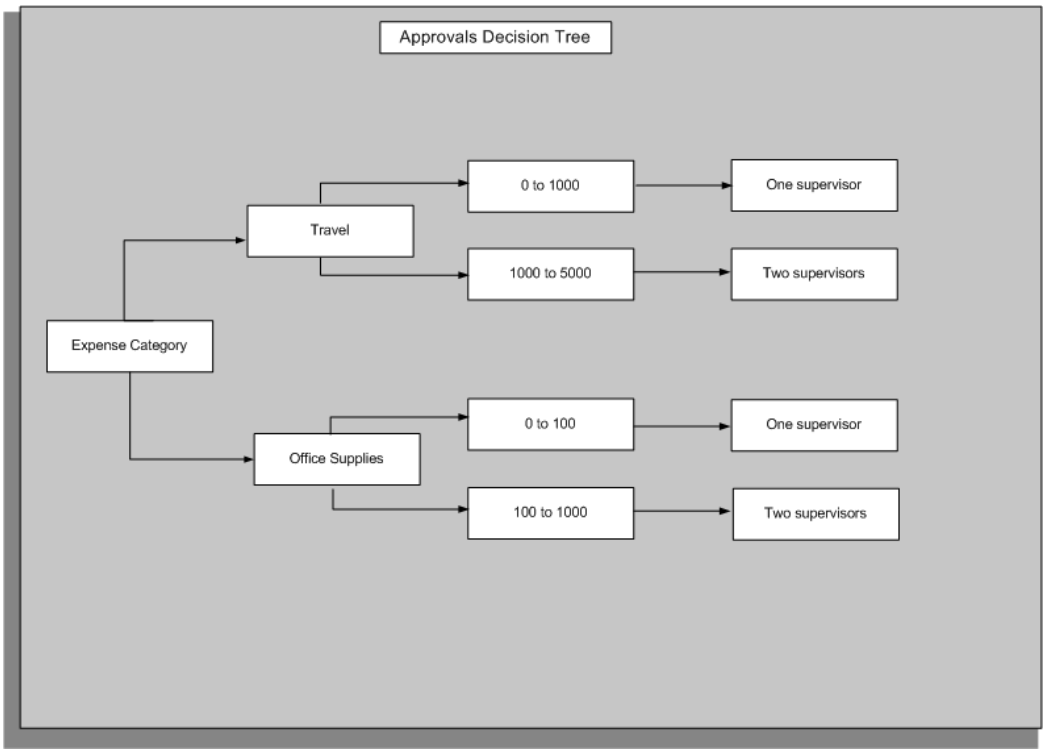
Rules and Rule Use

If your organization has more than a few approvals rules, it can be useful to represent the rules in an approvals matrix or a decision tree. An *approvals matrix* is just a table that had one row per rule. The right-most column contains one or more actions; the other columns contain sets of allowed values for a given attribute. Here's a fragment of an approvals matrix for expense reports, where the rules only depend on the expense category and the expense-report total:

Expense Category	Total	Action
Travel	up to $1,000	one supervisor
Travel	over $1,000 and not over $5,000	two supervisors
Office supplies	up to $1,000	one supervisor
Office supplies	over $1,000 and not over $5,000	two supervisors

(This table is a fragment because it does not account for important business cases. For example, it does not account for expense reports for travel totaling over $5,000. A real approvals matrix should generally be enumerating the business cases exhaustively, even when some of them require no approval, or merely require requestor approval.)

A decision tree typically has one column of nodes for each attribute, with each branch leaving a node representing a set of allowed values for the attribute represented by the node. The final (leaf) nodes represent actions, and a path from the root node to a leaf node represents a rule. The following decision tree is equivalent to the above approvals-matrix fragment:



Decision trees are more flexible than approvals matrixes because they do not have to have the same attributes in the same order along all of the paths from the root node to the leaf nodes. This makes decision trees appropriate where your approval rules' conditions will not all be defined on the same set of attributes. It also complicates verifying that the decision tree represents all of your business cases. When you use a decision tree to represent your set of rules, make sure there is a one-to-one correspondence between your business cases and the tree's paths from the root node to the leaf nodes.

Whatever representation you choose for your sets of rules, keep in mind the following suggestions:

- Make sure your rules capture every business case requiring approval.
- Make sure your rules do not express conflicting outcomes for any given business case.
- Minimize the number of rules that apply to any given transaction.
- Minimize the total number of rules in a transaction type.

Test Cases

A test case should represent a business case. It should specify a value for each of the transaction type's active attributes (that is, all attributes that are mandatory, required by an action type used in a rule that the transaction type uses, or referenced by a condition

used in such a rule). A test case should also specify the approver list that AME should generate in response to the test case's attribute values. AME should produce the correct approval process for all of a transaction type's test cases in a test environment, before the transaction type goes into production.

You may find it convenient to create test cases as transactions in the appropriate integrating application. This lets you test the application's AME integration by viewing approver lists in the application, as well as testing AME's behavior by viewing approver lists on AME's Test Workbench. It also avoids the tedium and possibility of error related to repeatedly entering test cases' attribute values in the Test Workbench's test-transaction pages.

Parallel Approval Process

Approval process parallelization shortens a transaction's approval process time. The parallel approval process imposes a hierarchical (tree) structure on the transaction's approver list and enables each part of the tree at a given level to progress through its notification-approval cycle asynchronous. This enables the approval process of each item in the transaction to continue irrespective of the progress of approval process of other items in the transaction and reduces the transaction's approval process time.

The detailed behavior of parallelization is as follows:

- AME engine constructs an approver tree based on the parallelization configuration at each level.
- The AME engine calculates the order numbers for each level and the approvers at each leaf of approver tree.
- The approval process of nodes which are in parallel with respect to each other start at the same time.
- There after, the approval process of each node proceeds independently.

Approver List

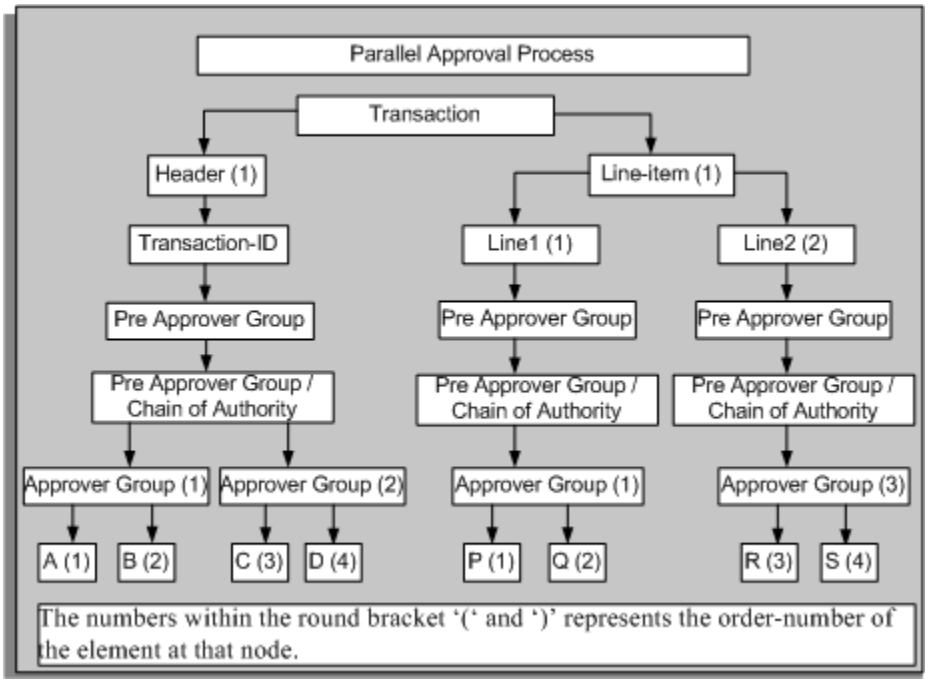
The approver list for a single transaction has a hierarchical (inverted tree) structure with six subordinate levels. Proceeding from the root downward, the levels are:

- Item class
- Item
- Sub-list
- Action type
- Group or chain

- Group or chain member

There can be many objects at a given level for a single object at the next higher level.

The following graphic illustrates an example of the parallel approval process:



From the example given in the graphic, the approver-list tree requires the following eight approvals:

- A
- B
- C
- D
- P
- Q
- R
- S

For example, as seen in the graphic, the sub-tree under 'header' item class progresses independently irrespective of approval process of the sub-tree under 'line-item' item class because they have the same order number. The progress of approvers A, B, C, and D is independent from the progress of approvers P, Q, R, and S, as the item-classes are

in parallel. The parallel approval process progresses as follows:

- A and P are notified
- A approves
- B notified
- B approves
- C notified
- P approves
- Q notified
- Q and C approve
- D and R notified
- D approves
- At this stage, AME does not return any approvers as there are no approvers after D to approve.
- R approves
- S notified
- S approves
- AME completes the approval process.

If you decided to start the approval processes for each item class and again for each item at the same time, then there would be three sub-processes running in parallel. The process for the header would start with A. The line item 1 will start with P and line item 2 would start with R.

To shorten the transaction's approval process further, you could notify all of the approvers in a group or chain at once. Then the process for the header would begin by notifying A, B, C, and D. When all these approvers approve, the approval process for header is complete. Now the longest sub-process of the transaction's overall approval process would require just two notification-approval cycles. In terms of approver order numbers, the list would be:

1. 1 - A
2. 1 - B

3. 1 - C
4. 1 - D
5. 1 - P
6. 2 - Q
7. 1 - R
8. 2 - S

Returning to the example, the decision to parallelize approvals at the item class and item levels amounts to assigning order number 1 to all the item classes, and setting each item class' parallelization mode to parallel. Likewise, the decision to parallelize the transaction's approver groups and chains of authority amount to choosing consensus voting regimes for the approver groups 1 and 2. The example's final approver list, which is consistent with the order number and ordering mode decisions explained above is as follows:

1. 1 - A
2. 1 - B
3. 1 - C
4. 1 - D
5. 1 - P
6. 2 - Q
7. 1 - R
8. 2 - S

Suppressed Repeated Approvers

If the approver list has repeated approvers, then AME suppresses them in a dynamic manner and based on the option selected for the configuration variable repeatedApprovers. See: Configuration Variables, page 9-3

It notifies repeated approvers in the first occurrence during the approval process. For example, let us take the case of two items whose approval is progressing in parallel. Item 1 has three approvers, M, N, and O; Item 2 has three approvers, T, W, and O. The approvers are in serial for each item. With the repeated approver functionality, the occurrence of O in item 2 is set to repeated. However, during the approval process, based on the progress of approvals for item 2, the occurrence of O in item 1 becomes

repeated.

How to Parallelize an Approval Process

This section explains the approval-process parallelizations options available at each level of the approver-list tree and at the same level as the approver-list tree. Please refer to the appropriate topics to learn how to choose an option for a given level of the tree.

The approval process parallelization options at each level of the approver-list tree are:

Item Classes

Item-class parallelization is controlled explicitly via the item class order numbers that a transaction type assigns the item class for which it has an item class use.

Items

An item-class' use parallelization mode determines the item order numbers of the item class' items. If the mode is serial, then the items' item order numbers are sequential, in ascending order of item ID. If the mode is parallel, then all the items are assigned the item order number one.

Sub-Lists

An item-class' sub-list mode determines the sub-list order numbers of the sub-lists in the approver lists of the item class' items. There are four options:

- **Serial** mode assigns pre-approvers the sub-list order number one, authority approvers the sub-list order number two, and post-approvers the sub-list order number three.
- **Parallel** mode assigns all approvers the sub-list order number one.
- **Pre-approvers first, then authority and post-approvers** assigns pre-approvers the sub-list order number one, and other approvers the sub-list order number two.
- **Pre-approvers and authority approvers first, then post-approvers** assigns pre-approvers and authority approvers the sub-list order number one, and post-approvers the sub-list order number two.

Note that all these modes, except the serial mode are only weakly consistent with the ordering implicit in the pre-approver/authority approver/post-approver nomenclature. That is, a pre-approver never follows an authority or post-approver, and an authority approver never follows a post-approver, under any of the sub-list modes.

Action Types

Action-type parallelization is controlled explicitly via the action-type order numbers that a transaction type assigns the action types.

The approval process parallelization options at the same level as the approver-list tree are:

Approver Groups and Chains of Authority

Approver groups and chains of authority occur at the same level of the approver-list tree, and so share the group-or-chain order number.

- **Approver groups**

  Approver-group parallelization is controlled explicitly via the approver-group order numbers that a transaction type assigns the approver groups.

- **Chains of Authority**

  A transaction type assigns a chain of authority ordering mode to each action type that generates a chain of authority. The serial mode gives the chains ascending group-or-chain order numbers, in the order that the chains are generated. The parallel mode gives all chains the group-or-chain order number one.

Approver-group and Chain of Authority Members

Approver-group members and chain of authority members occur at the same level of the approver-list tree, and so share the group-or-chain-member order number.

- **Approver-group Members**

  An approver group functioning as a pre-approval or post-approver group determines its members' group-or-chain-member order numbers using the group's voting regime, and possibly also the member order numbers assigned to static members or generated for dynamic members. (Dynamic members are assigned sequential member order numbers according to the order in which the group's SQL query returns the members.) Approver-group voting regimes can have any of four values. The values determine not only the group-or-chain-member order numbers, but also whether all group members or one must approve the item.

  - The **serial** regime assigns group-or-chain-member order numbers in ascending order of the member order numbers, breaking ties arbitrarily. All group members must approve.

  - The **consensu**s regime assigns group-or-chain-member order number one to all members. All group members must approve.

  - The **first-responder-wins** regime assigns group-or-chain-member order number one to all members. In this case, only the first responder must approve; all other members' responses are recorded but ignored.

  - The **order number** regime sets each member's group-or-chain-member order number to the member's member order number. All group members must approve.

- **Chain of Authority Members**

  An action type that generates a chain of authority has a voting regime. The voting

regime determines the group-or-chain-member order numbers of the approvers in the chains of authority generated by the action type. (This is even true for the chains of authority generated by the approver-group chain of authority action type. This action type ignores the approver-group voting regimes of the groups it uses.) An action type's voting regime can have any of the values allowed for an approver group's voting regime, other than the order-number regime.

AME Roles and Responsibilities

Oracle Approvals Management uses roles and responsibilities to define access levels. It provides security at two levels:

- Data security, which enables you to define access to transaction types for a limited user role.
- Function security, which enables you to define access to AME functions (modules) for a business analyst and administrator.

  To implement function security, AME predefines

  - Permissions, which are a security function that can be used to determine access permissions of an user to an object.
  - Permission sets, which are groups of permissions or navigation functions and permission sets. These sets are used to create hierarchy of permissions.
  - Roles, which are generalized set of functions which can be assigned to a user or group of users.

Permission Sets

If you create your own roles, then use the following predefined permission sets to allow or prevent users from navigating to respective pages:

> **Note:** You must specifically grant individual permission sets to enable users access to multiple pages. For example, if you want a user to access update pages and also want the user to view the details, then you must specifically grant view permissions.

Attributes Tab

Permission Set	Functions
AME Attribute Viewer	Attribute tab access and view attribute details

Permission Set	Functions
AME Attribue Create	Attribute create, copy, and use existing
AME Attribute Update	Attribute update
AME Attribute Delete	AME Attribute Delete
AME Attribute Modifier	AME Attribue Create AME Attribute Update AME Attribute Delete AME Attribute Viewer

Conditions Tab

Permission Set	Functions
AME Condition Viewer	Conditions tab access and view condition details
AME Condition Create	Condition Create (both regular and list mod)
AME Condition Update	Condition Update (both regular and list mod)
AME Condition Delete	AME Condition Delete
AME Condition Modifier	AME Condition Create AME Condition Update AME Condition Viewer AME Condition Delete

Action Types

Permission Set	Functions
AME Action Viewer	Actions tab access and view all details

Permission Set	Functions
AME Action Type Create	Action Type Create permission, and also action type config create
AME Action Type Update	Action Type update and action type config create
AME Action Type Delete	Action Type Delete
AME Action Type Modifier	Action Type Create Action Type Update Action Type Delete
AME Action Create	Action Create
AME Action Update	Action Update
AME Action Delete	Action Delete
AME Action Modifier	AME Action Create AME Action Update AME Action Delete AME Action Viewer
AME Action Type Config Create	Add action type config
AME Action Type Config Update	Action Type Config Update
AME Action Type Config Update	Action Type Config Update
AME Action Type Config Delete	Action Type Config Delete
AME Action Type Config Modifier	AME Action Type Config Create AME Action Type Config Update AME Action Type Config Delete AME Action Viewer

Approver Groups Tab

Permission Set	Functions
AME Approver Group Viewer	Approver Group Tab access and view all details
AME Approver Group Create	Approver group Create in transaction type specific create page
AME Approver Group Update	Approver Group Update in both transaction type specific update and global update page
AME Approver Group Delete	Approver group Delete
AME Approver Group Config Create	Add config to existing groups
AME Approver Group Config Update	Approver Group Config Update
AME Approver Group Config Delete	AME Approver Group Config Delete
AME Approver Group Modifier	AME Approver Group Create AME Approver Group Update AME Approver Group Delete AME Approver Group Viewer

Test Workbench Tab

Permission Set	Functions
AME Test Viewer	Test Workbench tab, view test details, run real and stored test cases, view approval process stages access
AME Test Create	Test Case Create/Save
AME Test Update	Test Case Update
AME Test Delete	Test Case Delete

Permission Set	Functions
AME Test Modifier	AME Test Create AME Test Update AME Test Delete AME Test Viewer

Rules Tab

Permission Set	Functions
AME Rule Viewer	Rules tab access, view rule details, rules table view in dashboard page
AME Rule Create	Rule Create/Duplicate/Use Existing
AME Rule Update	Rule Update
AME Rule Delete	Rule Delete
AME Rule Modifier	AME Rule Create AME Rule Update AME Rule Delete AME Rule Modifier

Administrator Dashboard

Permission Set	Functions
AME Admin Dashboard Viewer	Administrator dashboard access with view page of transaction type
AME Admin Create	Transaction Type Create
AME Admin Update	Transaction Type Update

Permission Set	Functions
AME Admin Delete	Transaction Type Delete
AME Admin Modifier	AME Admin Create AME Admin Update AME Admin Delete AME Admin Dashboard Viewer

Business Dashboard

Permission Set	Function
AME Business Dashboard Viewer	View Dashboard page: Includes business dashboard and transaction type view page

Miscellaneous

Permission Set	Function
AME Setup Report Viewer	Access to Setup Report to view details
AME Exceptions Log Viewer	Access to Exceptions Log page to view details and purge permission
AME Config Variable	Access to configuration variables page with permission to create/Modify transaction type specific values
AME Calling Applications	Permission set to be used for data security grant on AME Transaction Types object

Responsibilities

An Oracle Applications' user must have one of the two available AME end-user responsibilities to use AME. One responsibility is for non-technical (business) users while the other is for technical (administrative) users. The remainder of this guide indicates when AME user-interface functionality requires administrative privileges. Otherwise, you may assume that the business-user responsibilities can access the

functionality that this guide describes.

AME predefines the following responsibilities:

- Approvals Management Business Analyst responsibility enables you to access areas of the user interface that do not require expertise in SQL or PL/SQL programming, or technical knowledge of Oracle Applications.

- Approvals Management Administrator, which has full access to AME's user interface. You typically must grant at least one user administrative privileges in AME, to perform technical tasks such as setting AME's configuration variables.

  Note: If you are an existing customer, then you must ensure to assign these responsibilities to the existing users. You must run the Approvals Management Post Upgrade Process to migrate existing users to the new responsibilities. See: Implementing Oracle Approvals Management, page 2-20

Roles

The following roles are predefined:

- Approvals Management Process Owner - has view only access to Business dashboard, Attributes, Conditions, Action Types, Approver Groups, Test Workbench, Rules. Also access to the setup report page.

- Approvals Management System Viewer - has view only access to the Admin dashboard and Setup Report.

- Approvals Management Business Analyst - has Business dashboard view access, attributes, conditions, groups, test, rules access with create, update, delete permissions. Access to setup report page and configuration variables page with permission to change transaction specific configuration values. Can create, update, delete actions; create, update, delete action type configuration values but cannot create, update, delete action types.

- Approvals Management System Administrator - Admin dashboard access, setup report, exceptions log access, configuration variables access with permission to define transaction type specific values. Can create, update, delete transaction types.

- Approvals Management Administrator - Has all access rights of Business Analyst and System administrator. Can create, update, delete action types. Can modify default configuration values.

Implementing Oracle Approvals Management

To implement AME, you need to carry out the following steps:

1. Install the application.

   AME's installation routines and administration features determine which applications can use AME. Installation and administration are typically jobs for a technical specialist. Installation is generally done only once, and administrative tasks (using AME's Administrator Dashboard) are usually only necessary to enable a new application to use AME, or to access or clear a transaction's error log.

2. Set up AME security by completing the following:

   - Attach the predefined roles to a user or group of users.

     AME uses the Role Based Access Model (RBAC) to provide users access to AME functions. This access model has the following predefined roles:

Role	Description
Approvals Management Administrator	Role inherits Process Owner role and System Administrator role. User with this role can create action types and modify default configuration variables.
Approvals Management Business Analyst	Role provides full access to Business Analyst dashboard pages. Role does not provide access to default configuration variables and create Action Types.
Approvals Management System Viewer	Role provides view only access to Administrator dashboard pages.
Approvals Management System Administrator	Role enables users to create, update, or delete transaction types. Users can access exception logs and configuration variables. This role inherits the System Viewer role.
Approvals Management Process Owner	Role provides view access to all Business Analyst Dashboard view pages.

     The Approvals Management Administrator role inherits all the other roles. Based on your business requirements, you can assign specific roles to users in your enterprise.

     Each of the five predefined roles has a specific set of functional grants. The grants provide users access to AME. To enable functional grants, you must assign roles to users using the User Management page. For information about accessing User Management page, see: *Oracle E-Business Suite System*

Administrator's Guide - Security

Set up user access as follows:

1. Login as Administrator.
2. Select the User Management responsibility.
3. Select the Users page.
4. Search for the user to whom you wish to grant AME roles.
5. In the results table, click Update. In the Update User page, you can view user details along with a list of roles available to the user.
6. Click Assign Roles.
7. Select the roles from the resulting list and click Apply.

When you assign any of the five predefined roles to a user, you are indirectly assigning AME responsibilities to the user.

> **Note:** You can create custom roles to meet your requirements. See: *Oracle E-Business Suite System Administrator's Guide - Security*. For information about permission sets for roles, see: AME Roles and Responsibilities, page 2-14.

- Grant data access to users.

  As AME restricts access to transaction types using Data Security, you grant users access to the transaction types using the Grants page. Set up user access as follows:

  1. Login as Administrator.
  2. Select the Functional Administrator responsibility.
  3. Select the Grants tab.
  4. Click Create Grant.
  5. Select Specific User as grantee type.
  6. Select the user as grantee key.
  7. Select AME Transaction Types as object:
     1. All rows: This grants access to all AME Transaction Types to the User

2. Instance: This grants access to a specific AME Transaction Type specified by the following parameters: FND_APPLICATION_ID => Application ID of the application to which the transaction type belongs TRANSACTION_TYPE_ID => Unique identifier of the AME Transaction type within application

3. Instance set: This grants access to one or more AME Transaction Types specified by the following parameters:

   - Use the predefined instance set AME Transaction Type Instance Set.

   - In the next page, select FND_APPLICATION_ID as Parameter1 and a wild card search string for TRANSACTION_TYPE_ID as Parameter2.

8. In the next page there will be three options for instance type: select AME Calling Applications as the permission set.

9. Review and Finish.

For more information, see: *Oracle E-Business Suite System Administrator's Guide - Security*

Note: If you are an existing customer, then you must ensure your existing users have the Approvals Management Business Analyst and Approvals Management Administrator responsibilities. You must run the Approvals Management Post Upgrade Process to migrate existing users to the new responsibilities. See: Running the Approvals Management Post Upgrade Process, page 2-29

3. Set the user profile - AME:Installed

   This user profile is predefined by AME and is available for integrating applications such as Internet Expenses, iProcurement, to make use of. It is set at the application level and can be used by that application to determine if AME is installed and if so what action to take. In some instances this determination is used to decide if AME should be used for the approval process of that application, this is not always the case however. Please review the specific documentation of the integrating application to determine if that application uses this user profile.

4. Configure transaction types.

   An application administrator should review AME's configuration-variable values as soon as AME is installed and its security has been set up. AME has the following kinds of configuration variables:

- Single-Valued Configuration Variables

  AME's configuration variable *distributedEnvironment* has a single value for the entire application. This variable describe various aspects of AME's computing environment. You must set its value for AME to function properly.

- **Transaction-Type-Specific Variables**

  Other AME configuration variables can have a default value, as well as a value for each transaction type. These variables are:

  - adminApprover
  - allowAllApproverTypes
  - allowAllItemClassRules
  - allowFyiNotifications
  - currencyConversionWindow
  - forwardingBehaviors
  - productionFunctionality
  - purgeFrequency
  - repeatedApprovers
  - rulePriorityModes

  These variables determine many aspects of how AME generates a transaction type's approval processes and are similar to the mandatory attributes. The difference is, their values are always constant for all transactions in the transaction type. Ensure you are satisfied with these variables' default values before using AME.

  Note: You must run the purge utility from the concurrent manager daily to remove old transaction data. Failure to perform this task will eventually result in performance degradation and unlimited growth of the size of certain AME database tables. See: Purging Transaction Data, page 2-28

Implementing the Transaction Type

To implement the transaction type, you need to specify or create, if required, the following components of the approval process:

1. Create item-class use (Optional).

   Once you have documented the approval processes you want a transaction type to implement, you can begin implementing the transaction type. The first step is to register any custom item-class use your transaction type requires.

2. Create transaction attributes (Optional).

   In AME, an attribute is a named business variable such as TRANSACTION_AMOUNT, whose value AME fetches at run time, when it constructs transactions' approver lists. Only a user with the Application Administrator responsibility can create or alter attributes (using the Attributes tab), because doing so generally requires entering or changing an SQL query.

   AME includes the attributes commonly required for the transaction type(s) of each application that can use AME. If your organization has customized an application, or has defined flexfields in it, and wants to use these in the application's approval processes, a user with the AME Application Administrator responsibility must create new attribute names representing the customizations or flexfields, and must define SQL queries that fetch their values at run time. Business users can only select from existing attributes, when they create conditions for AME rules.

3. Create conditions (Optional).

   In AME, a condition specifies a list or range of attribute values required to make a rule apply to a transaction. For example:

   Example
   USD1000 < TRANSACTION_AMOUNT < USD5000

   You create and maintain conditions using the Conditions tab.

4. Create approver groups (Optional).

   You can create AME rules to include one or more approver groups in a transaction's approver list. You create and maintain approver groups using the Approver Groups tab. You must create an approver group before using it in an approval-group rule. You can also add existing approver groups to an approval-group rule.

5. Prepare to use the action types.

   You add action types to a transaction type using the Action Types tab. The action types available to use are:

   - **Predefined action and approver types**

     AME comes with many predefined action types and actions for them. The predefined action types currently support three types of approvers: HR employees (in the HR supervisory hierarchy), HR positions (in the HR position hierarchy), and Oracle Applications (FND) users. The predefined action types ascend the HR hierarchies in many different ways.

An action determines which approvers are included in a transaction's approver list. Typically an action type represents a way to ascend a certain organizational hierarchy, including in a transaction's approver list an appropriate chain of authority from the hierarchy; and an approval specifies where the chain starts and ends. If your organization wishes to require approvals from an organizational hierarchy that none of AME's predefined action types ascend, you need to use a custom action type. The procedure to create a custom action type is detailed within the AME Developers guide.

- **Custom action and approver types**

  AME can support approvers from any originating system registered with Workflow Directory Services (that is, any entity that can function in Workflow as an approver). If your organization requires chains of authority structured differently than those generated by AME's predefined action types, or approvals from approvers in an originating system that AME does not yet support, you may elect to code a custom action type. This requires a significant programming effort (a typical action-type handler PL/SQL package is several hundred lines of code), and an application administrator must register the resulting PL/SQL package with AME. You may also have to register the non-predefined approver type with AME. Currently there is no user interface to register an approver type; one must do so from the SQL*Plus command line. We encourage you to request that AME development release a patch supporting the approver type your organization requires, rather than registering the approver type yourself.

- **Add approvals to existing approval types**

  Your organization may plan to use AME's predefined action types, but may require additional actions. For example, the supervisory-level action type comes with actions for a supervisory hierarchy having at most 10 levels. If your organization has 15 levels, you must create supervisory-level actions for levels 11-15. An application administrator can add these action types using the actions tab.

- **Preparing to use the Job-Level approval types**

  If your organization plans to use one of the job-level action types, it must first assign a job level to each job defined in HRMS (that is, it must first populate the approval_authority column of the HRMS table per_jobs). Your organization should also have a business process for maintaining job levels.

6. Define approval rules.

   With your item-class use, attributes, conditions, approver groups, action types, and actions prepared, you can create your approval rules using the Rules tab. Again, an approvals matrix or decision tree may serve as a convenient checklist.

   In AME, an approval rule associates one or more conditions with an approval

action. The rule applies to a transaction if and only if all of the rule's conditions are true for the transaction.

Each application that can use AME defines one or more transaction types. Each transaction type has its own set of approval rules. Several transaction types can share attribute names, while defining separate use for those attribute names. This makes it possible for several transaction types to share conditions and rules. See: Using Attributes, page 3-7.

7. Test approval rules.

   Once a transaction type has a set of rules, it is critical to test the rules, to ensure they apply to the proper cases and do not contain logical gaps or inconsistencies. You can store these test cases to reuse later.

   There are three ways to test a transaction type:

   - Create a transaction in the integrating application, and use the application's user interface to view the transaction's approver list.
   - Create a transaction in the integrating application, and use AME's Test Workbench tab to view the transaction's approver list.
   - Create a test transaction and view its approver list using AME's Test Workbench tab.

8. Create custom transaction types.

   It is possible to create a custom transaction type from scratch, for instance to use AME as the approvals engine for a custom application. Transaction-type creation is beyond the scope of this guide. If your organization wants to create a custom transaction type, contact Oracle Support and request the *Oracle Approvals Management Developer Guide*. Also see: Creating a Transaction Type, page 9-11.

9. Configure Oracle applications to use AME.

   An Oracle Application should be configured to use AME only after thoroughly testing the set(s) of rules defined for that application's transaction type(s) in AME. Consult the application's user or technical documentation to learn how to configure the application to use AME.

Running the Approvals Deviation Report

You can run the Approvals Deviation report to track transaction deviations after the approval process is complete. This report displays deviations to the generated approver list.

Before you run this report, ensure that you set the configuration variable Record Approval Deviations to Yes at transaction type level to run and track deviations.

1. Using the System Administrator responsibility, navigate to the Submit Request window.

2. Enter Approvals Deviation Report as the name of your request.

3. Click in the Parameters field if the Parameters window does not open. If the Parameters window opens by default, then enter the parameter details specified in the next steps.

4. Select the application for which the transaction types exist.

5. Select the transaction type for which you want to run the report.

6. Specify the From and To dates. You can run the report for transactions between the specified dates.

7. Click OK.

8. Click Submit to purge the deviation data.

Purging Transaction Data

You can run the purge utility to remove old transaction data. You can define the age of transaction data by setting the required value for the configuration variable purge Frequency. See: Configuration Variables, page 9-3

To purge transaction data:

Use the Submit Request window.

1. Using the System Administrator responsibility, navigate to the Submit Request window.

2. Enter Approvals Management Transaction Data Purge as the name of your request.

3. Click in the Parameters field if the Parameters window does not open. If the Parameters window opens by default, then enter the parameter details specified in the next steps.

4. Select the transaction type whose old data you want to purge.

5. Select the transaction to be purged. You can purge all completed, in progress, approved, and rejected transactions.

6. Click OK.

7. Click Submit to purge the old transaction data.

Running the Approvals Deviation Data Purge Process

You can run the Approvals Deviation Data Purge process to purge data of completed transactions. Purging of data is based on a specified date and the type of transaction.

Before you run this process, you must set the date for the system profile AME Deviation Purge Date, as all deviation data to be purged is based on this date. If you do not specify the date or leave it as null, data will be purged till the date specified in the Purge Up To Date parameter.

To Run the Approvals Deviation Data Purge Process:

1. Using the System Administrator responsibility, navigate to the Submit Request window.
2. Enter Approvals Deviation Data Purge as the name of your request.
3. Click in the Parameters field if the Parameters window does not open. If the Parameters window opens by default, then enter the parameter details specified in the next steps.
4. Select the application for which the transaction types exist.
5. Select the transaction type whose data you want to purge.
6. Specify the date till which you want to purge data.
7. Click OK.
8. Click Submit to purge the deviation data.

Running the Approvals Management Post Upgrade Process

If you are an existing customer, then you must ensure your existing users have the Approvals Management Business Analyst and Approvals Management Administrator responsibilities. You must run the Approvals Management Post Upgrade Process to migrate the existing users to these responsibilities.

Use the Submit Request window.

To run the Approvals Management Post Upgrade Process:

1. Using the System Administrator responsibility, navigate to the Submit Request window.
2. Enter Approvals Management Post Upgrade Process as the name of your request.

3. Click in the Parameters field if the Parameters window does not open. If the Parameters window opens by default, then enter the parameter details specified in the next steps.

4. To migrate, do one of the following:

   - Select MIGRATE_USERS to migrate the existing users.

   - Select MIGRATE_ITEM_CLASS_USAGES to migrate the item class use.

   - Select MIGRATE_ALL to migrate users and item class use.

5. Click OK.

6. Click Submit to migrate existing users to the Approvals Management Business Analyst and Approvals Management Administrator responsibilities.

3

Attributes

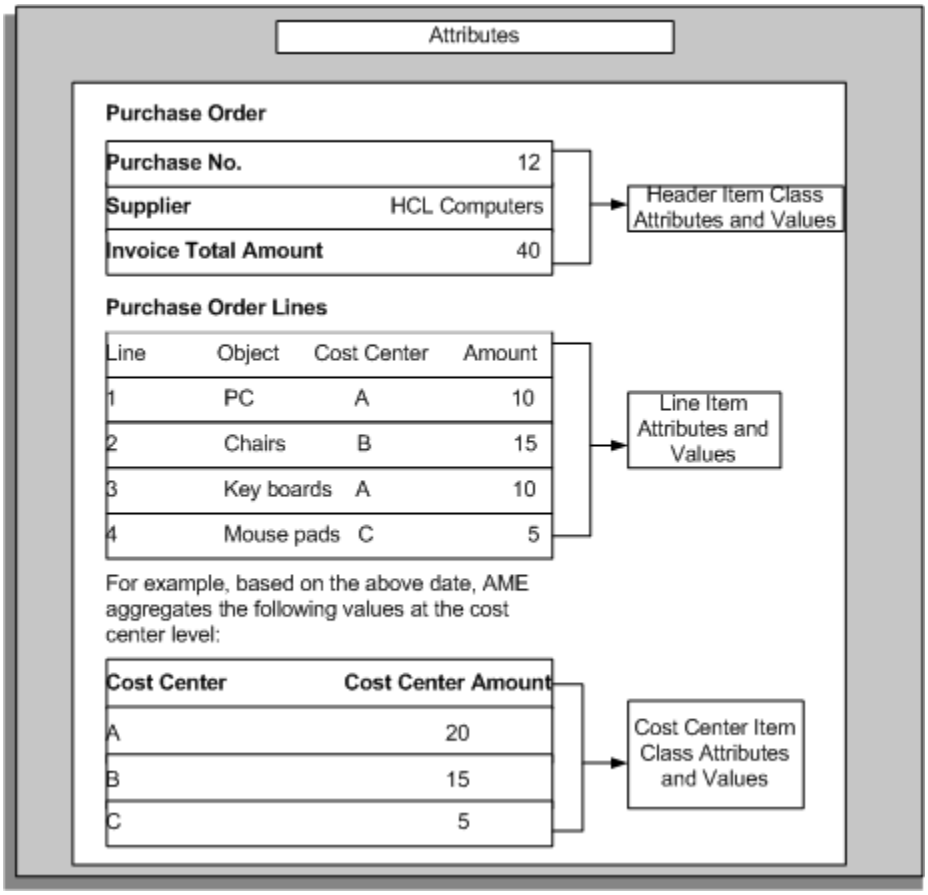
Attributes

Overview

Attributes are business variables with a single value for a particular transaction. Examples of attributes are:

- transaction's total amount
- percentage of discount
- an item's category
- a person's salary

Approval rules use the attributes to determine the outcome such as a value for the transaction's total amount or a value for the percentage of discount. Attributes have an item class such as Header, Line Item, Cost Center, and Project Code. The following graphic uses the example of a purchase order to illustrate attributes and their values:



Creating attribute names and defining attribute use are two important steps in the implementation process for a transaction type. Often a transaction type predefines the attribute names and attribute use that your organization requires. You can use the predefined attributes or create attributes to meet your business requirements.

Attribute Properties

An attribute has the following properties:

Name

An attribute name is a string that represents a decision variable. An attribute name can be at the most 50 bytes long (50 characters long) when you have configured your Oracle Applications to use a single-byte character set. In AME, attribute names always appear in upper case, for example, TRANSACTION_AMOUNT.

> **Note:** If you enter an attribute name in lower or mixed case, AME converts it to upper case before storing it.

All attribute names are shareable. After you create an attribute name, it is available for use in all transaction types. When you create an attribute name, ensure its level of generality reflects your intentions. For example,

- If you want to create an attribute specific to the Web Expenses transaction type, then you might begin your attribute name with the prefix WEB_EXPENSES_.
- If you want to share the attribute name across several transaction types owned by a single integrating application, then give the name a prefix that refers to the integrating application such as HRMS_WEB_EXPENSES_.
- If you want to use the attribute name in all the transaction types, then avoid any references to integrating applications in the attribute name.

Item Classes

An attribute item class determines what class of items have a value for the attribute. For a given transaction, each of the items in an attribute's item class has its own value for the attribute. An attribute always has the same item class, regardless of the transaction type. Attributes belonging to the header item class are sometimes termed header-level attributes. Attributes belonging to subordinate item classes are sometimes termed line-item-level attributes or cost-center-level attributes.

Use

An attribute use tells AME how to get the attribute's value for a given item and transaction, in a given transaction type. Every transaction type can define its own use for a given attribute. This means several transaction types can share conditions and rules, so that an organization can define a single set of rules that applies to several transaction types, thereby implementing a uniform approvals policy across several

applications. It also means that an attribute name can exist without a given transaction type having access to it, or to conditions defined on it, because the transaction type has not yet defined a use for it.

Attribute use can be of two types:

- Static

  A static use specifies a fixed value for the attribute, for all items and transactions.

  It is a common practice to give static use to most or all of the mandatory attributes. One reason is that a static use requires less runtime overhead than a dynamic use. Another is that a static use expresses uniform business policies for all transactions in a given transaction type.

- Dynamic

  A dynamic use specifies an SQL query that AME executes at run time to fetch the attribute's value for each item in the attribute's item class, for a given transaction.

  The execution of dynamic attribute use constitutes the majority of AME's run time overhead. Therefore, optimizing your dynamic use can have a big impact on AME's performance. Ensure you optimize these queries thoroughly, especially if the transaction type that owns them processes a high volume of transactions.

  > **Note:** Oracle strongly recommends that SQL queries used to retrieve information must use only base tables and not secure views. The SQL query must also not include any context-sensitive logic.

Attribute Types

An attribute type indicates the data type of the attribute's values. An attribute's type is always the same, regardless of the transaction type. You cannot change the attribute's type after you create the attribute name. AME distinguishes the following five attribute types:

- Boolean
- Number
- String
- Date
- Currency

Boolean Attributes

Boolean attributes have one of two allowed values at run time: 'true' and 'false'. These strings are case-sensitive. AME defines constants for them: ame_util.booleanAttributeTrue and ame_util.booleanAttributeFalse. Use the values as follows:

- constants in the dynamic attribute use or the source code generating them.
- use the actual values without quote marks in the static use.

Number Attributes

Number attributes can have any numeric value allowed in PL/SQL. Static use for number attributes should either be integers or decimal numbers using the decimal-point character required by the user's character set. For example, '1' and '-2' are acceptable integer values, and '-3.1415' is an acceptable decimal value.

Dynamic use for number attributes should convert a number column to canonical form using the function fnd_number.number_to_canonical. For example the following could be a dynamic use for the TRANSACTION_REQUESTOR_PERSON_ID attribute:

```
select
 fnd_number.number_to_canonical(requestor_person_id)
 from some_application_table
 where transaction_id = :transactionId
```

A number attribute can represent an approver. If you want this to be the case, then you must choose the attribute's approver type when you create the attribute.

> **Note:** You cannot edit a number attribute's approver type after creating the attribute.

String Attributes

String attributes can have any text value up to 100 bytes long (100 characters). The text value can include spaces and ordinary punctuation, and is case-sensitive. You must not enter any return characters in the attribute use as AME removes the return characters from the attribute use. However, there is a difference in the way AME removes the return characters for static and dynamic use. This difference implies that:

- As AME removes the return characters during run time for a static use, a static use for a string attribute cannot contain any return characters.
- A string attribute's dynamic use can technically have a value that includes a return character as AME only removes return characters from the query, not from the query results.

Caution: We encourage you to avoid non-printable characters such as return characters in string attribute values because of the discrepancy between AME's behavior for static and dynamic use with respect to return characters, and as return characters do not display clearly.

Date Attributes

Date attributes can have any value represented as a date variable in PL/SQL as AME distinguishes dates down to the second. Date-attribute values must conform to the format model ame_util.versionDateFormatModel , which currently has the value 'YYYY:MON:DD:HH24:MI:SS' . This value is unlikely to change. For example, '2001: JAN:01:06:00:00' is an acceptable date value. AME's format model for dates differs slightly from the canonical format model. The canonical format model contains a space character, which creates certain technical difficulties.

Ensure your dynamic use for the attribute dates convert date columns to strings using ame_util.versionDateFormatModel, like this:

```
select
 to_char(sysdate, ame_util.versionDateFormatModel) from dual
```

Currency Attributes

Currency attributes represent monetary amounts. The reason AME defines a currency attribute type, rather than treating currency values as number attributes, is to allow for conversion between currency denominations when evaluating a condition defined on a currency attribute. This implies the following:

- The currency attributes have three components: amount, denomination, and conversion method
- Your Oracle Applications instance must use Oracle General Ledger's currency conversion functionality

For example, if TRANSACTION_AMOUNT is a currency attribute, then at run time, AME might need to evaluate the condition

```
TRANSACTION_AMOUNT < 500 USD
```

Upon fetching the attribute's value, AME finds that the amount is in British pounds (not U.S. dollars), and that the attribute requires the 'Daily' conversion method. AME would then use General Ledger's currency-conversion functionality to convert the attribute's value into U.S. dollars, using the 'Daily' conversion method. Later AME evaluates the condition using the converted dollar amount.

If your organization only uses one currency denomination, then you can use number attributes rather than currency attributes to represent monetary values.

Currency-attribute values must have the form:

```
amount,code,type
```

where *code* is a valid currency code and *type* is a valid currency-conversion type. Ensure there are no space characters other than those in the code and type values. For example,

'5000.00,USD,Corporate' might be a valid static currency value.

The amount component of a currency value must be in canonical form. Dynamic currency use must use the function fnd_number.number_to_canonical to return a number value in canonical form. For example, the following SQL statement could be a dynamic use for a TRANSACTION_TOTAL attribute:

```
select
 fnd_number.number_to_canonical(transaction_amount),
 currency_code,
 currency_conversion_type
 from some_application_table
 where transaction_id = :transactionId
```

Using Attributes

All transaction types in AME can share an attribute name, while defining their own method of determining the attribute's value at run time as an attribute use. The ability to share an attribute's name enables several transaction types to share conditions and rules. This further enables an organization to define a single set of rules that applies to several transaction types and have a uniform approvals policy. It also means that you can define an attribute name while a given transaction type may not have yet defined a use for the attribute. A transaction type only has access to conditions defined on attributes for which the transaction type has defined a use. Only users with the System Administrator responsibility can create and edit an attribute's use.

There are two kinds of attribute use: static and dynamic.

Static Attribute Use

A static attribute use assigns a constant value to an attribute, for a given transaction type. A static use always stores an attribute value as a string, regardless of the attribute's data type. It is not necessary to have a value for a static use. To create a null static use, see: Creating an Attribute, page 3-18 and Editing an Attribute, page 3-19

A static use is common but not required for certain mandatory boolean attributes that affect how AME treats all transactions, for example, the AT_LEAST_ONE_RULE_MUST_APPLY attribute. They are similarly used for certain required boolean attributes, for example INCLUDE_ALL_JOB_LEVEL_APPROVERS.

Syntax Rules for Static Use

1. A static use must not use single or double quote marks to demarcate strings.
2. A static use for boolean attributes must be one of the strings 'true' and 'false'.
3. A static use for number attributes must be either an integer or a decimal number in

decimal (not scientific) notation. For example, '1' and '-2' are acceptable integer values, and '-3.1415' is an acceptable decimal value.

4. A static use for date attributes must use the format model ame_util. versionDateFormatModel, whose value is: YYYY:MON:DD:HH24:MI:SS

   For example, '2001:JAN:01:06:00:00' is an acceptable date value.

   > **Note:** This format model differs slightly from the "canonical" format model, which contains a space character. Space characters are problematic for variable values passed to a Web server via the HTTP GET method.

5. A static use for string attributes can be any text value that fits in a varchar2 of length ame_util. attributeValueTypeLength, which is currently 100. (You may wish to check your AME installation's source code to verify that this constant still has the same value.). The text value may include spaces and ordinary punctuation. It is case-sensitive.

6. A static use for currency attributes must have the form:

   amount,code,type

   where code is a valid currency code and type is a valid currency-conversion type. There should be no space characters other than those in the code and type values. The amount should use a period, not a comma, as a decimal point (if any). For example, '5000.00,USD,Corporate' is a valid currency value.

7. A static use may be null. To create a null static use, leave the Usage field empty on the Create an Attribute, Edit an Attribute, or Mandatory Attribute Query Entry page.

Dynamic Attribute Use

A dynamic attribute use assigns an SQL query to an attribute, for a given transaction type. The query must follow certain syntax rules. AME executes the query at run time to determine the attribute's value. A dynamic use is common for all attributes other than the two classes of boolean attributes described under Static Attribute Use.

A dynamic use can be up to 4000 bytes long and should not end with a semicolon. A dynamic use can reference a transaction's ID by using the bind variable ':transactionId'. A dynamic use for header-level attributes must return one row per transaction. A dynamic use for attributes belonging to subordinate-level item classes must return one row for each item in the item class, for a given transaction. The rows must be ordered so that the ith row returned by the dynamic use is the value for the ith item ID returned by the transaction type's item-class use for the attribute's item class. Typically this means the dynamic use will include an order-by clause that references a column containing the

item ID. For example, the query:

```
select item_quantity
 from some_application_table
 where transaction_id = :transaction_id
 order by line_item_id
```

might be a dynamic use for a line-item-level attribute named ITEM_QUANTITY.

Syntax Rules for Dynamic-use Queries

1. The query must fit in the column ame_attribute_usages.query_string, which is a varchar2(2000). If your query is long, you may wish to compare its length with the current table definition in your applications instance.

   > **Note:** You can avoid the length constraint by encapsulating your query in a function that you compile on the database, and then selecting the function's value from dual in your query.

2. The queries for all data types other than currency must select one column; queries for the currency data type must select three columns.

3. Each selected column must convert to a value that fits in a varchar2 of length ame_util. attributeValueTypeLength, which is currently 100. (You may wish to check your AME installation's source code to verify that this constant still has the same value.)

4. Queries for boolean attributes must select one of two possible values, ame_util. booleanAttributeTrue and ame_util.booleanAttributeFalse.

5. Queries for date attributes must convert a date value into a varchar2 using the ame_util.versionDateToString function, to guarantee that AME stores the date value using the ame_util.versionDateFormatModel. AME can only evaluate conditions defined on a date attribute correctly when that attribute's dynamic use converts the attribute's date value using this format model, because AME stores the date as a varchar2, and attempts to convert the stored value back to a date using the same format model. For example the following is a correct dynamic use for a date attribute:

   ```
   Select ame_util.versionDateToString(sysdate) from dual
   ```

6. Queries for number and currency attributes must select the number or currency amount converted to a varchar2 by:

   ```
   fnd_number.number_to_canonical
   ```

7. Queries for header-level attributes may (but are not required to) contain the transaction-ID placeholder ame_util.transactionIdPlaceholder, which is ': transactionId'. The transaction-ID placeholder is a true dynamic PL/SQL bind variable. That is, at run time, AME binds a transaction-ID value to this variable before dynamically executing the query. A condition of a where clause referencing

this bind variable must have the form:

```
transaction ID = :transactionId
```

where transaction ID is a column that contains the transaction ID passed to AME at run time by the application whose transaction type uses the query.

8. Queries for attributes belonging to subordinate-level item classes must return one row for each item in the item class, for a given transaction. For example for OIE Expenses the Item Class line item ID query string is as follows:

```
select
  distribution_line_number from ap_expense_report_lines_all
  where report_header_id = :transactionId
  order by distribution_line_number
```

Therefore a corresponding line level attribute must match this query string, in respect of the number and the ordering of the line item Ids. So for example for the attribute LINE_COST_CENTRE the SQL is as follows:

```
select
  flex_concatenated from ap_expense_report_lines_all
  where report_header_id = :transactionId
  and distribution_line_number in
  (select distribution_line_number
  from ap_expense_report_lines_all
  where report_header_id = :transactionId )
  order by distribution_line_number
```

9. Queries for the dynamic required attributes can generate different starting approvers for each subordinate item in a transaction. See the required attributes in Attribute Classifications, page 3-11

To enable this feature, the dynamic queries must include the item class and item ID bind variables in addition to the transaction ID. For example, the query can be as follows:

```
select
  person_id from invoice_table
  where transaction_id = :transactionId
  and item_class = :itemClass
  and item_id = :itemId
```

Can a dynamic attribute use reference the value of another attribute?

For example:

```
Attribute1: select column1 from table1
  where column2 = :transactionId
Attribute2: select column3 from table2
  where column4 = :Attribute1
```

A dynamic attribute use cannot reference other attributes' values. To implement the above example, the second attribute's dynamic use would be:

```
select column3 from table2 where column4 =
  (select column1 from table1 where column2 = :transactionId)
```

AME does not allow references to attribute values within the dynamic use for two reasons:

1. It is not practical for AME to guarantee that a transaction type's active attributes (those attributes whose values the engine and/or rules require to determine which rules apply to a transaction of that type) will always be fetched in a fixed order. As a result, AME may not have fetched the value of Attribute1 before it tries to fetch the value of attribute2. Even if AME were to guarantee an attribute-fetching order, that would not solve the general problem. It would only solve the problem for attributes that come after the ones they depend on, in the order.

2. Were AME to allow this sort of dependency among attributes, cycles could occur. Attribute1 could depend on Attribute2, and Attribute2 could depend on Attribute3, which in turn could depend on Attribute1. There would be no way in logic to resolve these attributes' values.

Attribute Classifications

An attribute classification indicates whether an attribute is mandatory, required, active or represents business variables.

Mandatory Attributes

Mandatory attributes determine various facets of AME runtime behavior. A mandatory attribute must have a use in all transaction types. AME displays the mandatory attributes at the top of the attributes list that appears when you select the Attributes tab in the application. AME predefines the following mandatory attributes:

- **ALLOW_DELETING_RULE_GENERATED_APPROVERS**

  This is a boolean attribute. It determines whether AME enables the integrating application to suppress from a transaction's approver list the approvers required by the transaction type's rules.

- **ALLOW_REQUESTOR_APPROVAL**

  This is a boolean attribute. It determines whether AME enables a requestor to approve their own transaction, if they have sufficient signing authority. When this attribute is true, and a requestor has sufficient authority, the relevant action type makes the requestor the only approver in their chain of authority, and assigns the requestor the initial approval status ame_util.approvedStatus (approved). When the attribute is false, the relevant action type does not include the requestor in the chain of authority. Instead, the action type includes in the chain of authority at least one of the requestor's superiors.

  In the previous versions of AME, the AME engine evaluated the attribute ALLOW_REQUESTOR_APPROVAL, so that it was appropriate for the attribute to be mandatory. In the current version of AME, the following action types require the action type instead:

- Absolute job level
- Final approver only
- Manager then final approver
- HR position
- HR position level
- Supervisory level

It is likely that ALLOW_REQUESTOR_APPROVAL will be removed from the set of mandatory attributes, and treated instead as a required attribute for the above action types (and any others that require the attribute).

- **AT_LEAST_ONE_RULE_MUST_APPLY**

  This is a boolean attribute determining whether AME raises an exception when no rules apply to a transaction at run time.

- **REJECTION_RESPONSE**

  This is a string attribute. It determines how AME responds when an approver rejects an item. The attribute has three allowed values:

  - **ame_util.continueAllOtherItems** currently has the value 'CONTINUE_ALL_OTHER_ITEMS'. When REJECTION_RESPONSE has this value, AME continues the approval processes of all items other than the item(s) that were rejected.
  - **ame_util.continueOtherSubItems** currently has the value 'CONTINUE_OTHER_SUBORDINATE_ITEMS'. When REJECTION_RESPONSE has this value, AME continues the approval processes of all subordinate-level items other than the item(s) that were rejected, but stops the approval process of the header and the rejected item(s).
  - **ame_util.stopAllItems** currently has the value 'STOP_ALL_ITEMS'. When REJECTION_RESPONSE has this value, AME stops the approval processes of all of the transaction's items, including the header.

    Note: Ensure you use the actual value in a static use, and the ame_util constant in the code executed by a dynamic use.

- **USE_RESTRICTIVE_ITEM_EVALUATION**

  This is a boolean attribute. It indicates whether AME requires a single subordinate-level item to satisfy all conditions on attributes belonging to the subordinate item

class, in a given rule, for the rule to apply. If the attribute is true, then a rule containing conditions on a subordinate-level item class' attributes only applies if one of the transaction's items in that class satisfies all of the rule's conditions on attributes of that item class. If the attribute is false, then different items may satisfy different conditions on the subordinate item class' attributes.

For example, consider the rule:

```
If
LINE_ITEM_CATEGORY in {office furniture, office supplies}
and LINE_ITEM_TOTAL < 1,000 USD
 then
  require approvals up to the first three superiors,at most
```

Suppose a transaction has two line items with the attribute values shown in the following table:

Line-Item ID	Attribute	Attribute Value
1	LINE_ITEM_CATEGORY	office furniture
1	LINE_ITEM_TOTAL	1,100 USD
2	LINE_ITEM_CATEGORY	travel
2	LINE_ITEM_TOTAL	900 USD

If USE_RESTRICTIVE_ITEM_EVALUATION is false, then the rule applies as line-item one satisfies the first condition and line-item two satisfies the second. If the attribute is true, then the rule does not apply as, neither line item satisfies both the conditions.

- **EFFECTIVE_RULE_DATE**

  When AME begins to process a transaction, this date determines which rules are active for a given transaction. AME then evaluates each active rule's conditions to see whether the rule actually applies to the transaction.

  For most transaction types, the system date (sysdate) is the appropriate EFFECTIVE_RULE_DATE value. To use this value, provide EFFECTIVE_RULE_DATE a static use with no value. This is more efficient at run time than giving it a dynamic use that selects the system date from dual.

- **EVALUATE_PRIORITIES_PER_ITEM**

  This is a boolean attribute. It determines whether AME evaluates the applicable rules' use priorities per item. Per-item evaluation is relevant for relative rule-priority modes. When the attribute is true, AME evaluates the rules' use priorities that apply to each item as a group. When the attribute is false, it evaluates the use

priorities of all the rules applying to the transaction together.

- **USE_WORKFLOW**

  This is a boolean attribute. It indicates whether AME should log the transaction type's exceptions to the Workflow exception stack. Ordinarily this attribute must have a static use.

- **WORKFLOW_ITEM_KEY**

  This is a string attribute. It is a transaction's item key in the integrating application's workflow. The item key is typically also the AME transaction ID, and in this case, the attribute's dynamic use can select :transactionId from dual. AME uses this attribute's value when it logs exceptions in Workflow's exception stack. If the integrating application does not use Oracle Workflow (if USE_WORKFLOW is false), then ensure this attribute has a static use with no value.

- **WORKFLOW_ITEM_TYPE**

  This is the item type of the integrating application's workflow. AME uses this attribute's value when it logs exceptions to Workflow's exception stack. If the integrating application does not use Oracle Workflow, then ensure this attribute has a static use with no value.

- **REPEAT_SUBSTITUTION**

  This is a boolean attribute with a default value of false, in order to make it backward compatible. AME uses this attribute to process the substitution rules a second time at the end of the processing cycle. This is done to apply the Substitution rules on adhoc insertions and surrogate approvers. The attribute's value can be set or reset dynamically for any transaction. For example, if you would like the substitution rules to run again only for a certain set of transactions, this attribute value can be set dynamically to true or false.

Required Attributes

An action type may rely on the values of one or more attributes to govern various aspects of the action type's runtime behavior. Such attributes are the action type's required attributes. The required attributes must have a use in a given transaction type, for the action type to be available in the transaction type.

AME predefines the following required attributes whose use is more generally explained in Required Attributes For Action Types, page 5-4 and Rules, page 7-2:

- **ALLOW_EMPTY_APPROVAL_GROUPS**

  When this attribute is false, AME raises an exception if an approver group has no members at run time. This typically occurs if the approver group is dynamically generated and the SQL that populates the group, fetches no members. For further information, see Approver Groups, page 6-2.

- **FIRST_STARTING_POINT_PERSON_ID**

  and **SECOND_STARTING_POINT_PERSON_ID**

  These attributes' values identify the first approver in each chain of authority generated by the action type Dual Chains of Authority.

- **INCLUDE_ALL_JOB_LEVEL_APPROVERS**

  This attribute determines if all approvers with the same job level should be included when building the chain of authority for the action types that depend on Job Level. For further information, see Required Attributes For Action Types, page 5-4.

- **JOB_LEVEL_NON_DEFAULT_STARTING_POINT_PERSON_ID**

  If this attribute is not null, the approver it identifies will be used as the starting point for action types that depend on Job Level. Its value will override the setting for TRANSACTION_REQUESTOR_PERSON_ID.

- **NON_DEFAULT_POSITION_STRUCTURE_ID**

  The position level action type handler uses this attribute to walk up the hierarchy. If the value of this attribute is null, then AME uses the primary position structure of the current business group for finding the parent position, otherwise AME uses the specified position structure.

- **NON_DEFAULT_STARTING_POINT_POSITION_ID**

  If this attribute is not null, then the position it identifies will be used as the starting point for position level action types. Its value will override the setting for TRANSACTION_REQUESTOR_POSITION_ID.

- **SUPERVISORY _NON_DEFAULT_STARTING_POINT_PERSON_ID**

  If this attribute is not null, then the approver it identifies will be used as the starting point for action types that depend on Supervisory Level. Its value will override the setting for TRANSACTION_REQUESTOR_PERSON_ID.

- **TOP_POSITION_ID**

  Similar to the explanation below for TOP_SUPERVISOR_PERSON_ID, it represents the top of the position hierarchy.

- **TOP_SUPERVISOR_PERSON_ID**

  This attribute should be assigned to the person ID of the employee at the top of the hierarchy (typically the CEO of a corporation). This is used in Action Types that climb the supervisor hierarchy and is needed to determine if AME has reached the top of the hierarchy as opposed to reaching a gap in the hierarchy setup.

- **TRANSACTION_REQUESTOR_PERSON_ID**

This attribute should be assigned to the ID of person requesting the transaction. Several of the defined Action Types will use this as the starting point for traversing the hierarchy.

- **TRANSACTION_REQUESTOR_POSITION_ID**

  This attribute specifies the requestor's position ID for a transaction

Additionally, the following required attributes' values can identify a different first approver for each subordinate item in a transaction:

- FIRST_STARTING_POINT_PERSON_ID
- SECOND_STARTING_POINT_PERSON_ID
- JOB_LEVEL_NON_DEFAULT_STARTING_POINT_PERSON_ID
- NON_DEFAULT_POSITION_STRUCTURE_ID
- NON_DEFAULT_STARTING_POINT_POSITION_ID
- SUPERVISORY_NON_DEFAULT_STARTING_POINT_PERSON_ID

> **Note:** To enable this feature, you must ensure the dynamic query for these attributes includes the item class and item ID bind variables in addition to the transaction ID. See the syntax rules for dynamic use queries in Using Attributes, page 3-7.

Business-Variable Attributes

Non-mandatory attributes that are not required attributes represent business variables used in approval rules. In AME, these are the attributes that appear in conditions. (You can also use required attributes in conditions, though doing so is uncommon. You cannot use mandatory attributes in conditions.)

Active Attributes

An attribute is active for a rule if one or more of the rule's conditions is defined on the attribute, or if one of the action types of the rule's actions requires the attribute. An attribute is active for a transaction type (or simply active) if the attribute is either mandatory or active for at least one of the transaction type's rules. AME only fetches the values of active attributes at run time. It lists all attributes on the Test Workbench tab, even if they are not active.

How does AME use Attributes?

When AME starts to calculate a transaction's approver list at run time, it does the following:

1. It fetches the values of each attribute that is active for the transaction type.

   To do this, AME either fetches the constant value assigned to the attribute, or fetches the attribute's query and then executes it. If an attribute having a dynamic use is a header-level attribute, AME executes the query once for the entire transaction. If the attribute is a line-item-level attribute, AME executes the query once for each of the transaction's line items.

2. After fetching all of the active attributes' values, AME checks whether each of a transaction type's rules applies to the transaction.

   It does this by determining whether each of the rule's conditions is true. For conditions defined on header-level attributes, the condition is true if the attribute's value lies within the list or range of values that the condition defines. For conditions defined on line-item-level attributes, the condition is true if the value of the attribute for any one line item lies within the list or range of values that the condition defines.

Deciding Whether to Create or Edit Attributes

Ideally, a transaction type delivered with AME predefines the attribute names and attribute use that your organization requires. However, to meet your business requirements you may want to add attributes and their use, or change the predefined use.

> **Note:** Even if you do not need to change the predefined attributes or their use, you still must verify that the predefined attributes and their use meet your business requirements.

The following steps explain the general procedure for deciding which attributes and attribute use you need to create or edit, how to add or edit them as needed, and test the results:

1. Consult the integrating application's documentation to determine the purpose and meaning of each predefined attribute.

2. Compare the predefined attributes' use with those your transaction type's implementation document specifies. See the Mandatory Attributes subsection of the Representation of Business Cases in AME, page 2-4.

3. Edit any attribute use that does not match your requirements.

4. Create any attribute names and attribute use that your implementation requires, and which are not predefined.

5. Use the Test Workbench tab to compare the attribute values of your test cases defined with their expected values.

6. Using your Application account, execute the dynamic attribute use from the SQL*Plus command line for your test cases to verify that the attributes have the same values in SQL*Plus as on the Test Workbench tab.

Viewing Attributes

You can view attributes using the Attributes tab.

The Attributes page enables you to view attributes present in your transaction type, create new, and reuse existing attributes to define how Oracle Approvals Management fetches business facts from a transaction.

To display the list of attributes for a transaction type:

Use the Business Dashboard and Attributes pages.

1. Select the required transaction type in the Approval Process Setup available in the Business Dashboard.

2. Click the Attributes link.

   The Attributes page opens displaying the attributes list.

Creating Attributes

To create an attribute:

Use the Create Attributes page.

1. Click Create on the Attributes page to open the Create Attributes page.

2. Enter the attribute's name.

   If you plan to create one or more rules referencing the attribute, and to share the rule(s) across several transaction types, you just need to create the attribute's name for the first transaction type, and then select it from the list of shareable attribute names for all remaining transaction types. You must enter a distinct use for the attribute name, for each transaction type.

   All attribute names, including those you create, are shareable; so make sure that your attribute name's degree of generality reflects your intentions. For example, if you want to create an attribute specific to the Web Expenses transaction type, you

might begin your attribute name with the prefix 'WEB_EXPENSES_'.

> **Note:** If you enter an attribute name in lower or mixed case, AME changes it to all upper case when it saves your work.

3. Select the attribute's level (header or line-item) if the transaction type has enabled line-item attributes.
4. Select an attribute type. Ensure to use the currency type for attributes reflecting monetary values.
5. Optionally, select the approver type for Number data type.
6. Select a value set.

   > **Note:** A value set contains business specific data. The application enables Value Set when you select Number, String, or Currency data type.

7. Select the attribute use. If you select Dynamic, then enter an SQL query and click validate to ensure the query is correct.
8. Click Create Another to create another attribute, if required.
9. Click Apply to add the attribute to your transaction type.

   Your new attribute appears in the appropriate section of the attributes list.

Editing Attributes

To edit an attribute:

Use the Update Attribute page.

1. In the Attributes page, display the list of attributes.
2. Click the Update icon for the attribute that you want to edit.
3. Make your changes on the Update Attribute page.
4. Click Apply to update the attribute.

Restrictions

When you change an attribute's description, your changes apply for all transaction

types. Changes to the attribute use only apply to the transaction type for which you make the change.

You cannot change the name, type, or description of predefined attributes. You must instead create a new attribute with the name, type, and description that you want. In some cases, you can create or edit a predefined attribute's use. You can change the use and value set of an attribute. If you change a use type from static to dynamic, then you must change the use value accordingly.

> **Note:** Queries for the dynamic required attributes can generate different starting approvers for each subordinate item in a transaction if the query includes the item class and item ID bind variables in addition to the transaction ID. See the required attributes in Attribute Classifications, page 3-11

Deleting Attributes

To delete an attribute:

Use the Attributes page.

1. Display the list of attributes.
2. Click the Delete icon for the attribute that you want to delete.
3. Confirm the deletion when prompted.

> **Note:** You cannot delete predefined attributes. You cannot delete an attribute if it has a condition based on it, which a current or future rule uses. Additionally, when you delete an attribute that is not shared with any other transaction type, AME deletes all the conditions associated with this attribute.

Copying Attributes

To copy an existing attribute into your transaction type:

Use the Use Existing Attribute page.

1. Click Use Existing Attribute on the Attributes page to open the Use Existing Attribute page.
2. Select an attribute and click Continue to copy the existing attribute into your transaction type.

3. Modify the attribute's use.

   Note: Queries for the dynamic required attributes can generate different starting approvers for each subordinate item in a transaction if the query includes the item class and item ID bind variables in addition to the transaction ID. See the required attributes in Attribute Classifications, page 3-11

4. Click Finish to use the selected existing attribute in your transaction type.

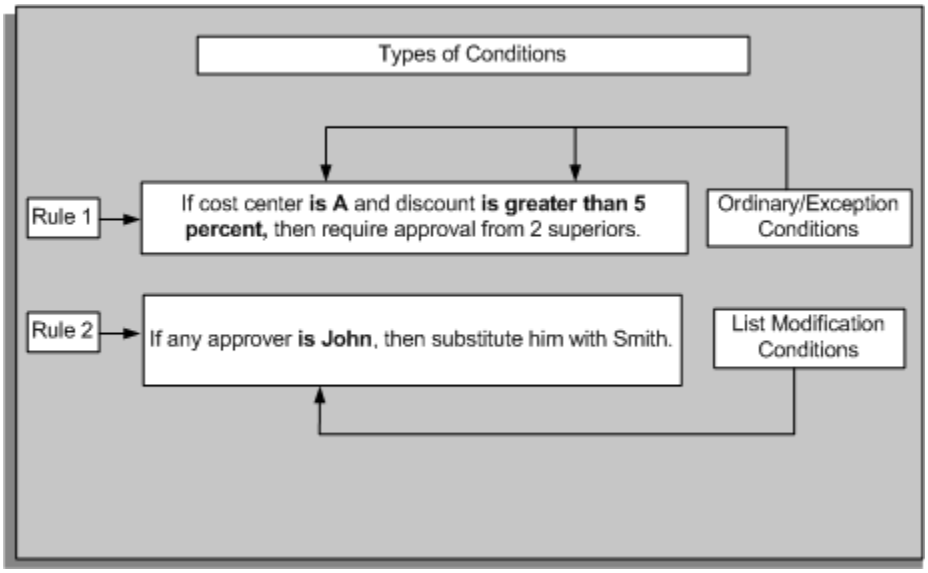
4

Conditions

Conditions

Overview

The *if* part of an approval rule consists of zero or more conditions. A condition is a statement that is either true or false, for a given transaction. For the rule to apply to a transaction, all of its conditions must be true for the transaction. The following graphic illustrates the conditions in rules and indicates the types of conditions:



While implementing the transaction type, creating conditions is an optional step.

Condition Types

There are two types of conditions:

- Regular, which has Ordinary and Exception conditions
- List-modifiers

Regular Conditions

Ordinary

An ordinary condition associates an attribute with a set of allowed values or range. Such a condition is true when the attribute has one of the allowed values. We then say that the attribute's value satisfies the condition. Ordinary conditions have one of the following three forms.

Conditions on Date, Number, and Currency Attributes

Conditions on number attributes associated with an approver type have the form,

```
attribute_name = approver_name
```

That is, the condition is true when the attribute's value is the ID of a specified approver of the type associated with the attribute.

> **Note:** While defining a condition, if the attribute is linked to a value set, then you can make use of the available values in the value set.

Conditions on date and currency attributes, and number attributes not associated with an approver type, have the form,

```
lower_limit <{=} attribute_name <{=} upper_limit
```

When you create or edit this kind of condition, you decide whether to include the equals sign in the condition by selecting the appropriate values in the Expression region on the Create Condition and Edit Condition pages. If you create a set of conditions of this type on the same attribute, ensure to include each lower or upper limit only in one condition. For example, the conditions

```
1,000 <= TRANSACTION_AMOUNT <2,0002,000 <= TRANSACTION_AMOUNT < 3,000
```

have successive ranges of values that avoid overlapping at 2,000. In this way, at most one of the conditions will ever be true for any given transaction, which is typically the desired behavior.

Conditions on Boolean Attributes

Conditions on boolean attributes have the form,

```
attribute_name is {true, false}attribute_name is {true, false}
```

Ensure you understand the logic of your boolean conditions. Consider the following truth table:

Attribute Value	Allowed Value	Condition Value
true	true	true
true	false	false
false	true	false
false	*false*	*true*

That is, there are two ways a condition on a boolean attribute can be true. (Some AME users forget the italicized case in the above truth table.)

Conditions on String Attributes

Conditions on string attributes have the form,

```
attribute_name in {value_1, value_2, . . . }
```

The allowed values are case-sensitive, so *allowed value* and *ALLOWED value* are different allowed values. If you want a condition on a number or date and the condition to have a list of allowed values, then define a string attribute representing the textual representation of the number or date. Later define a condition on this string attribute.

Exception

Exception conditions have the same forms as ordinary conditions. The difference is, an exception condition can only appear in exception rules. See Rule Types, page 7-2 for details.

List-Modifiers

A list-modification condition checks for the presence of a given target approver at a specific set of allowed locations in the default approver list. When the target approver is present in an allowed location, we say that the default approver list satisfies the condition.

There are two types of allowed location. When the condition specifies that the target approver must be the final approver, the target approver must be the last approver in a chain of authority. When the condition specifies that the target approver may be any approver, whether the approver must be in a chain of authority depends on the action type used by the rule that contains the list-modification condition. Action types for list-modification rules limit their scope to chains of authority. Substitution rules using an any-approver list-modification condition apply to all occurrences of the target approver throughout an approver list.

Condition Scope and Availability

An ordinary or exception condition is available in any transaction type that defines a use for the attribute that the condition references. If the value of the allowAllApproverTypes configuration variable is set to Yes, then all list modification conditions are available for the transaction type. If this configuration variable is set to No, then only the list modification conditions based on HR People and FND USER are available to the transaction type. The sharing of conditions simplifies rule sharing among transactions. It also means that when you edit a condition, the change affects all rules that use the condition; and when you delete a condition, the condition is no longer available in any transaction types.

Viewing Conditions

To display the list of conditions for a transaction type:

Use the Business Dashboard and Conditions pages.

1. Click the Conditions tab to display the Conditions page. If you are navigating from the Business Dashboard, then select the required transaction type in the Approval Process Setup available in the Business Dashboard.

2. Click the Conditions link.

   The Conditions page opens displaying the list of conditions that are available to the current transaction type.

Creating an Ordinary or Exception Condition

To create an ordinary or exception condition:

Use the Create New Condition page.

1. Click the Conditions tab to open the Conditions page.

   Note: By default, the page displays the Regular conditions consisting of the ordinary and exception conditions.

2. Click Create.

3. Select the condition type.

4. Select the attribute. Based on the attribute you select, you can record the following condition-related information in the Expression region:

   - If the attribute is of type number, and is associated with an approver type, then query for and select an approver of the type associated with the attribute.

   - If the attribute is of type date or currency, or is a number attribute not associated with an approver type, then enter the condition's lower and upper limits, and indicate whether each should be included in the condition's set of allowed values. If you leave a lower or upper limit blank, then the range of allowed values is unbounded on that side. If the attribute is of currency type, then choose the denomination of the lower and upper limits. The limits must be in the same denomination.

   - If the attribute is of type boolean, then select the condition's allowed value.

   - If the attribute is of type string, then enter the condition's allowed values.

     Note: If the attribute is associated with a value set, then you must select the value from the list of values.

5. Click Create Another to create another condition.

6. Click Apply to make the condition available for use within your approval rules.

Creating a List-Modification Condition

To create a list-modification condition:

Use the Create New List Modification Condition page.

1. Click the Conditions tab.

2. Click the List Modifiers link to display the list modification conditions Search page.

3. Click Create.

4. Select the target approver's allowed location as the approver order.

5. Select the type of the condition's target approver.

6. Query for and select the target approver.

7. Click Apply to make the condition available for use within your approval rules.

Editing Conditions

To edit a condition:

Use the Update Condition page to edit an ordinary or exception condition and the Update List Modifier page to edit a list modification condition.

1. In the Conditions page, display the list of regular or list modification conditions.

2. Click the Update icon for the condition that you want to edit.

3. Make your changes.

4. Click Apply to update the condition.

Deleting Conditions

You can delete a condition only if it is not used in any current or future rule.

To delete a condition:

Use the Conditions page.

1. Display the list of regular or list modification conditions, as required.
2. Click the Delete icon for the condition that you want to delete.

   Warning: If an inactive rule is using the deleted condition, then you cannot reinstate that rule.

3. Confirm the deletion when prompted.

5

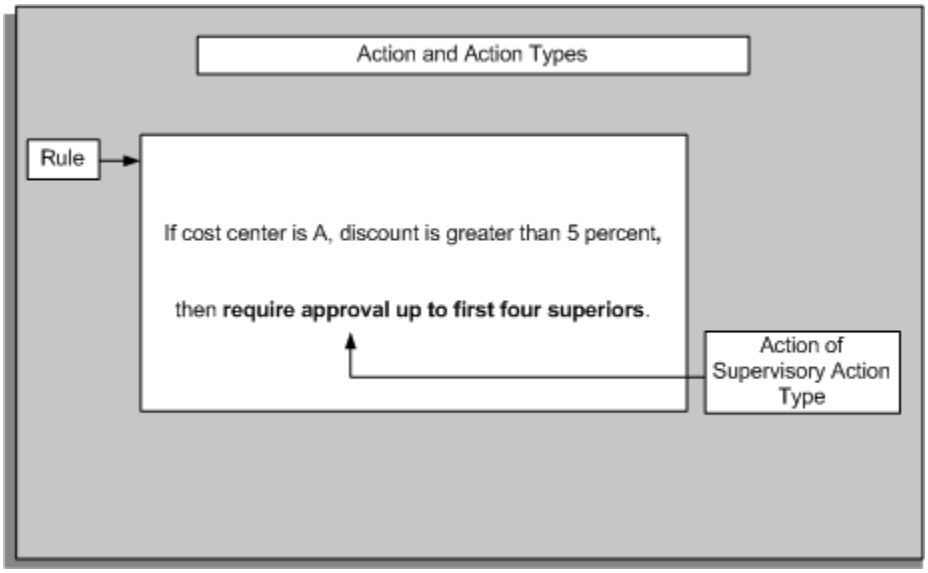
Actions

Actions

Overview

An *action* is an instruction to AME to modify a transaction's approval process in the manner you specify. An approval rule's *then* part (its *consequent*) consists of one or more actions. Actions that modify a transaction's approver list are Approval actions or simply Approvals. Actions that generate a production (a variable-name or a value pair) are Production actions.

The following graphic illustrates an action in the approval rule:



Typically, AME provides all the action types and actions your implementation will require. There are many action types to choose from, so even if you don't expect to create a new action type or action, you must review the descriptions of the action types, to ensure you understand how to translate your business rules into appropriate actions.

Action Types

Every action belongs to an action type. An action type is a collection of actions having similar functionality. For example, actions in the absolute-job-level action type all require approvals up to a certain job level in the HR supervisory hierarchy. The main difference among the actions are the job level they require.

AME includes a set of predefined action types, many of which enable you to ascend commonly used organizational hierarchies. If none of the predefined action types meet your organization's requirements, you need a custom action type. An AME user with administrative privileges must use the Actions tab to define a new action type. See

Chapter 2 of the *Oracle Approvals Management Developer's Guide* for details.

Action Type Properties

An action type has the following properties:

Name

An action-type name is a string up to 50 bytes in length.

Action-Type Handlers

An action-type handler is a PL/SQL package that implements the action type's functionality. AME calls the package's procedures at run time to determine how to process applicable actions of the corresponding action type.

All action types, except production action types, have handlers specified that are used at run time. Production action types require a handler name to be specified but is not used at runtime. AME does not use the handler name.

See Chapter 2 of the *Oracle Approvals Management Developer's Guide* for more details about action-type handlers.

Rule Types

An action type's allowed rule type is the type of rule that can use the action type's actions. Every action type has exactly one allowed rule type.

Action-Type Descriptions

An action-type description is a string up to 100 bytes in length. It describes the functionality shared by the action type's actions.

Allowed Approver Types

Action types with an allowed rule type other than substitution or production have one or more allowed approver types. These are the types of approvers that the action type can generate. Action types that use approver groups allow all approver types.

Order Numbers

An action type has an order number, which is always a counting number. The set of all action types for a given allowed rule type is ordered by the action types' order numbers. The order numbers start at one and ascend to at most the number of action types for the allowed rule type. Order numbers are not necessarily unique, so the highest order number may be lower than the number of action types.

AME uses the action types' order numbers to decide the order in which the action types of the applicable rules of a given rule type are processed at run time. This affects the order in which AME notifies the approvers. See Parallel Approval Process, page 2-8 for details.

Chain of Authority Ordering Modes

Action types that have the chain of authority allowed rule type specify an ordering

mode for the chains of authority generated by the action type's actions. When a single action type generates several chains of authority for a single transaction, the ordering mode tells AME how to order the chains of authority. This affects the notification order of the approvers in the chains of authority. See Parallel Approval Process, page 2-8 for details.

> **Note:** The chain of authority rule type includes both the list creation and list creation exception rule types.

Voting Methods

Action types that have the chain of authority allowed rule type specify a voting methods for the chains of authority generated by the action type's actions. This affects the notification order of the approvers in the chains of authority, and how AME treats their responses. See Parallel Approval Process, page 2-8 for details.

Required Attributes For Action Types

Some action types require that a transaction type define a use for one or more attributes, to use actions of that type in its rules. These attributes are the action type's required attributes. An action type typically uses the values of its required attributes to calculate how the action type's actions should contribute to a transaction's approval process. The Approvals tab lists each approval type's required attributes.

Action Properties

An action has the following properties:

Action Descriptions

An action description is a string at most 100 bytes long. It must have the form of a complete imperative sentence expressing the action's effect on a transaction's approval process. When you create an action for a predefined action type, the new description must resemble as much as possible those of existing actions of the same type. For example, a supervisory-level action that requires the chain of authority to ascend the supervisory hierarchy exactly 15 approvers must have the description *Require approvals up to the first 15 superiors*.

Action descriptions are of two types:

- Static

  A static action description is a fixed string that AME provides for a predefined action, or that you enter upon creating an action. Most predefined actions have static descriptions.

- Dynamic

A few action types must generate descriptions for their actions at the moment the description is displayed. These actions have dynamic action descriptions. An action type that produces dynamic action descriptions does so by executing an SQL query that is registered with AME when the action type is first created. The query must return a single row that is a string at most 100 bytes long. It may use either or both of the bind variables ':parameterOne' and ':parameterTwo'. AME assigns these variables the values of the action's parameters. See Chapter 2 of the *Oracle Approvals Management Developer's Guide* for details.

Action Parameters

An action's parameters tell the action type's handler how to process the action. Each action type has its own approval-parameter syntax rules. You must adhere carefully to these rules when you create a new action for a predefined action type.

Predefined Action Types

This topic provides the following information about each action type:

- What approver type(s) the action type uses
- What attributes it requires
- How the action type works
- The syntax and semantics of the action's parameters

The action types are sorted by allowed rule type and are broadly categorized as:

- Chain of authority action types
- List Modification action types
- Substitution action types
- Production action types

Chain of Authority (List Creation and List-Creation Exception) Action Types

The predefined action types that generate chains of authority do so by ascending the HR supervisory hierarchy or HR position hierarchy, or by treating an approver group as a chain of authority. In most cases one or more required attributes determine the first approver in the chain, and the particular action determines where the chain ends.

Action Types for the HR Supervisory Hierarchy

All of the action types that ascend the HR supervisory hierarchy require the TOP_SUPERVISOR_PERSON_ID attribute. This attribute's value should be the person

ID of the employee at the top of the hierarchy, typically the CEO of a corporation. When this attribute has the appropriate value, a chain of authority can end with the top supervisor without AME raising an exception, even if the rules require a longer chain. If, however, the TOP_SUPERVISOR_PERSON_ID attribute does not identify the top supervisor, or if an employee is not the top supervisor but has no supervisor assigned to them, then AME raises an exception if it needs to generate a longer chain than the hierarchy allows.

Absolute-Job-Level Action Type

The absolute-job-level action type generates a chain of authority by ascending the HR supervisory hierarchy starting at a given approver and continuing until an approver with a sufficient job level is found. The HRMS per_all_assignments_f, per_jobs, and per_all_people_f tables and view define the supervisory hierarchy.

First Approver

By default, the first approver (starting point) in an absolute-job-level chain is the supervisor of the person identified by the required number attribute TRANSACTION_REQUESTOR_PERSON_ID. If the required number attribute JOB_LEVEL_NON_DEFAULT_STARTING_POINT_PERSON_ID has a non-null value, however, the value identifies the first approver in the chain.

> **Note:** An integrating application can override both of these first approvers via AME's programming interface. See Appendix A of the *Oracle Approvals Management Developer's Guide* for details.

Final Approver

The absolute-job-level action type's ascent up the supervisory hierarchy stops when it reaches one or more approvers having a sufficient job level. A job level is a value in the authority_level column of the Oracle HRMS table per_jobs. What *sufficient* means varies according to following:

- The job level required by the action. AME predefines actions for job levels one through 10.
- Whether the action is an at-most action or an at-least action.
- The value of the required attribute INCLUDE_ALL_JOB_LEVEL_APPROVERS.

If one or more approvers in the appropriate supervisory chain have the required job level, the chain's final approver does not depend on whether the action is an at-most action or an at-least action. In this case, if INCLUDE_ALL_JOB_LEVEL_APPROVERS is false, the chain includes and ends with the first approver with the required job level. If INCLUDE_ALL_JOB_LEVEL_APPROVERS is true, then the chain includes and ends with the last approver with the required job level.

If no approver in the appropriate supervisory chain has the required job level, and the action is an at-most action, then the chain includes and ends with the last job level

before the first approver having a job level exceeding the required job level. If the action is an at-least action, then the chain instead ends with the first job level exceeding the required job level. In either case, whether the action type includes all of the approvers at the final job level depends on the value of INCLUDE_ALL_JOB_LEVEL_APPROVERS.

Two job-level actions may combine in an interesting way. Suppose, for example, that one rule requires approvers up to at least job level five, and another requires approvers up to at most job level six. If the hierarchy skips from job level four to job level seven, then AME can only satisfy both rules by including in the chain of authority one or all of the approvers with job level seven (depending on the value of INCLUDE_ALL_JOB_LEVEL_APPROVERS). In this case, the at-least action is more stringent than the at-most action, even though the at-most action requires a higher nominal job level. AME always satisfies the most stringent action, so be aware that job-level gaps can have surprising consequences.

Required Attributes

Following are the required attributes for this action type:

- JOB_LEVEL_NON_DEFAULT_STARTING_POINT_PERSON_ID
- INCLUDE_ALL_JOB_LEVEL_APPROVERS
- TOP_SUPERVISOR_PERSON_ID
- TRANSACTION_REQUESTOR_PERSON_ID

Action Parameters

The parameters of absolute-job-level actions have the syntax

`n{+,-}`

where *n* is a positive integer representing the job level, a plus sign indicates an at-least action, and a minus sign indicates an at-most action. For example, the action with the description 'Require approvals up to at least level 1.' has the parameter '1+'.

Difference between At Least and At Most

At most specifies an action that has to be satisfied so that ascending the hierarchy requires an approver to be returned with a value "at most" to the level specified. For instance if the hierarchy has approvers with job level 2,3,5,6 and the rule stated Absolute job level require approvers up to at-most level 4 then the approvers with job level 2 & 3 would be returned but not the approver at level 5 even though at most level 4 was specified. However if the action is defined as "at-least" level 4 then approver 2,3,5 would be returned since we require an approver to at least level 4 and since this job level didn't exist the approver with level 5 was returned.

Relative-Job-Level Action Type

The relative-job-level action type behaves exactly like the absolute-job-level action type, except its actions' parameters represent the difference between an approver's job level and the requestor's job level. For example, the relative-job-level action with the

parameter '3+' ascends the chain of authority until it reaches a job level at least three levels higher than the requestor's job level. If the requestor has a job level of two, the action would ascend at least to job level five.

Required Attributes

The action type has the same required attributes as the absolute-job-level action type, and uses them in the same way. Following are the required attributes for this action type:

- INCLUDE_ALL_JOB_LEVEL_APPROVERS
- JOB_LEVEL_NON_DEFAULT_STARTING_POINT_PERSON_ID
- TOP_SUPERVISOR_PERSON_ID
- TRANSACTION_REQUESTOR_PERSON_ID

Action Parameters

The action parameters have the same syntax, with the integer value described as explained above.

Manager-then-Final-Approver Action Type

The manager-then-final-approver action type is another variant of the absolute-job-level action type. Instead of requiring approval from every person in an ascent up the supervisory hierarchy, this action type only includes the first and final approvers.

Required Attributes

The action type's required attributes are the same as those of the absolute-job-level action type. Following are the required attributes for this action type:

- INCLUDE_ALL_JOB_LEVEL_APPROVERS
- JOB_LEVEL_NON_DEFAULT_STARTING_POINT_PERSON_ID
- TOP_SUPERVISOR_PERSON_ID
- TRANSACTION_REQUESTOR_PERSON_ID

Action Parameters

The action type's action-parameter-syntax rules are the same as those of the absolute-job-level action type.

Final-Approver-Only Action Type

The final-approver-only action type is also a variant of the absolute-job-level action type. Instead of requiring approval from every person in an ascent up the supervisory hierarchy, this action type only includes the final approver.

Required Attributes

The final-approver-only action type's required attributes are the same as those of the absolute-job-level action type. Following are the required attributes for this action type:

- INCLUDE_ALL_JOB_LEVEL_APPROVERS
- JOB_LEVEL_NON_DEFAULT_STARTING_POINT_PERSON_ID
- TOP_SUPERVISOR_PERSON_ID
- TRANSACTION_REQUESTOR_PERSON_ID

Action Parameters

The action type's action-parameter-syntax rules are similar. The syntax is

```
{A,R}n{+,-}
```

For example, 'A3+' means the final approver should have an absolute job level at least three levels above the requestor's.

Dual-Chains-of-Authority Action Type

Some transactions require approval from two chains of authority, each starting with a different approver in the supervisory hierarchy. For example, an employee transfer may require approval from chains of authority starting at the employee's old and new supervisors. In such cases, use the dual-chains-of-authority action type.

First Approvers

The dual-chains-of-authority action type requires two attributes: FIRST_STARTING_POINT_PERSON_ID and SECOND_STARTING_POINT_PERSON_ID. These attributes' values identify the first approver in each chain of authority generated by the action type, so they must always have non-null values when rules using the action type apply to a transaction.

At Least One Action per Chain

The dual-chains-of-authority action type is unique in requiring that multiple actions apply to a transaction, at least one for each chain of authority generated by the action type. Typically dual-chains actions apply to a transaction in pairs, with one action in the pair for the first chain and the other for the second chain. In such cases, make each pair of actions part of the same rule. For example:

```
If
  TRANSACTION_CATEGORY in {TRANSFER}
then
 Require approvals at most 3 levels up in the first chain.
 Require approvals at least 2 levels up in the second chain.
```

Final Approvers

As usual, when multiple dual-chains actions for the same chain apply, AME determines which has the most stringent requirement for that sub chain, and enforces that requirement.

Required Attributes

Following are the required attributes for this action type:

- INCLUDE_ALL_JOB_LEVEL_APPROVERS
- FIRST_STARTING_POINT_PERSON_ID
- SECOND_STARTING_POINT_PERSON_ID
- TOP_SUPERVISOR_PERSON_ID
- TRANSACTION_REQUESTOR_PERSON_ID

Action Parameters

The parameters of dual-chains-of-authority approvals have the syntax:

```
{1,2}{A,R}n{+,-}
```

The first integer (a one or a two) identifies the (first or second) chain. The letter (an 'A' or an 'R') indicates whether n (a positive integer) should be interpreted as an absolute or a relative job level. The plus or minus sign is interpreted as it would be for the absolute- or relative-job-level action type, as appropriate. For example, the action with the description 'Require approvals at most 8 levels up in the second chain.' has the parameter '2R8-'.

Line-Item Job-Level Chains-of-Authority Action Type

The line-item job-level chains-of-authority action type is included in the current version of AME for backwards compatibility. It generates one chain of authority for each line item. These chains of authority are associated with the header item (as in previous versions of AME, which did not generate separate approver lists for items in subordinate item classes), not with the line items, in the transaction's approver list.

First Approvers

The line-item-level required attribute LINE_ITEM_STARTING_POINT_PERSON_ID identifies the first approver in each chain of authority. Because this attribute has a distinct value for each line item, each chain can be unique, and can start at a distinct job level.

Final Approvers

The line-item job-level chains-of-authority action type generates chains in the same way as the absolute-job-level action type. Because the same actions generate all of the chains, the chains all end at the same job level. The required attributes INCLUDE_ALL_JOB_LEVEL_APPROVALS has the same function for this action type as for that one.

Required Attributes

Following are the required attributes for this action type:

- INCLUDE_ALL_JOB_LEVEL_APPROVERS
- LINE_ITEM_STARTING_POINT_PERSON_ID
- TOP_SUPERVISOR_PERSON_ID

Action Parameters

The line-item job-level chains-of-authority action type's action parameters have the same syntax and interpretation as for the absolute-job-level action type.

Supervisory-Level Action Type

The supervisory-level action type ascends the HR supervisory hierarchy, generating a chain that has a fixed number of approvers in it. Note that there is no precise correlation between the number of supervisors ascended and the job level. Job levels can be skipped, or can occur several times, in a supervisory chain of authority.

First Approver

By default, the first approver in the chain is the supervisor of the employee identified by the required attribute TRANSACTION_REQUESTOR_PERSON_ID. However, if the required attribute SUPERVISORY_NON_DEFAULT_STARTING_POINT_PERSON_ID is non-null, the chain starts with the approver identified by this attribute instead. (As always, an integrating application can override both of these first approvers via AME's programming interface. See Appendix A of the *Oracle Approvals Management Developer's Guide* for details.)

Final Approver

The chain of authority generated for a given action contains the number of approvers specified by the action's parameter, when it is possible to ascend the HR supervisory hierarchy that many approvers. When the hierarchy contains fewer approvers than the action requires, the result depends on whether the action is an at-most action. If it is, and if the last approver is identified by the required attribute TOP_SUPERVISOR_PERSON_ID, the chain ends with this approver. However, if this attribute's value is null, or if the action is not an at-most action, AME raises an exception.

Required Attributes

Following are the required attributes for this action type:

- TOP_SUPERVISOR_PERSON_ID
- SUPERVISORY_NON_DEFAULT_STARTING_POINT_PERSON_ID
- TRANSACTION_REQUESTOR_PERSON_ID

Action Parameters

Supervisory-level actions have parameters of the form

n[-]

where n is a positive integer indicating how many approvers the chain should contain, and the optional minus sign makes the action an at-most action when present.

Action Types for the HR Position Hierarchy

There are two action types for the HR position hierarchy (residing in per_pos_structure_elements, per_pos_structure_versions, and per_position_structures). The difference between them is how they decide where to end a chain of authority.

hr position level action type requires the TOP_POSITION_ID attribute, which is wholly analogous to the TOP_SUPERVISOR_PERSON_ID attribute for the action types that ascend the HR supervisory hierarchy.

Both action types also require the NON_DEFAULT_POSITION_STRUCTURE_ID attribute. By default, the chain of authority ascends the primary position structure of the business group of the transaction requestor. However, if NON_DEFAULT_POSITION_STRUCTURE_ID is non-null, the chain will follow the position structure identified by the value instead.

'hr position' Action Type

The 'hr position' action type generates a chain of authority ending with a specified position. Note that position actions have dynamic descriptions.

First Approver

By default, a position chain of authority starts with the position above the position identified by the required attribute TRANSACTION_REQUESTOR_POSITION_ID. If however the required attribute NON_DEFAULT_STARTING_POINT_POSITION_ID is non-null, the chain starts with this position instead.

Final Approver

The final approver is the position specified by the action's parameter.

Required Attributes

Following are the required attributes for this action type:

- TRANSACTION_REQUESTOR_POSITION_ID
- NON_DEFAULT_POSITION_STRUCTURE_ID
- NON_DEFAULT_STARTING_POINT_POSITION_ID

Action Parameters

A position action's parameter is the wf_roles.name value of the position at which the chain should end. Currently positions in wf_roles have name values such as 'POS:471', where the positive integer following the colon is a position ID.

Position-Level Action Type

The position action type generates a chain of authority having a specific length.

First Approver

The position-level action type starts its chains the same way the position action type does.

Final Approver

The position-level action type ascends the position hierarchy a fixed number of positions (in analogy with the supervisory-level action type for the HR supervisory hierarchy).

Required Attributes

Following are the required attributes for this action type:

- TOP_POSITION_ID
- TRANSACTION_REQUESTOR_POSITION_ID
- NON_DEFAULT_POSITION_STRUCTURE_ID
- NON_DEFAULT_STARTING_POINT_POSITION_ID

Action Parameters

The parameters of position-level actions have the syntax

n+

where n is a positive integer representing the number of positions the action requires.

Actions Types for Approver groups

There are three predefined action types that use approver groups, one for each sub-list in an item's approver list. All three approval-group action types require the ALLOW_EMPTY_APPROVAL_GROUPS attribute. When this attribute is false, AME raises an exception if a group's has no members at run time.

Approver-Group Chain of Authority Action Type

The approver-group chain of authority action type treats an approver group as a chain of authority. AME ignores the approver group's voting regime and uses the action type's instead. When the action type uses the serial voting regime, the serial order is consistent with the group members' order numbers.

Required Attributes

ALLOW_EMPTY_APPROVAL_GROUPS is the required attribute for this action type.

Pre- and Post-Chain of Authority Action Types

The pre-chain of authority action type inserts approver groups before the authority sub-list of an item's approver list. The post-chain of authority action type inserts approver groups after the authority sub-list of an item's approver list. The notification order is not necessarily the same as the order in the approver list. Both use approver groups' voting regimes.

Required Attributes

ALLOW_EMPTY_APPROVAL_GROUPS is the required attribute for this action type.

List-Modification Action Types

There are two predefined list-modification action types, and they have opposite effects. One grants signing authority to someone who normally lacks it; the other revokes the signing authority of someone who normally has it.

Final-Authority Action Type

The final-authority action type effectively grants a target approver signing authority in contexts where they usually lack. It does so by truncating each chain of authority containing the target approver, after the approver. (If the target approver forwards without approving, the action type truncates the chain of authority after the last forwardee following the target approver.) The action type has only one action, which has a null parameter.

Non-Final-Authority Action Type

The non-final-authority action type effectively revokes a target approver's signing authority in contexts where they normally have it. It does so by extending a chain of authority ending with the target approver beyond the target approver, up to some absolute or relative job level, depending on the particular action's requirement. Because the actions depend on job levels, the target approver must be an HR employee, and the HR implementation must use job levels.

Non-final authority actions' parameters have the syntax

```
{A,R}n{+,-}
```

The 'A' or 'R' indicates whether n (a positive integer) represents an absolute or relative job level. The plus or minus sign is interpreted as for the job-level action types. For example, the action having the description 'Require approvals at most 2 levels up.' has the parameter 'R2-'

Substitution Action Type

There is just one predefined substitution action type. It replaces the target approver with the approver identified by the substitution action. The parameters for substitution approvals are wf_roles.name values. When you create or edit a substitution action, the

AME user interface lets you select an approver type and then query for an approver, so you don't have to create or edit the parameter or action-description directly.

Production Action Type

There is just one predefined production action type. Each production action causes AME to output to the integrating application the variable-name/value pair in the action's parameters. The variable name and value are strings up to 50 bytes long.

Production Levels

If a production action is used by a rule that also has one or more approver-generating actions, the production is associated with each approver generated by the rule's approver-generating actions. Such actions are approver-level productions. Production actions in production rules are associated with the transaction as a whole, and so are transaction-level productions

Choosing an Action

Choosing the right action for a rule requires three choices. The subsections below explain them.

1. Choose a rule type.

   See Rule Types, page 7-2 for a complete explanation of the rule types. Here are brief explanations of when to use each:

   - Use list-creation rules to generate chains of authority.
   - Avoid using list-creation exceptions. Use list-creation rules and rule-use priorities instead. See Rule Priorities, page 7-7 and Using Rules, page 7-9 for details about rule-use priorities.
   - Use pre- and post-chain of authority rules to let groups of subject-matter experts approve a transaction before or after management.
   - Use list-modification rules to grant unusual signing authority, or to revoke usual signing authority, in special cases.
   - Use substitution rules to replace one approver with another in special cases, or for short periods of time (such as vacations or leaves of absence).
   - Use combination rules when several of your business rules apply to exactly the same cases. There are two kinds of combination rules: those combining multiple list-creation, pre-chain, and/or post-chain actions; and those combining multiple list-modification and/or substitution actions.

- Use production rules when an integrating application uses AME productions. Consult your integrating application's documentation to learn whether the application uses productions.

2. Choose an approver type

   Most of the rule types have just one predefined action type available to them. There are two action types available to list-modification rules; which to use depends on whether you want to grant or revoke signing authority. List-creation rules have many action types available to them. Before you can choose one of them, you must decide which approver type should occur in the chains of authority your list-creation rule will generate. AME current supports three approver types: HR employees (sometimes called persons/HR people in the AME UI), HR positions, and FND (Oracle Applications) users.

3. Choose an action type.

   Once you have chosen an approver type for your list-creation rule, you can choose an action type that generates a chain of authority containing that approver type, and having the structure you want. The following tables list the action types that generate chains of authority, sorting them by approver type, and indicating the structure of the chains of authority they generate.

Action Type	Chain Structure for HR Employee (Person) Approver Type
absolute job level	The chain ends with one or more approvers at, just under, or above a fixed job level.
relative job level	The chain ends with one or more approvers at, just under, or above a job level a fixed number of levels above the requestor's job level.
manager then final approver	The chain includes the first and last approvers in the chain that would be generated by a similar absolute or relative-job-level action.
final approver only	The chain includes just the last approver in the chain that would be generated by a similar absolute or relative-job-level action.
dual chains of authority	There are two chains of authority, each with its own first approver, and each having an absolute- or relative-job-level structure.

line-item job-level chains of authority	There is one chain of authority per line item. All of the chains are in the header item's approver list. To generate line-item-specific chains of authority within the line items' approver lists, use one of the other action types for HR employees, and associate your rule with the line-item item class.
supervisory level	The chain contains a fixed number of HR employees.

Action Type	**Chain Structure for HR Position Approver Type**
hr position	The chain ends with an approver having a fixed position.
hr position level	The chain contains a fixed number of positions.

Action Type	**Chain Structure for All Available Approver Types**
approval-group chain of authority	The approver group allows all available approver types.

Creating an Action Type

Creating an action type involves a significant programming exercise, as well as registering the new action type with AME. These tasks are beyond the scope of this guide. See Chapter 2 of the *Oracle Approvals Management Developer's Guide* for details.

To create an action type:

Use the Create New Action Type page.

1. Click the Action Types tab to display the Action Types page. If you are navigating from the Business Dashboard, then select the required transaction type in the Approval Process Setup available in the Business Dashboard and click the Action Types link.
2. Click Use Existing Action Type to open the Use Existing Action Type page.
3. Click Create to open the Create New Action Type page.
4. Enter a name.

5. Enter the action type's handler to implement the action type's functionality. AME uses the handler information to determine how to process applicable actions of the corresponding action type.

6. Select the rule type to specify the allowed type of rule that can use the action type's actions.

7. Enter the action type description to explain the functionality shared by the action type's actions.

8. Select the allowed approver type to enable the action type to generate the specified types of approvers.

9. Select the dynamic action description. If you select Query Generate, then enter the action description query and click Validate to ensure the query is well formed.

10. Click Apply to make the action type available to all the transaction types.

Editing an Action Type

To update an action type:

1. Click the Action Types tab to open the Action Types page.

2. Click Use Existing Action Type to display the available action types.

3. Select the action type and click the Update icon to edit.

4. Change the action type's properties as appropriate.

5. Click Apply to submit your changes.

> **Note:** You cannot edit predefined action types' properties; you can only add and delete user-defined actions.

Deleting an Action Type

To delete an action type:

1. Click the Action Types tab to open the Action Types page.

2. Click Use Existing Action Type to display the available action types.

3. Select the action type and click the Delete icon.

4. Confirm the deletion when prompted.

   You can delete several action types at once.

   Note: You cannot delete predefined action types.

Using an Existing Action Type

To use an existing action type into your current transaction type:

Use the Use Existing Action Type page.

1. Click Use Existing Action Type to open the Use Existing Action Type page.
2. Select an action type and click Continue.
3. Review the attributes that appear.

   Note: The Use Existing Action Types Review page displays the required attributes that are not currently available but will appear in your transaction type based on the selected action type.

4. Click Finish to use the selected existing action type in your current transaction type.

Creating an Action

To create an action for an existing action type:

Use the Create Action page.

1. Review the action type's parameter syntax and semantics rules.
2. Click the Action Types tab.
3. Select the action type to which you want to add actions.
4. Click Create to open the Create Action page.
5. Enter the new action's parameter(s) and description.
6. Click Apply to add the action to the selected action type.

   Note: You never need to add actions to the approver-group action

types. AME creates these actions automatically when an approver group is created.

Editing an Action

To edit an action:

Use the Update Action page.

1. Click the Action Types tab to open the Action Types page.
2. Select the action type whose action you want to edit.
3. Select the action and click the Update icon to edit.
4. Change the action's properties as appropriate.
5. Click Apply to submit your changes.

> **Note:** You cannot edit predefined actions, or approver-group actions. AME maintains approver-group actions automatically.

Deleting an Action

To delete an action:

Use the Action Types page.

1. Click the Action Types tab to open the Action Types page.
2. Select the action type whose action you want to delete.
3. Select the action and click the Delete icon.
4. Confirm the deletion when prompted.

   You can delete several actions at once.

> **Note:** You cannot delete a predefined action, but you can delete an action that has been added to a predefined action type. Also, you cannot delete approver-group actions. AME deletes these automatically when you delete the approver group.

6

Approver Groups

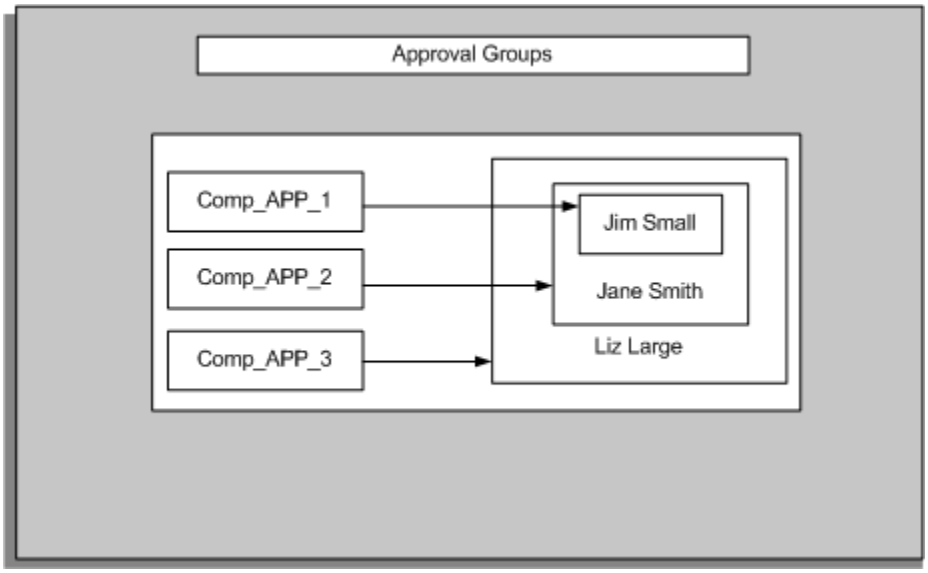
Approver Groups

Overview

An approver group can either be an ordered set of one or more approvers (persons and/or user accounts) or it can be a list, which is dynamically generated at rule evaluation time. A typical pre- or post-approval rule adds an approver group's members (in order) to a transaction's approver list. Typically approver groups represent functional approvers outside a transaction's chain of authority, such as human-resource management and internal legal counsel, that must approve a transaction before or after management has done so. However, it is possible to insert an approver group into a chain of authority if required.

When you create an approver group, AME creates for you the corresponding actions of the pre- and post-approval action types automatically. These actions are available to all transaction types.

You can nest approver groups or in other words it is possible to make one approver group a member of another approver group. The following graphic illustrates the example discussed below:



For example, a purchasing department might define the following groups to handle post-approvals of computer-hardware purchases:

Example
COMP_APP_1 = {Jim Small} COMP_APP_2 = {COM_APP_1, Jane Smith} COMP_APP_3 = {COMP_APP_2, Liz Large}

AME would evaluate the membership of COMP_APP_3 to be (Jim Small, Jane Smith,

Liz Large}, in that order.

You can add all the members of an existing group to the new group to create a nested approver group. To create a nested approver group, select the approver type as Nested Group and select the required group in the Create New Approver Group page. The select list that appears on the Create New Approver Group page only includes a group under three conditions:

- It is not the target group itself.
- It does not contain the target group (either explicitly or implicitly).
- It is not contained in the target group already.

The first two of these requirements prevent an infinite loop in the resolution of nested groups. The third prevents redundant group members only in the narrow sense that if group A already contains group B as a member, you cannot add group B to A again. However, if group A contains B, and group C also contains group B, you can add C to A. In this case, the ordering of A's members would place B's members before C's members other than those in B, in A's membership list. Thus:

Example
B = {1, 2} C = {3, 4, B} A = {B, C} = {{1, 2}, {3, 4, B}} = {{1, 2}, {3, 4, {1, 2}}} = {1, 2, 3, 4}

Nested approver groups let you build an approvals matrix using AME approver groups. For example, the purchasing department defining the computer-hardware approver groups above might define three corresponding post-approval rules:

Example
If ITEM_CATEGORY in {COMPUTER_HARDWARE} and ITEM_AMOUNT <= 1,000 USD then require post-approval from the COMP_APP_1 group.

Example
If ITEM_CATEGORY in {COMPUTER_HARDWARE} and 1,000 USD < ITEM_AMOUNT <= 10,000 USD then require post-approval from the COMP _APP_2 group.

Example
If ITEM_CATEGORY in {COMPUTER_HARDWARE} and 10,000 USD < ITEM_AMOUNT then require post-approval from the COMP_APP_3 group.

These rules effectively define a hierarchy of per-item dollar-amount signing authorities for three subject-matter approvers. You can predefine a hierarchy of nested approver groups containing no members other than nested groups, along with a set of approval rules like those above; and your customers can easily populate the groups with person or user approvers upon installation.

It is possible to nest a dynamic group in a static group, but it is not possible to nest either a dynamic or static group in a dynamic group. Dynamic groups can only include persons and users.

AME maintains a non-recursive approver list for each approver group, so that the engine does not need to evaluate group nestings at run time. Rather, the engine fetches

all static members and all dynamic members' queries in a single query, and then executes the dynamic members' queries in turn, substituting the results into the membership list. This means you can nest groups to arbitrary depth without impacting AME performance adversely, but you should take care to limit the number of dynamic groups nested in a single group.

Empty Approver groups

The action types that use approver groups share the required boolean attribute **ALLOW_EMPTY_APPROVAL_GROUPS**. When this attribute has the value 'true', AME allows an approver group not to have any members at run time. When the attribute has the value 'false', AME raises an exception if an approver group does not have any members.

Deciding When to Use an Approver Group

There are action types that let you use approver groups for pre-approvals, chain of authority approvals, and post-approvals. The following factors enable you to decide when to use an approver group:

- Using Approver groups for Pre- and Post-Approvals

  Typically approver groups represent functional or subject-matter-expert approvers outside a transaction's managerial chain of authority, such as human-resource management and internal legal counsel that must approve a transaction before or after management has done so.

- Using Approver groups for Chain of Authority Approvals

  Sometimes you need AME to generate chains of authority containing several approver types, or containing an approver type that lacks a hierarchical structure. In these cases you can use nested approver groups to construct an approver hierarchy, and then use the hierarchy with the approver-group chain of authority action type.

Approver Group Properties

Approver groups have several properties:

Name

An approver group's name can be up to 50 bytes long. Typically an approver group's name should avoid referring to a specific transaction type or approver sub-list (pre-approval, chain of authority, post-approval). This is because an approver group can be used in any transaction type and any sub-list, even if it was originally created for a specific transaction type or sub-list. Likewise, an approver group's name should avoid indicating whether the group's membership is (currently) dynamic or static, because this property might change. Instead, an approver group's name should reflect the

group's position or function in the organization. For example, 'Office-Furniture Purchasing' is better than 'Office Furniture Requisitions Dynamic Pre-Approvers'.

Description

An approver group's description can be up to 100 bytes long. The same caveats that apply to approver groups' names apply to their descriptions.

Order Number

Every approver group has an order number. Approver-group order numbers determine how the approver groups in the same sub-list are ordered. Do not confuse an approver group's order number with the order numbers of the group's members. (See "Membership" below.)

Voting Regime

An approver group has a voting regime assigned to it. A voting regime determines the order in which a group's members are notified, and how the group makes approval decisions. The following four regimes are available:

- Serial Voting

  In serial voting, the members are notified one after the other, in an order consistent with the members' order numbers. AME breaks ties arbitrarily. All members must approve for the group to approve.

- Consensus Voting

  In consensus voting, the members are notified in parallel. All members must approve for the group to approve. Members' order numbers are ignored in this instance.

- First-Responder-Wins Voting

  In first-responder-wins voting, the members are notified in parallel. The response of the first member to respond to the notification requesting approval becomes the group's approval decision. AME logs the responses of the remaining group members, but otherwise ignores their responses. Members' order numbers are ignored in this instance.

- Order-Number Voting

  In order-number voting, the members are notified in the order of their order numbers. Members having the same order numbers are notified in parallel. All members must approve for the group to approve.

Active List

An approver group can have two membership lists. Only one of the lists is active (in force) at any given time. You only need to define one list or the other, but you may define both. The two membership lists are:

- Static Membership List

  You choose an approver group's static members using AME's approver-query wizard, while creating or editing the group.

- Dynamic Membership List

  AME generates an approver group's dynamic members by executing an SQL query you enter while creating or editing the group. Each row fetched by the query must have one of two forms.

  First, it can consist of one of the approver-type labels 'person_id' and 'user_id', followed by a colon, followed by a valid ID for an appropriate approver type. Note: The case of the labels is not important so 'Person_id' is equivalent to 'person_id.'

  Second, it can consist of wf_roles role type (orig_system), followed by a colon, followed a valid wf_roles.orig_system_id.

  The order of the rows returned by the query determines the dynamic members' order numbers, at run time. The query can reference the transaction-ID placeholder : **transactionId**.

Creating an Approver Group

You create, edit and delete approver groups using the Approver Groups tab.

To create an approver group:

Use the Create New Approver Group page.

1. Click the Approver Groups tab to display the Approver Groups page. If you are navigating from the Business Dashboard, then select the required transaction type in the Approval Process Setup available in the Business Dashboard and click the Approver Groups link.

2. Click Create to open the Create New Approver Group page.

3. Enter the group's name and description.

   Note: Avoid referring to a specific transaction type or approver sub-list such as pre-approval, chain of authority, post approval as you can use the approver group in any transaction type or approver sub-list.

4. Enter the order number to determine how the approver groups in the same sub-list are ordered.

5. Select the voting method to determine the order in which AME notifies the group's

members and how the group makes approval decisions.

6. Select the group's use type to determine the group's membership list. If you select the dynamic use type, then enter an SQL query and click Validate to ensure the query is well formed. If you select the static use type, then you must enter members for the static approver group in the Group Members region. You can add all the members of an existing group to the new group to create a nested approver group. To create a nested approver group, select the approver type as Nested Group and select the required group.

   Note: Oracle strongly recommends that SQL queries used to retrieve information must use only base tables and not secure views. The SQL query must also not include any context-sensitive logic.

   The SQL query for a dynamic approver group can generate different approvers for each subordinate item in a transaction. To enable this feature, you must ensure the dynamic query for the dynamic approver group includes item class and item ID variables in addition to the transaction ID. For example, your query can be:

```
select
  'PER:'||person_id from invoice_table
  where transaction_id = :transactionId
  and item_class = :itemClass
  and item_id = :itemId
```

7. Click Create Another to create another approver group.

8. Click Apply to add the approver group to your transaction type and to make it available for other transaction types.

   Note: The new approver group will appear on the list in the order of its approver-group order number. Note that when you create an approver group, AME creates for you the corresponding approver-group actions automatically.

Editing an Approver Group

There are several kinds of changes you can make to an approver group. You can change the description of an approver group and delete group members. You cannot update the approver group name.

To update an approver group:

Use the Update Approver Group page.

1. Click the Approver Group tab.
2. Select the Update icon for approver group you want to edit.
3. Change the properties of interest.

   To add members to an approver group's static membership, click Add Another Row in the Group Members region.

   To delete one or more members of an approver group's static membership, click the Delete icon next to the approver(s) and confirm the deletion.
4. Click Apply to submit your changes.

Deleting an Approver Group

You can delete an approver group only if it is not used in any current or future rule and it is not a predefined group.

To delete one or more approver groups:

Use the Approver Groups page.

1. Click the Approver Groups tab.
2. Select the Delete icon for the approver group you want to delete.

   Warning: If an inactive rule is using the deleted approver group, then you cannot reinstate that rule.
3. Confirm the deletion when prompted.

7

Rules

Rules

Overview

Creating rules and rule use are the main steps in the AME implementation process. Rarely will an organization's business rules match any rules that are predefined with a transaction type. Instead, you must translate the business rules you documented into AME rules yourself.

Rule associate one or more conditions with an approval in an if-then statement. Before you can create rules, you must create conditions for the rules to use. You may need to create (or have a system administrator create) some custom attributes and/or approvals. You may also need to create some approver groups. Thus, while creating rules is your ultimate goal, it is also the last thing you do when you set up AME.

Rule Types

There are eight rule types in AME. Seven of them provide rules that participate in generating transactions' approver lists; these are the approver-generating rule types. The eighth generates productions.

- List-creation rules
- List-creation exceptions
- List-modification rules
- Substitutions
- Pre-list approval-group rules
- Post-list approval-group rules
- Combination Rules
- Production Rules

Different rule types use different condition and action types, and have different effects on a transaction's approver list.

List-Creation Rules

List-creation rules generate chains of authority. They use action types that ascend an organizational hierarchy to generate one or more chains of authority. A required attribute typically identifies the first approver in each chain, and the specific action

determines how many approvers are in each chain.

List-creation rules can use any of the following action types:

- Absolute job level
- Relative Job Level
- Dual chains of authority
- Final approver only
- Manager then final approver, relative job level
- Supervisory level
- Approver group chain of authority
- Line-Item Job-Level chains of authority
- HR position
- Position level

See Predefined Action Types, page 5-5 for descriptions of each action type.

This is an example of a list-creation rule:

Rule A
If TRANSACTION_AMOUNT < 1000 USD, then require approvals up to at least job level 2.

Rule A has one condition, on the attribute TRANSACTION_AMOUNT. The approval is of the absolute-job-level type. So if the condition is true for a given transaction, AME will extend the transaction's chain of authority from the transaction requestor's supervisor up to the first approver in the supervisory hierarchy who has a job level of at least 2.

If several list-creation rules of the same action type apply to a transaction, AME enforces the most stringent approval required by those rules. For example, if two absolute-job-level rules apply to a transaction, the first requiring approvals up to at least job level 2 and the second requiring approvals up to at least job level 3, AME will extend the chain of authority up to the first approver with a job level of at least 3. So, when you want to create an exception to a list-creation rule, and the exception should require extra approval authority of the same type as the list-creation rule, the exception should itself be another list-creation rule.

List-Creation Exceptions

List-creation exceptions also generate chains of authority. The difference between a list-creation rule and a list-creation exception is that an exception suppresses selected list-

creation rules.

Sometimes you want to create an exception to a list-creation rule that decreases the level of, or changes the type of, the approval authority that the list-creation rule would otherwise require. In this case, you need a list-creation exception (or simply an exception). An exception contains at least one ordinary condition and at least one exception condition (see Conditions, page 4-2 for an explanation of condition types), as well as an approval. An exception can use any of the approval types available for list-creation rules.

The circumstances under which an exception overrides a list-creation rule are somewhat subtle. The two rules do not have to have the same ordinary conditions. Instead, both rules' ordinary conditions have to be defined on the same attributes, and both rules' conditions must all be true (including the exception's exception conditions). In this case, AME ignores the list-creation rule, and the exception has suppressed the list-creation rule.

There are several reasons AME does not require that an exception have the same ordinary conditions as a list-creation rule that it suppresses. First, one exception may be designed to suppress several list-creation rules. In this case, the scope of the exception's ordinary conditions must be broad enough to encompass that of each list-creation rule it suppresses. Second, an exception may be designed to apply to a more narrow set of cases than a list-creation rule. Third, it is sometimes desirable to adjust the scope of either the exception or the list-creation rule(s) that it suppresses, without simultaneously adjusting the scope of the other(s).

This is an example of an exception that suppresses Rule A:

Rule B
If TRANSACTION_AMOUNT < 500 USD and Exception: COST_CENTER is in {0743}, then require approvals up to at least job level 1.

(In Rule B, the first condition is an ordinary condition, and the second condition is an exception condition.) Note that the ordinary condition is defined on the same attribute (TRANSACTION_AMOUNT) as the condition in Rule A, but the two conditions are different. Rule B carves out a exception to Rule A for transactions with totals under $500 U.S. for a certain cost center. In this narrow case, the exception requires less approval authority (of the absolute-job-level type) than what Rule A would otherwise require, which is why the rule must be an exception, rather than a list-creation rule.

> **Note:** The list creation and list creation exception rules are together referred as chain of authority rule type.

List-Modification Rules

Sometimes you want to make exceptions regarding the approval authority granted to specific approvers, rather than the approval authority required for specific kinds of transactions. To do this, you need a list-modification rule. AME applies list-modification rules to modify the default chain of authority generated by all applicable

list-creation and exception rules. A list-modification rule can have (but need not have) ordinary conditions. However, it must have exactly one list-modification condition (see Conditions, page 4-2 for an explanation of condition types).

There are two common uses for list-modification rules: reducing an approver's signing authority, and extending an approver's signing authority. In the former case, the list-creation rules might effectively grant a certain level of signing authority to managers having a given job level, and you might want not to grant that authority to a certain manager, in spite of their having the requisite job level. To do this, you tell AME to extend the chain of authority past the manager by using the nonfinal-authority approval type in conjunction with the job level action types only. In the latter case, just the opposite is true: the list-creation rules require approvals beyond a given job level, but you nevertheless want to grant a certain manager at that job level signing authority. To do this, you tell AME to truncate the chain of authority after the manager by using the final-authority approval type. In both cases, you can limit the scope of the list modification to a broad or narrow set of ordinary conditions.

Here are some examples of list-modification rules that illustrate the possibilities:

Rule C
If PURCHASE_TYPE is in {OFFICE FURNISHINGS, OFFICE SUPPLIES} and any approver is: Kathy Mawson, then grant the approver final authority. (In Rule C, the third condition is a list-modification condition.) Rule C grants Kathy Mawson final authority for a narrow scope of purchase types.

Rule D
If TRANSACTION_AMOUNT > 1000 USD and the final approver is: Kathy Mawson, then require approvals at least one level up.

(In Rule D, the second condition is a list-modification condition.) Rule D revokes Kathy Mawson's signing authority for a broad set of transaction amounts.

Substitutions

Sometimes you want to delegate one approver's authority to another approver. To do this, you need a substitution rule. AME applies substitution rules to the chain of authority after modifying it by applying any applicable list-modification rules. Like a list-modification rule, a substitution rule can have (but need not have) ordinary conditions. However, it must have exactly one list-modification condition. So a substitution rule differs from a list-modification rule only in its use of the substitution approval type.

This is a sample substitution rule:

Rule E
If TRANSACTION_AMOUNT < 500 USD and CATEGORY in {MISCELLANEOUS OFFICE EXENSES} and any approver is: John Doe, then substitute Jane Smith for the approver.

(In Rule E, the third condition is a list-modification condition.) Rule E delegates John Doe's authority to Jane Smith, for the class of transactions defined by the rule's ordinary

conditions.

Pre- and Post-List Approval-Group Rules

The four rule types described above generate a transaction's chain of authority. You may want to have one or more groups of functional specialists approve a transaction before or after the chain of authority does. In such cases, you need a *pre-* or *post-list approval-group rule*. An approval-group rule must have at least one ordinary condition, and must use either the pre- or post-chain of authority-approvals approvals type. For example:

Rule F
If TRANSACTION_AMOUNT < 1000 USD and CATEGORY_NAME in {Marketing Event}, then require pre-approval from Marketing Approvals Group.

Combination Rules

Combination rules combine actions from action types having different allowed rule types. There are two kinds of combination rule. One kind allows action types for list-creation rules, list-creation exceptions, production rules, pre-chain of authority rules, and post-chain of authority rules. The other kind allows action types for list-modification and substitution rules. Use a combination rule when several of your business rules apply to exactly the same cases. Doing so collects the business cases in a single rule for convenience. It also reduces runtime performance overhead.

Production Rules

Production rules output productions to the integrating application. Each integrating application uses productions in its own way. Consult your integrating application's documentation for information about what AME productions (if any) the application uses.

Rule Properties

A rule has the following properties:

Actions

Each rule must have one or more actions. Generally a rule's type determines which action types it can use. There are three exceptions:

1. Production rules can only use the production action type.
2. All of the approver-generating rule types can have production actions. The productions produced by these rules are approver-level productions.
3. Combination rules combine actions from different rule types.

Start and End Dates

A rule's start and end dates determine its lifespan, the period in which the rule can be used. When you select the current date as a rule's start date, AME sets the start date at the current time as well. When you select a future start or end date, AME sets that date to 12:00 a.m. (the beginning of the day).

A rule's start and end dates are not the same as the start and end dates of a rule use for the rule. The rule use's lifespan determines when the rule is in force for the rule use's transaction type, not the rule's lifespan. The rule's lifespan merely determine when the rule is available for use. Give a rule a future start date if you want it only to be available in a future period. Give a rule an end date of 31-DEC-4712 if you want it to be available indefinitely, after the rule's start date.

> **Note:** A rule cannot be created with a past date.

Description

A rule's description can be up to 100 bytes long. The description should be general enough to make sense in all transaction types that might use the rule.

Item Class

A rule used by a transaction type contributes to the approver lists of the items in the rule's item class. Note that header-level rules may have conditions on attributes belonging to at most one subordinate item class, and subordinate-item-class rules may have header-level conditions. Thus, the rule

Example
If TRANSACTION_TOTAL < 1000 USD and COST_CENTER in {481, 6012} then require approvals up to absolute job level 4.

could be either a header-level rule or a cost-center-level rule. In the former case, the rule would generate a chain of authority up to job-level four for the header item. In the latter case, if the header-level condition were satisfied, the rule would generate a chain of authority for the approver lists of cost centers 481 and 6012. Be careful to use rules assigned to the item classes you intend, when the rules have conditions on multiple item classes.

Rule Priorities

The purpose of rule priorities is to prioritize a transaction type's rules and, at run time, remove from the set of rules that would otherwise apply to a transaction, those rules of insufficient priority. A *rule priority* is a positive integer associated with a rule within a transaction type. (Each transaction type that uses a rule can assign the rule a different priority.)

> **Note:** The priority increases as the priority value (number) decreases:

two has priority over three, etc.

When rule priorities are enabled for a given rule type (within a given transaction type), the Rules tab and the rules details pages display rules' priorities; and one can edit a rule's priorities as one would edit any other rule property. When priorities are disabled, they are neither displayed nor modifiable.

The Administrator and Business Analyst dashboards have special pages for editing the rulePriorityModes configuration variable's value. The value contains a priority mode and threshold for each rule type. There are three possible priority modes: absolute, relative, and disabled. A threshold is a positive integer, and it only has meaning for the first two priority-mode values. See Configuration Variables, page 9-3 for details of each of these modes.

Let us look at the following rule priority processing example:

Requirement

It is required that transactions be routed to specific approver groups for a limited selection of cost centers otherwise route to a default group. For example if the stated requirement is:

- If transaction cost center is 1234 require post approval from approver group A
- If transaction cost center is ABCD require post approval from approver group B; otherwise require post approval from approver group Z

Solution

From the Configuration Variables, Rule Priority Modes, enable absolute priority mode for the rule type Post List Approver Group and set the rule priority threshold of 1. For each rule where there is a cost center approval group, assign priority of 1 to the rule. For your exception rule, assign priority 2 to the rule.

For example:

- Rule 1: Where cost center = 1234, require post approval from A, rule priority 1
- Rule 2: Where cost center = ABCD, require post approval from B, rule priority 1
- Rule 3: Require post approval from Z, rule priority 2

By setting the relative rule priority to 1, AME will determine how many rules of each priority value are active and discard any that fall outside of the configuration setting. Therefore, if any rule with priority 1 is active, then this rule would be used and the over-arching case Z would be discarded and the group A or B selected. If no rule for priority 1 is active, then the over-arching rule will not be discarded and so approver group Z will be notified.

Using Rules

The deployment of a rule depends upon its use. It is possible to have multiple use for a rule, each one spanning a specific data range which allows rules can be switched on/off as required.

A rule use can also span transaction types, effectively sharing rules between them. In both cases, any such use is marked on the UI to indicate either future dated rules or rules shared across transaction type use.

Once a rule is created, it is available for use in all transaction types. A rule use tells AME that a given transaction type should use a given rule for a given lifespan. A rule use involves the following:

- **Transaction Type**

  You always create a rule use within your current transaction type in the AME user interface. This is the transaction type to which the rule use is assigned.

- **Rule**

  There are two ways to create a rule use. First, you can create a use for a rule when you create the rule. Second, you can create a use for a rule that was created previously. In the second case, you can create a use for a rule already used by your current transaction type (for a lifespan that does not overlap with those of pre-existing uses for the same rule, in the current transaction type), or for a rule used by another transaction type.

- **Start and End Dates**

  A rule use's start and end dates make the rule active for the use's lifespan, in the use's transaction type. As for rules' start and end dates, when you select the current date as a rule use's start date, AME sets the start date at the current time as well. When you select a future start or end date, AME sets that date to 12:00 a.m. (the beginning of the day). When you create a new rule use, make sure its lifespan does not overlap the lifespan of another use for the same rule in your current transaction type.

- **Priority**

  When the current transaction type has enabled rule priorities for the type of the rule in a rule use, the use must define a priority for the rule. (See the description of the rulePiorityModes configuration variable in Configuration Variables, page 9-3 for information about enabling rule priorities.) The priority must be a counting number (positive integer).

  Each transaction type can enable or disable rule priorities for each rule type, using the rulePriorityModes configuration variable. When this variable enables priorities for a given transaction type and rule type, the configuration variable's value

specifies whether AME should interpret priority values as absolute or relative. The configuration variable also specifies a threshold value. How AME interprets the threshold at run time depends on whether the priority mode is absolute or relative.

When absolute priorities are enabled, all rules use with priorities that are numerically higher (larger) than the threshold are suppressed. For example, suppose

- The threshold for list-creation rules is one.
- The priority mode is absolute.
- Two list-creation rules' conditions are satisfied.
- The satisfied rules' use has priority one and two, respectively.

In this case, only the first rule (whose use has priority one) applies. The second rule (whose use has priority two) will be suppressed.

When relative priorities are enabled, if the threshold is t, AME determines the t numerically lowest (smallest) priorities among the rules of the relevant rule type that otherwise would apply. All rules of this type having a priority that is numerically higher (larger) than the numerically lowest priorities are suppressed. For example, suppose

- The threshold for list-creation rules is two.
- The priority mode is relative.
- Four list-creation rules' conditions are satisfied.
- The satisfied rules' use has priorities three, four, three, and seven, respectively.

In this case, the first three rules (whose all use have one of the two lowest priorities) apply. The fourth rule (whose use has priority seven, which is not one of the two lowest priorities) will be suppressed.

- **Approver Category**

  When for-your-information (FYI) notifications are enabled for a given transaction type, the transaction type's rule use for all approver-generating rule types must specify an approver category (FYI or approval). The approver category is assigned to each approver generated by the rule in the use. This lets a rule generate approval approvers in one transaction type, and FYI-notification recipients in another transaction type.

How AME Handles Multiple Requirements for an Approver

Sometimes an approval consisting of several rules of different types may require that a

given approver appear in different places in a transaction's approver list. AME assumes that an approver should only approve a transaction once, so it has to decide which rule (s) requirements prevail. There are two common cases of this problem. Here are brief descriptions of each case, and explanations of how AME handles them:

- A list-creation, exception, list-modification, or substitution rule includes the approver in the chain of authority; and a pre- or post-approval rule requires the approver's pre- or post-approval. In this case, AME only includes the approver in the chain of authority. The reason is that omitting the approver from the chain of authority might change the identity of the approvers that follow the approver, in the chain of authority; and AME assumes that preserving the chain of authority is more important than preserving the default approver order.

- Two approval-group rules include the approver in different (pre or post) approver groups. In this case, AME includes the approver in the first approver group in the approver list. AME here assumes that the approver should approve at the earliest opportunity.

Example of Setting Up an Exception Rule

Suppose you already have a rule requiring managerial approvals up to a manager with a job level of at least six for purchase requisitions totaling less than 5000 USD. You want to create an exception to this rule that requires approvals only up to a job level of at least four, when the requisition is for computer equipment. The rule you want would be of the exception type (not the list-creation type), because it decreases the level of approval authority required. So you would follow these steps to create the rule:

1. The pre-existing list-creation rule for the normal case would already have required a TRANSACTION_AMOUNT currency attribute, and a condition defined on that attribute. However, the Purchase Requisition transaction type might not include a suitable predefined attribute for a transaction's category. Supposing it does not, create a string attribute named (say) 'CATEGORY'. (If the attribute name already exists, share the attribute name. If not, create it with a sufficiently generic description for other transaction types to share the name, because the attribute name itself is sufficiently generic for several transaction types to want to use it.) Enter for the attribute a (dynamic) use that selects a Purchase Requisition's category from the appropriate tables or views.

2. Create an *exception* condition on the CATEGORY attribute having only one allowed value, (say) 'COMPUTER EQUIPMENT'.

   You can now create the rule itself:

3. Enter the description 'computer equipment up to 5000 USD'.

4. Select the list-creation exception rule type.

5. Let the start date default to today.
6. Leave the end date blank, so the rule is in force indefinitely.
7. Select the absolute-job-level approval type.
8. Select the approval, 'Require approvals up to at least job level 4.'
9. Select the ordinary-condition attribute TRANSACTION_AMOUNT.
10. Select the ordinary condition, '0 <= TRANSACTION_AMOUNT < 5000'.
11. Select the exception-condition attribute CATEGORY.
12. Select the exception condition, 'COMPUTER EQUIPMENT.'

Creating a Rule

You must always create a use for a rule in the process of creating the rule.

To create a rule:

Use the Create New Rule page.

1. Click the Rules tab to display the Rules page. If you are navigating from the Business Dashboard, then select the required transaction type in the Approval Process Setup available in the Business Dashboard and click the Rules link
2. Click Create to open the Create New Rule page.
3. Enter the rule details. The information you must enter varies with the rule type you select.

   Note: Before you create a production rule, you must ensure the following:

   - Allow productions for your transaction type by selecting an appropriate value for the productionFunctionality configuration variable. See: Configuration Variables, page 9-3
   - Include the production action type for your transaction type. See: Predefined Action Types, page 5-5
   - Add the production actions, if required. See: Creating an Action, page 5-19

   If you are creating a List Creation, Combination: List Creation, List Creation

Exception, Pre-List Approver Group, Post-List Approver Group rules, then provide the following information:

- Enter the rule name. Ideally, you can include the required approval action in the rule name.
- Select the type of rule to enable AME to generate the transaction's approver list or productions.
- Select the item class to enable the rule to generate approver lists of the items in the rule's item class.
- Select the approver category if you have enabled for-your-information notifications for a given transaction type. AME assigns the approver category to each approver that the rule generates in its use. This enables a rule to generate approval approvers in one transaction type, and FYI-notification recipients in another transaction type.

  Note: This feature will be available if the integrating application has implemented FYI Notifications. See: Configuration Variables, page 9-3

- If you have enabled rule priority, then enter a priority. AME uses the rule priority to prioritize a transaction type's rules while generating the approver lists or productions.
- Select the start and end date for the rule to determine its lifespan and indicate the period in which you can use the rule.

For List Modification and Substitution rule, you do not require item class and approver category details.

4. Click Next to add conditions.
5. Click Add Conditions to display the available conditions.
6. Select the required condition and click Continue to add the condition to the rule.

   Note: You can create new conditions using Create or update an existing condition using the Update icon. The changes to an existing condition are applicable wherever this condition is used.

7. Click Next to add actions.
8. Select the action type and action.

Note: For a production rule, ensure you select the production action type and production action.

9. Click Next to review the rule details.

10. Click Finish to use the rule in your transaction type and also make it available for all transaction types.

Creating a Rule Use

To use a rule, from another transaction type, in the current transaction type:

1. Click the Rules tab.

2. Click Use Existing Rule to open the Use Existing rule: Create Usage page.

3. Query the transaction type already using the required rule.

4. Select the start and end dates to determine the rule use's lifespan.

   Note: Ensure the new use's lifespan does not overlap the lifespan you noted in step two above.)

5. If the rule has FYI notifications enabled, then select the use's approver category.

6. If the rule has use priorities enabled, then enter the use's priority.

7. Click Finish to add the rule use to the current transaction type.

To reuse an existing rule in the current transaction type:

1. Click the Rules tab to display the current and future rules in the current transaction type.

2. Click the Duplicate icon for the required existing rule.

3. Select the lifespan, approver category, and priority for the new use.

4. Click Apply to add the new rule use to the current transaction type.

Editing a Rule and its Use

To edit a rule and its use:

Use the Update Rule page.

1. Click the Rules tab to display the current and future rules in the current transaction type.
2. Click the Update icon for the rule you want to edit.
3. Modify the rule properties of interest.

   Note: If you want to add a condition or an action, then click Add Condition or Add Action and enter the required details.

4. Modify the lifespan, approver category, and priority of the rule use and click Apply.

Deleting a Rule and its Use

To delete a rule, you must ensure that the rule is not used in any transaction type. When the rule is not used in any transaction type, AME automatically deletes the rule.

To delete a rule use:

Use the Rules page.

1. Click the Rules tab.
2. Click the Delete icon for the specific use of rule that you want to delete.
3. Confirm the deletion when prompted.

Reinstating an Inactive Rule

You can reinstate an inactive rule to use it in your transaction type.

To reinstate an inactive rule:

Use the Use Existing Rule page.

1. Click Use Existing Rule in the Rules page to open the Use Existing Rule page.
2. Query the required transaction type.

3. Query the inactive rules in the transaction type.

4. Select the inactive rule and click Continue.

5. Enter rule life span details and click Finish to reactivate the rule in the transaction type.

8

Testing

Overview of Testing

To ensure your transaction type applies appropriate rules to each of your business cases, you create test transactions on AME's Test Workbench, or 'real' transactions in the integrating application. You can select a real transaction and create a test case from it for future testing or create the initial test case. You can store the simulated and the real transaction test cases for future reference.

The Test Workbench displays a list of previously defined test cases. You can select an existing test case and have AME treat this as a real transaction to determine the active rules and the final approver list. By copying a test case, you can have many variations on a standard transaction. This enables you to exhaustively test your approval setup. When defining or updating a test case, you can supply values for the attributes of interest. For the test case, it is possible to define header values and if appropriate create detail lines in order to mimic an invoice for example. The Test Workbench enables you to check for item level, approver level, and transaction level productions. In addition, it displays the attribute values for the subordinate items in a transaction. For debugging and testing, you can see the various stages of the approver list.

Planning Your Test Cases

While planning you implementation strategy, your implementation document, showing your approval policies, should specify test cases sufficient to verify that your transaction type does the following things according to your business rules:

- Fetch correct item IDs.
- Fetch correct attribute values.
- Evaluate the rule use correctly.
- Process the rule use priorities correctly.

- Process production rules and actions correctly.
- Produce the correct default approver list.
- For real transactions, handle inserted approvers correctly.
- For real transactions, handle suppressed approvers correctly.
- Process repeated approvers correctly.
- Parallelize a transaction's approval process correctly.
- Process forwardings correctly.

You can do most of your testing using the Test Workbench. You may also use SQL*Plus and the integrating application to test certain aspects of your implementation.

Testing Your Approval Rules

Use the Test Workbench tab to perform either real-time or simulated tests for the given approval rules setup of a transaction type. Using the Test Workbench, you can do the following:

- View the name and description details of all the saved test cases for the given transaction type.
- View the attributes and attribute-values stored in the test cases.
- Update the existing test cases by clicking the Update icon. The Test Case Update page displays the attributes that you can update.
- View the final applicable rules and the final processed approver list for a selected test configuration by clicking Run in the Test Workbench page.
- View the different approval processing stages by clicking View Approval Process Stages in the Test Workbench page.
- Copy a selected test case by clicking Copy in the Test Workbench page. AME adds a prefix to the test name as a new test case. You can later update the new test case.
- Create a new test case by clicking Create in the Test Workbench page. You provide a name and description for the test case and also set the attribute values. You can add item class attributes and set their values from this page. You can either save the test case or invoke it as an adhoc test case by clicking Run Test Case.
- View real transactions by clicking Run Real Transaction Test in the Test workbench page. This opens the Real Transaction Test page in which you can query for a given transaction (using the transaction-ID) for the given transaction type. After AME

retrieves the values, you can test-run the approval process, by clicking Run Test Case. You can save the real transactions from this page.

- Test forwarding behaviors. AME currently does not have any test functionality that would let you update an approver's approval status such as the approver forward. To test how AME responds to forwardings, you must do the forwardings by responding the approval-required notifications generated by the integrating application, and then viewing the transaction's approver list on the Test tab using Run Real Transaction Test.

9

Administration

Administration

Overview

The Administrator Dashboard is available only to users with the Application Administrator responsibility. You can use this dashboard to maintain AME's configuration variables and transaction types. Additionally, you can view and analyze AME's runtime exceptions.

An AME application administrator has many tasks. As an application administrator, you can perform the following application and transaction-type administrative tasks:

- Set the configuration variables' default values (application-wide values)
- Set the transaction type's configuration variables' values
- Create a transaction type
- Update a transaction type
- Delete a transaction type
- View the exception log
- Clear the exception log
- Run the Setup report
- Add approver types

Transaction Type

In AME, a transaction type represents a set of transactions generated by a given application, for which AME maintains a single set of approval rules. A single application can have several transaction types. For example, the Web Expenses self-service application represents one of several possible transaction types for Accounts Payable.

In AME, transaction type administration concerns the following tasks that affect your current transaction type, without deleting it:

- Setting the transaction type's configuration variables' values.
- Viewing the transaction type's exception log.
- Clearing the transaction type's exception log.

- Viewing a specific transaction's exception log.
- Clearing a specific transaction's exception log.
- Editing and adding item classes to the transaction type.
- Updating the mandatory attributes for the transaction type.

Configuration Variables

This topic explains what each configuration variable means, and whether a configuration variable can have a transaction-type-specific value. Additionally, it indicates in each case whether a transaction type can override the default value.

Admin Approver: adminApprover

The adminApprover variable identifies the person or user account that AME identifies as a transaction's next required approver to the application requesting the approver's identity, when AME encounters an exception while generating the transaction's approver list. A transaction type can override this variable's default value.

A widget is provided to select the person or the user who is the adminApprover.

Allow All Approver Types: allowAllApproverTypes

The allowAllApproverTypes configuration variable is boolean value. When its value is 'yes' for a given transaction type, the transaction type can use rules that use all approver types known to AME. When the variable's value is 'no', the transaction type can only use rules that use the HR person (employee) and FND (Oracle Applications) user approver types.

When you select the allowAllApproverTypes variable at step 4 above, the 'Edit a Configuration Variable Default Value' form presents a 'yes' and 'no list . Select one of these values, and then select the 'Submit Changes' button to update the variable's default value.

> **Note:** You cannot edit this variable's value for a specific transaction type; you can only edit the default value.

Allow All Item Class Rules: allowAllItemClassRules

The allowAllItemClassRules configuration variable is boolean valued. When its value is 'yes' for a given transaction type, the transaction type can use rules pertaining to any item class. When the value is 'no', the transaction type can only use header-level rules.

When you select the allowAllItemClassRules variable at step 4 above, the 'Edit a

Configuration Variable Default Value' form presents a select list with the possible values 'yes' and 'no'. Select one of these values, and then select the 'Submit Changes' button to update the variable's default value.

> **Note:** You cannot edit this variable's value for a specific transaction type; you can only edit the default value.

Allow Fyi Notifications: allowFyiNotifications

The allowFyiNotifications variable is boolean valued. When its value is 'yes' for a given transaction type, the transaction type can have a rule use that generate FYI notifications. When its value is 'no', the transaction type can only have a rule use that generate requests for approval.

When you select the allowFyiNotifications variable at step 4 above, the 'Edit a Configuration Variable Default Value' form presents a select list with the possible values 'yes' and 'no'. Select one of these values, and then select the 'Submit Changes' button to update the variable's default value.

> **Note:** You cannot edit this variable's value for a specific transaction type; you can only edit the default value.

Currency Conversion Window: currencyConversionWindow

The currencyConversionWindow variable identifies how many days AME should look back , at most, to find the a currency conversion rate. The default value is set to 120 days. AME uses the GL Daily Rates table and associated routines to perform currency conversions.

When you select the currencyConversionWindow variable at step 4 above, the 'Edit a Configuration Variable Default Value' form presents a form that lets you enter the variable's value. Enter the value, and then select the 'Submit Changes' button to update the variable's default value.

Distributed Environment: distributedEnvironment

The distributedEnvironment variable indicates whether AME has been installed in a distributed-database environment. It has two possible values: 'yes' and 'no'. A transaction type cannot override this variable's default value.

AME has its own exception log, and in most cases it logs runtime transactions to that log. If the application that owns a given transaction type calls AME from within a workflow, AME also logs exceptions to Workflow (see Runtime Exceptions: page 116 for details); and it uses an autonomous transaction to commit exception data to its own log, to avoid interfering with Workflow's transaction management.

If however AME is installed in a distributed-database environment, and is called from within a workflow, it cannot initiate an autonomous transaction, because such transactions are not yet possible in a distributed environment. In this case it must query Workflow directly (where possible) for a record of AME exceptions. This log is less robust than AME's own log, and so AME avoids using it where possible.

In short, the distributedEnvironment variable is necessary to make sure AME logs exceptions internally whenever possible, while avoiding autonomous transactions where they would produce runtime errors.

Forwarding Behaviors: forwardingBehaviors

The forwardingBehaviors screen defines a set of constants, which determines how AME handles forwarding in a number of special cases.

The behavior for forwarding to someone not already in the list is always the same: the forwardee is inserted as an approver of the same type, immediately after the forwarder. When the forwarder and forwardee are chain of authority approvers, and the forwarder lacks final authority, AME extends the chain of authority starting from the forwarder until it finds someone with final authority. (This would normally be the typical case.).

AME predefines default values that are expected to be the normal use for each forward type. Oracle Application teams seeding transaction types can override these defaults to ones more suitable for the particular business process.

There are a number of different types of forwarding scenario that might occur during the approval process. For each of the listed scenario below, the AME engine can be instructed to amend the approver list in a pre-defined way. Not all possible outcomes are applicable for each forwarding scenario, these all indicated below.

- A chain of authority approver forwards to a previous approver in the same chain of authority.
- A chain of authority approver approves and forwards to a previous approver in the same chain of authority.
- A chain of authority approver forwards to a subordinate in the same approver hierarchy, but not in the same chain of authority.
- A chain of authority approver approves and forwards to a subordinate in the same approver hierarchy, but not in the same chain of authority.
- A chain of authority approver forwards to another approver already in the approver list, but not in the same hierarchy.
- A chain of authority approver approves and forwards to another approver already in the approver list, but not in the same hierarchy.
- An ad-hoc approver forwards to an approver already in the approver list.

- An ad-hoc approver approves and forwards to an approver already in the approver list.

Depending on the case, the allowed sub-values may include any of the following:

- **ignore**: Ignore the forwarding (treat it as an approval).
- **forwardee only**: Just add the forwardee to the approver list.
- **forwardee then forwarder**: Add the forwardee and the forwarder to the approver list.
- **skip forwarder**: Extend the chain of authority starting with the forwardee, but skip the forwarder in the extended chain.
- **repeat fowarder**: Extend the chain of authority starting with the forwardee, and include the forwarder in the extended chain.
- **remand**: Add to the approver list all approvers between the forwardee and the forwarder including the forwarder.

When you select the forwardingBehaviors variable at step 4 above, the 'Edit a Configuration Variable Default Value' form presents eight select lists, one for each special case. Select a sub-value on each select list, and then select the 'Submit Changes' button to update the variable's default sub-values.

Production Functionality: productionFunctionality

The productionFunctionality variable indicates which kinds of productions are allowed within a given transaction type. The possible values are as follows:

- All production rules
- Per-approver production rules
- No production rules
- Per-item production rules

When you select the productionFunctionality variable at step 4 above, the 'Edit a Configuration Variable Default Value' form presents a radio-button input with these options. Select the one you want, and then select the 'Submit Changes' button to save the value. (Note that when you cannot edit this variable's value for a specific transaction type; you can only edit the default value.)

Purge Frequency: purgeFrequency

The purgeFrequency variable indicates how many days AME should preserve

temporary data before purging it from AME's database tables. The value must be a positive integer.

When a transaction's temporary data is purged, its approval process starts over, as if no approver had approved the transaction. Therefore, the purgeFrequency variable's value should be high enough so that no transaction will require this many days to be approved by all of its approvers. At the same time, the purge frequency should be sufficiently low to avoid unnecessary growth in the size of AME's temporary-data tables.

A transaction type can override this variable's default value. When a transaction type overrides purgeFrequency, AME preserves that transaction type's temporary data only for the overriding value. This enables you to set the purge frequency differently for each transaction type, adjusting its value for each to a reasonable upper limit on the number of days required for a transaction of that type to complete its approval process.

Repeated Approvers: repeatedApprovers

Indicates how many times to require an approver's approval in the absence of forwarding. An approver may appear many times within the overall approval list. This can be due to a number of factors such as the approver exists as a pre/post approver as well as appearing within a chain of authority. In these circumstances it may be desirable to restrict so that the approver is only required to approve once in the following circumstances.

One of the following options should be selected:

- Once per transaction
- Once per item class
- Once per item
- Once per sublist
- Once per action type
- Once per group or chain
- Each occurrence

Thus, you can choose to make an approver occur at most once within any given subtree of the approver-list tree.

> **Note:** AME considers an approver as repeated only if the approver exists in the approver list with matching approval category. If the approver exists in the approver list twice with approval category as 'FYI' and 'Approval', then the approver will not be marked as repeated.

Rule Priority Modes: rulePriorityModes

The rulePriorityModes variable has a set of sub-values for each rule type. A rule type's rulePriorityModes values determine whether AME allows rule use priorities for the rule type, within a given transaction type. The values also determine how AME handles priorities, when they are enabled.

There are two sub-values for each rule type. The priority **mode** indicates whether priorities are allowed, and if so, how AME interprets them. The modes that enable priorities use a numerical **threshold**. If the mode is **disabled**, AME does not allow the transaction type to use rule use priorities for the rule type. If the mode is **absolute**, rules whose conditions are satisfied, and which have rule use priorities that do not exceed the threshold, apply.

The **relative** priority mode is very useful, but it takes more explaining. In general, suppose the threshold is t, and a given transaction satisfies k rules' conditions. Let P be the set containing the t numerically lowest distinct priorities among the priorities of the satisfied rules' use. Then all satisfied rules whose use are in P apply to the transaction.

For example, suppose the threshold is three, and a transaction satisfies the conditions of five rules. Suppose also that these rules' use have priorities one, two, two, three, and four. Then the top three priorities are one, two, and three, so all but the last rule apply to the transaction.

Record Approvals Deviations: recordDeviations

The recordDeviations configuration variable enables you to record deviations. You must set this configuration variable to YES at transaction type level to track the deviations.

Item Classes

Item classes have two uses. First, they let you build header-level rules that are sensitive to item-level business variables such as a line item's cost or the total amount billed to a given cost center. Second, they let you build subordinate-item-level rules that generate separate approver lists for each subordinate item in a transaction.

Item Classes

An item class aggregates a type of subordinate items that a transaction can have. An item class only has one property, its name, which can be up to 100 bytes long.

Items

There can be many items of a given item class, in a single transaction. For example, a transaction can have many line items. Each item must have a unique item ID, such as a line-item or cost-center ID and it must be at most 100 bytes long.

AME Objects Associated with Item Classes

An attribute is always associated with a single item class, and has a separate value for each item in the item class. An approver-generating rule is always associated with a single item class. It contributes to the approver lists for the items in the item class.

Using Item Classes

You use an item class in a transaction type by specifying values for its properties. The header item class is special because all transaction types have a predefined item class use for the header. You cannot delete this use. Each transaction has exactly one header item, and it always has the transaction ID as its item ID. The item class use for all other item classes are optional.

The item class use have the following properties:

Order Number

An item class use has an order number, which is always an integer with a value of 1 to 'n'. The item class order numbers in the related item class use sort the set of all item classes used by a given transaction type. The order numbers start at one and ascend to at most the number of item classes the transaction type uses. The order numbers are not necessarily unique, so the highest order number may be lower than the number of item class use in the transaction type.

AME uses the item classes' order numbers at run time to sort items' approver lists by item class, for a given transaction. See Parallel Approval Process, page 2-8 for details.

Parallelization Mode

An item class use has a parallelization mode that has one of two possible values: serial and parallel. The mode governs how AME sorts items' approver lists by item ID. See Parallel Approval Process, page 2-8 for details.

Sublist Mode

An item class use's sublist mode determines how AME sorts the sublists in each item's approver list. There are four possible values:

- Serial
- Parallel
- Pre-approvers first, then authority and post approvers
- Pre-approvers and authority approvers first, then post-approvers

See Parallel Approval Process, page 2-8 for more details.

Item-ID Query

An item class use tells AME how to generate a transaction's item-ID list for the items in a given item class, within a given transaction. The use accomplishes this by defining an item-ID query. The query may reference AME's :transactionId bind variable, and must

return the item IDs in ascending order.

> **Note:** If the item-ID query does not return the IDs in ascending order, then any dynamic attribute use for the attributes associated with the item class will not work properly.
>
> Oracle strongly recommends that SQL queries used to retrieve information must use only base tables and not secure views. The SQL query must also not include any context-sensitive logic.

What Causes Runtime Exceptions in AME?

The most common reason AME raises an exception (which typically results in the related application's stopping a transaction's workflow) is that AME cannot ascend a hierarchy, either because a slot in the hierarchy is vacant, or because an approver's level in the hierarchy is indeterminate. For example, in the case of the HRMS supervisory hierarchy, an approver may have a null supervisor or a null job level. In this case, the missing data must be entered into the appropriate application before restarting the offending transaction's workflow.

What happens when AME raises an exception?

When AME cannot determine a transaction's next required approver (in response to a request from an application, or when you use the Test Workbench tab), it:

1. Raises an exception in the routine that has trouble generating the approver list, and re-raises the exception up its call stack until an AME API routine catches the exception. Note that AME does not generally raise the exception to the routine that called its API.

2. Logs each exception to its internal exception log (where possible), and to Workflow (when the AME API was called from a workflow in another application).

3. If AME was responding to a request from another application it identifies the next required approver the person or user account identified by the appropriate value of the adminApprover configuration variable, and sets the approval status of this approver to ame_util.exceptionStatus. (This is the only circumstance where AME uses this approval-status value).

The requesting application may or may not notice that AME has identified an administrative approver as the next required approver, or that AME has set the approver's status to indicate that an exception has occurred. If it does, it typically will respond by stopping the transaction's workflow and notifying a Workflow system administrator. In this case, the person or user identified by the adminApprover configuration variable will not at this time receive a notification regarding this transaction (unless that person happens to be the Workflow system administrator as

well). The application may elect instead merely to send a notification to the administrative approver identified by AME, indicating that an exception has occurred for the transaction.

If the requesting application does not notice that the next required approver is an administrative approver, it will treat the administrative approver as it would any other approver: by sending the approver a notification requesting their approval of the transaction. The approver would then have to discern that AME had encountered an exception while attempting to calculate the transaction's approver list.

Oracle recommends that you configure the adminApprover configuration variable to identify the same individual as the Workflow and AME administrator for a given transaction type. This will have the effect of making sure the same individual is always notified when that transaction type's workflow errors, regardless of whether the error arises within AME.

Creating a Transaction Type

To create a transaction type:

Use the Create New Transaction Type page.

1. Click Create Transaction Type on the Administrator Dashboard to open the Create New Transaction Type page.
2. Select the application for which you want to create the transaction type.
3. Enter the transaction type key and name. Typically, these properties must identify the nature of the transaction.
4. Click Next to add item classes to the transaction type.
5. AME adds header item class as the default item class for a transaction type. If you want to add another item class, select from the Add List and click Go. Enter the item class use details and click Apply to add the item class to the transaction type.

   See Item Classes, page 9-8 and Using Item Classes, page 9-9 for further information.
6. Click Next to add mandatory attributes for the transaction type.
7. Select the attribute values and click Next to review the transaction type details.
8. Click Finish to add the transaction type to Oracle Approvals Management.

Updating a Transaction Type

As an administrator, you can update the transaction type properties, edit the transaction type's item-class use to set the values of parallelization parameters, add new

item classes, modify mandatory attributes' values. It is also possible, but uncommon, to edit the use's item-ID query.

To update a transaction type:

Use the Update Transaction Type page.

1. On the Administrator Dashboard, click Update for the transaction type you want to update.
2. Modify the transaction type details in the Update Transaction Type page.
3. Click Finish to submit your changes.

Deleting a Transaction Type

You can delete transaction types that are not predefined.

To delete a transaction type:

Use the Administrator Dashboard.

1. Click Delete for the transaction type you want to delete.
2. Confirm the deletion when prompted.

Setting Default Values for Configuration Variables

AME defines a number of configuration variables. In all cases, the configuration variables have default values that are created when AME is installed. In some cases, each transaction type can override the default value with its own value as well. Configuration-variable names and values are case-sensitive.

To set the configuration variables' default values:

Use the Configuration Variables page.

1. Click Configuration Variables in the Quick Links on the Administrator Dashboard or Business Dashboard to open the Configuration Variables page.
2. Change the required default values.
3. Click apply to set the configuration variables default values.

Setting the Transaction Type's Configuration Variables' Values

You use the same page to set a transaction type's configuration-variable values that you

use to set the variables' default values, with two exceptions. First, you cannot change the values of the allowAllApproverTypes, allowAllItemClassRules, or allowFyiNotifications configuration variables for predefined transaction types, if you have set the values as Yes.

To edit a transaction type's configuration-variable values:

Use the Configuration Variables page.

1. Click Configuration Variables in the Quick Links on the Administrator Dashboard or Business Dashboard to open the Configuration Variables page.
2. Query the transaction type.
3. Modify the variables' value(s). See Configuration Variables.
4. If you want to use any of the default configuration variables value in your transaction type, then leave the specific transaction type variable field blank.
5. Click Apply.

What's Next

You can update the following configuration variables at the transaction-type level:

- adminApprover
- allowAllApproverTypes
- allowAllItemClassRules
- allowFyiNotifications
- currencyConversionWindow
- forwardingBehaviors
- productionFunctionality
- purgeFrequency
- repeatedApprovers
- rulePriorityModes

Viewing a Specific Transaction's Exception Log

A transaction's exception log contains only those exceptions raised by AME when it

processes the transaction. It can be useful to view the log to determine exactly why the transaction's approval process has been interrupted.

To view the exception log:

Use the Transaction Exception page.

1. Click Exception Log in the Quick Links on the Administrator Dashboard to open the Transaction Exception Log page.

2. Query the transaction type to view the exception details for all the transactions in the specific transaction type. Otherwise, query the specific transaction by entering the transaction ID to view exception details of the specific transaction.

   The exception log appears, sorting the exceptions in descending log-ID order (that is, most recent exceptions first).

3. Click Purge to clear the exception log and confirm the operation when prompted.

Running the Setup Report

You can view and print the approval setup details for you transaction type by running the Setup Report.

To run the setup report:

1. Click Setup Report in the Quick Links on the Administrator Dashboard to open the Setup Report page.

2. Query the transaction type for which you want to run the Setup report and click Go.

3. Click Printable Page to enable you to print the report.

4. Click Print in the File menu to print the report.

How Should an Administrator Respond to an AME Exception?

However a Workflow system administrator or AME administrative approver becomes aware that AME is having trouble processing a transaction, they should respond to the problem as follows:

1. Check AME's exception log for the transaction. See Viewing Specific Transaction's Exception Log, page 9-13.

2. Check Workflow's context log for any other relevant details.

3. Correct the problem, which is usually data, that caused AME difficulty.

4. Restart the transaction's workflow.

   Clear the transaction's exception log in AME. See Viewing Specific Transaction's Exception Log, page 9-13.

Glossary

Oracle Approvals Management Engine (AME)

A highly extensible approvals rules engine that enables organizations implementing Oracle Applications to simply and effectively define business rules that determine who must approve a transaction originating within an application. You can devise simple or complex rules, as your organization requires, which then form part of your overall business flow. A central repository holds all the rules to facilitate management and sharing between business processes.

Action

An Action is the *Then* part of an Approval Rule that specifies how the application must progress a transaction's approval process in a particular way depending on the conditions met.

See: Approval Rule, page Glossary-1

Action Type

An Action Type is the generic type of specific actions. It enables you to specify the action to take if a transaction meets the condition of an approval rule. The action type, thus, generates the appropriate approvers for a transaction. As an AME administrator you can make particular action types available for specified transaction types. See: Transaction Types., page Glossary-2

Approver Groups

An Approver Group is a collection of approvers you define, which you can include as part of actions when you set up your approval rules.

Approval Rule

A business rule that determines a transaction's approval process. You construct rules using *conditions* and *actions*. For example, you can write a business rule with the conditions that if the total cost of a transaction is less than 1000 USD, and the transaction is for travel expenses, then the action must be to obtain approval from the immediate supervisor of the person triggering the transaction.

See also Conditions, page Glossary-2, Actions, page Glossary-1

Attribute

Attributes are the business facts of a transaction, such as the total amount of a transaction, percentage of a discount, an item's category, or a person's salary and so on. These business variables form part of the conditions of an approval rule, and determine how the transaction must progress for approvals.

Condition

A Condition is the *If* part of an Approval Rule that specifies the conditions a transaction must meet to trigger an approval action. A condition consists of an attribute, which is a business variable, and a set of attribute values that you can define. When a transaction meets the specified attribute values, then the application triggers the appropriate action.

See: Approval Rule, page Glossary-1

Deviation

A change to the standard approver list is a deviation.

Integrating Application

An application that uses Oracle Approvals Management Engine to manage the approval processes of its transactions.

See: Oracle Approvals Management Engine (AME), page Glossary-1

Transaction Type

An integrating application that uses AME may divide its transactions into several categories, where each category requires a distinct set of approval rules. Each set of rules is a transaction type. Different transaction types can use the same attribute name to represent values that the application fetches from different places. This enables several transaction types to share approval rules, thus facilitating a uniform approval policy across multiple transaction types.

Index

D

E

I

L

O

P

R

S

T