Oracle® Contracts Core

Concepts and Procedures

Release 11*i*

November 2000

Part No. A83649-04

ORACLE®

Oracle Contracts Core Concepts and Procedures, Release 11*i*

Part No. A83649-04

Contents

Implementing Oracle Contracts Core

Using Contracts Core

Send Us Your Comments

Oracle Contracts Core Concepts and Procedures, Release 11*i*

Part No. A83649-04

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us via the postal service.

Oracle Corporation
CRM Content Development Manager
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Welcome to the **Oracle Contracts Core, Release 11*i***, suite of applications.

This Concepts and Procedures provides information and instructions to help you work effectively with Oracle iStore.

This preface explains how Concepts and Procedures is organized and introduces other sources of information that can help you.

Intended Audience

This guide is aimed at the following users:

- Technical Service Representatives (TSR)
- Customer Service Representatives (CSR)
- System Administrators (SA), Database Administrators (DBA), and others with similar responsibility.

This guide assumes you have the following pre-requisites:

1. Understanding of the company business processes.
2. Knowledge of products and services as defined by your marketing policies.
3. Basic understanding of Oracle and Developer/2000.
4. Background in SQL, PL/SQL, SQL* Plus programming.

Structure

This manual contains the following chapters:

“Understanding Oracle Contracts Core" provides overviews of the application and its components, explanations of key concepts, features, and functions, as well as the application’s relationships to other Oracle or third-party applications.

“Implementing Oracle Contracts Core” provides general descriptions of the setup and configuration tasks required to implement the application successfully.

“Using Oracle Contracts Core” provides process-oriented, task-based procedures for using the application to perform essential business tasks.

Related Documents

For more information, see the following manual:

- *Oracle Contracts Core R11i Oracle Technical Reference Manual*
- *Oracle Contracts for Service R11i Concepts and Procedures*

Understanding Oracle Contracts Core

This topic group provides overviews of the application and its components, explanations of key concepts, features, and functions, as well as the application's relationships to other Oracle or third-party applications.

Overview of Oracle Contracts Core

Oracle Contracts Core is used for any business that requires "buy or sell" contracts that do not require industry specific processes. It is a centralized repository for contract information and serves as the foundation for all contracts modules.

You can record, group, organize, and query contract data. Set up an automated approval process for your contracts.

Contracts are composed of articles and rules. Articles contain the terms, conditions, and text of the contract. Rules are contract terms and conditions with the addition of computer-readable formatting that may cause Contracts Core to take action based on specific information. Rules are created to control different processes that manage contracts.

Oracle Contracts Core includes:

- Launchpad: the main entry point to contracts.
- Contract Index: a quick overview of all components of a contract.

Oracle Contracts Core integrates with Oracle Order Capture, Oracle Order Entry, Oracle Accounts Receivable, Oracle Accounts Payable, Oracle Inventory, Task Manager, Resources. It also integrates with the Oracle Service Core applications, which include Installed Base.

See also

Contract Articles

Understanding the Contract Launchpad and Navigator

The Oracle Contracts window is separated into two tabs:

- Contract Launchpad
- Contract Navigator

Contract Launchpad

The Launchpad tab is divided into three regions: Inbox, Recent Documents, and Bookmarked Documents.

The Inbox displays notifications for contracts that require action.

- **Find** searches for a contract by status or notification type.
- **Open** opens a contract selected in the Inbox.
- **Respond** invokes workflow functions such as approving, signing, or rejecting a contract.

The Recent Documents region displays and gives you access to the most recently modified documents.

Use Bookmarked Documents to maintain your own personal list of contracts.

Contract Navigator

Use the Contract Navigator tab to view contracts by contract groups.

This tab is divided into two sides. The left side displays the contract groups and the right side displays the contracts in a grid that belong to the group you select from the left.

From the grid you can right-click to invoke several contract administration functions, or you can double-click the contract to get a quick overview of the contract. In addition to the overview, double-clicking gives you access to the contract history, schedule, and communications.

Contract Categories

You can use the authoring window to create and maintain many categories of contracts. During setup, categories are created and related to the class of a contract. the class identifies whether the contract is core or service. You can tailor the authoring window to suit each category using:

- Line styles
- Party roles and contact sources
- Rule groups
- Status and operations

Contract Groups

Contract groups provide a convenient way to organize contracts using any criteria you like. They are similar to folders.

Create as many contract groups as needed to help you and other users find related information about contracts and contract parties. You can use any number of groups to classify a single contract and you can nest groups within other groups.

The groups you create can be public or private. Use public groups to group your contracts in a structured way that follows your company or department organization structure or business procedures. Use private groups to allow users to group contracts by their personal preferences.

If the system administrator gives you the privilege, then you can also create public groups that are accessible to all users. Public groups can contain individual contracts as well as other public or private groups.

Groups do not permit or restrict access to contracts.

Contract Group Maintenance

Once you create a group you need to maintain its membership.

You can manage and oversee the groups that were created by the group definition. Use the Contracts Groupings window to maintain group hierarchies or to assign contracts to groups.

In each of the tabs, Contracts and Groups, you can choose between two views. One view gives you access to all unassigned contracts or groups. The other expands the candidates to all groups or contracts.

Use the left arrow in the middle to select a group or contract from the right and add them to the group on the left. To remove a contract or group, select that contract or group on the left side and press the right arrow.

Contract Articles

A contract article is the text that describes and details the terms and conditions that are attached to a contract. Also see Understanding Contract Rules.

There are two different types of articles: standard and nonstandard.

- Standard articles are referenced from a library of articles. The text of standard articles cannot be changed. You still may add variation to an article in a contract.

  You can assign releases to articles. When you insert an article into a contract, the application chooses the current release.

  If you assign incompatibilities to a standard article, then the Quality Assurance check detects any incompatible articles in your contract.

- Nonstandard articles are specific to the individual contract. They are either written by the author of the contract or copied from the library of articles and modified.

Library of Articles

The library of articles contains standard articles that can be referenced or copied into contracts. The standard articles are organized into article sets.

Use article sets to organize articles logically and make them easier to include in contracts. You can then drill down to articles from the sets. You can assign price types and categories to article sets. When you author a contract, the list of values for the standard articles are filtered by the assigned price types and categories.

You can translate article sets into rules. When an article set is referenced or copied into a contract, the associated rules are also copied into the contract.

Create new article sets using the lookup code OKC_ARTICLE_SET.

Understanding Contract Rules

This section defines contract rules and explains how they are used in Oracle Contracts Core. Because rules are what trigger many functional processes within contracts, it is essential that they be clearly understood. This section covers the following subjects:

- What is a Rule?

What is a Rule?

Rules are the terms and conditions of a contract. Contracts can trigger other application processes by using the computer-readable, formatted rules. Create rules to control different processes that manage contracts. Use billing rules, for example, to determine when and how a bill should be sent. Oracle Contracts Core contains a preset list of rules.

Difference Between Rules and Articles

Rules are contract terms and conditions with the addition of computer-readable formatting that cause Oracle Contracts Core to take action based on the information. Articles are the textual presentation of rules.

When you put an article into Oracle Contracts Core, that article represents text. With a rule, however, you can initiate functions within the application. For example, an article states that a confirmation letter must be sent within three days of the signature date. When this information is entered as a rule, a notification for a confirmation letter is generated.

Using Conditions and Rules

You can use conditions and rules together or you can use each component individually.

A condition is part of the events concept:

- A condition is the combination of actions, condition lines and outcomes.
- When combined with a rule, a condition does not require an outcome.
- Condition lines are always optional.

With conditions you can capture contract events, like the signing or expiration of a contract. You can use condition lines to filter, if an outcome is executed. For example, you can define

that a quote is automatically created 5 days prior to the expiration of a contract, if the contract total does not exceed $50,000.00.

With the notification rule, you can schedule a generic notification, that does not use a condition like the expiration or the initiation of a change request. For example, You can define that the project leader is notified every other week to schedule a status meeting for the following Friday.

Using rules and conditions together, you can base the notification on conditions. For example, You can send a notification to the project manager, each time a change request is initiated. The notification is not sent, if it was initiated by the project manager. In this example, you would integrate a contract specific condition in the notification rule. The condition would identify the change request initiation and filter, if the requester of the change request is the project manager herself.

Note: If the condition lines do not offer you the attributes you need for filtering, then you can use the process definition to register a PL/SQL function to return true or false. For more information on registering a function, see Defining a Process.

Tracking Deliverables with Rules

Rules capture additional information you want to store for a contract. You can group rules into rule groups that help to enforce business processes and company standards.

A schedule rule captures date information to help you manage deliverables. Schedule rules store point-in-time specifications that can be used with events. An example schedule rule is “drug-free certification must be provided within 30 days of contract signing.” In other words, rules can capture information about dates or future dates that are not known at the time a contract is authored. Schedule rules can represent date driven, event, or periodic phrases such as these:

- Date driven: First Monday of every month
- Event: After the contract is signed
- Periodic: Every 6 months, every week

Rule Groups

A rule group contains one or many rules. Rules collect one or many related pieces of information. Rules are stored in a formatted way as opposed to free text like articles.

Your company can choose to assemble rule groups and make some rules optional. Once you have selected a rule group, you can see if the rules contained are optional.

Assign rule groups to your contracts according to your business procedures.

You can save incomplete data, but all required rules have to be complete to pass Quality Assurance.

Rules with similar characteristics are grouped so that you can easily reference and access them. For example, create a rule group for billing which would group together rule types such as Bill-To, Schedule Billing, and Payment Terms.

Object and Subject

When you define categories or you create a contract, you may encounter situations, where the role of a contract party may be obvious. However, the system may not be able to identify which of the parties or party roles should be displayed in a list of values.

For example where you have:

- Multiple parties for a specific role (more than 1 customer in a syndication agreement)
- Multiple roles, that only have slight differences (customer and customer agent)

The subject is the acting party, the object is the party being acted upon. For example, the billing rule would need to know who (the subject) bills whom (the object).

The concept of object/subject is considered in two situations:

- Defining a category
- When authoring a contract and adding/changing rules

Not all of the rules need the concept. When you enter a rule, where none of the parties is ever displayed in a list of values, the subject/object concept is irrelevant such as the currency rule group, notify rule group, renewal rule group, and so on. Therefore you cannot enter a object or subject in the fields for such rules.

When you assign rules to a category, you may have to add the roles to the setup, when the rule group requires it. If that setup step is missing, then the authoring form will notify you of the incomplete setup of the category, when you try to use the rule group in question.

For the example given above, when you add the billing rule to a category, you would have to identify that the customer role and not the customer agent role should be subject of the billing rule group.

When you author a contract, you may have more than one customer, for example when you record the contract for a syndicate. In that case, the party roles field allows you to identify, to which of the customers the invoice is sent. Once you have identified the right customer, then you can select the from correct billing addresses.

Note: You need to identify the object and subject, even if there is only one eligible candidate.

Where Rules Can Be Used

Rules can be put into use in three different places:

- Contract header
- Contract line
- Article sets

Quality Assurance and Rules

Each rule can consist of one or more fields. One or more fields can be required entries. You can only add rule groups and not single rules to a contract. Quality Assurance enforces required fields.

The optional check box in the rule groups impacts the required fields of a rule in the following ways:

- If you make a rule in a rule group mandatory, then the required fields of a rule must be entered.
- If you make a rule in a rule group mandatory, but there are no required fields, then Quality Assurance passes.
- If you make a rule in a rule group optional, with all fields nonrequired, then Quality Assurance passes.
- If you make a rule in a rule group optional, then the required items are only enforced if at least one field of the rule was entered. If you don’t enter anything for a rule, the mandatory fields are not enforced. You will experience this only in very exceptional cases.

In addition to rules with nonrequired fields, Quality Assurance can enforce dependencies or alternatives. For example, in a prepayment rule, you either have to enter an amount or you have to enter a percentage, but neither of the fields are mandatory. In this case, the Quality Assurance check will result in an explanation of why the rule did not pass.

Understanding the Contract Index

You can find and examine various components of a contract by using the Contract Index, a tree-style structured navigator. By using the contract index, you can follow, access, and understand the relationship of contract parts and better understand the context for the contract as a whole.

You can select any part of the contract index tree structure and navigate to any appropriate tab, row, or column that has the information that you are looking for.

The Contract Index has five display options, or views. With these views of the various contract components, you can see some of the hierarchical relationships and certain characteristics of the contract such as parties, line items, articles, and rules. The default displays the contract view.

For example, you could choose to view the contract rules as the root node. Oracle Contracts Core would then show all of the rules for that particular contract. You could then choose to see the association that the rule has to the parties, line items, terms, or events of the contract.

If, for example, you wanted to find terms and conditions that were at the line item level and that dealt with X, then you could find line item rules, making it much easier and quicker to locate the information needed.

Subcontracts

In order to fulfill the obligations of an existing contract, a party may need to establish a new contract with a third party to provide some or all of the items, such as materials or deliverables, required in the original contract. This is called creating a subcontract. These subcontracts need to maintain relationships between the components, such as terms and lines, of the parent agreement and those of the new contract. You can create a subcontract and further specify those articles or rules from the prime contract or master agreement that are to flow down to, and be incorporated into a subcontract.

Also, you can create task orders from a master agreement, with the terms and conditions of each task order linked from the master agreement. Alternatively, you can simply reference other contracts (and by inference, the terms and conditions of those other contracts) without specifying any particular details.

Using subcontracts, you can:

- Subcontract the components into an existing contract or a new contract.
- Maintain data consistency for the subcontracted contract. For example, a subcontract would not contain your customer as a customer in the subcontract.
- Select multiple lines or sublines for subcontracting.
- Inherit terms and conditions from higher levels in the hierarchy for the subcontracted lines or sublines.
- Reference terms and conditions.
- Modify the copied components.

You can select the lines or sublines and also the components such as parties, articles, and rules at the contract header level for subcontracting. After selecting the components, click the right arrow button to move them to the right navigator window.

When you create the subcontract, all selected components are copied into the subcontract. You can modify the components. You can also select parties to copy to the subcontract, but normally this is not useful because the roles change for a subcontract.

Understanding Status and Operations

An operation is an action you can take with a contract, such as update on line, update via a change request, or delete.

The status of a contract is a label defining where the contract stands in its life cycle.

Contracts Core provides the following status types:

- Entered: Contract is currently being edited and it can be completed but not approved
- Signed: Contract is approved, but not yet effective. This status is used, when the contract is not yet due, but should have the same protection from changes an approved contract has.
- Active: Contract is approved, signed, and effective
- Hold: Contract was set on hold from signed, active, or another hold type. For example, use hold when a customer is moving or the contract is in dispute.
- Expired: Contract was active, but is not effective anymore
- Terminated: Contract was active, but was terminated from either side before the contract expired
- Canceled: Contract never became active and is not planned to become active

You can define as many statuses per status type as you need.

You can have a different status on the contract header and on the contract lines. For example, you can terminate a single contract line, while the rest of the contract is still active.

The status of a contract changes a number of ways:

- Manually
- Through the approval process
- Through the extension process
- Through the termination process

- Automatically by the Status Change concurrent program

From the Launchpad you can:

- Initiate a manual status change.
- Extend a contract or contract line.
- Terminate a contract or contract line

See Working with the Contract Launchpad and Navigator.

From the contract authoring window you can submit a contract for approval.

Some status changes have to be done through a concurrent program. For example, the program Status Change automatically changes a Signed contract to Active when the start date is reached.

User Status

To provide greater control of contract status, you can add custom statuses for any status type. For example, to distinguish between entered contracts that are draft and those pending approval, you can define draft and pending statuses for the entered status type. For each status type, you define the default status that is automatically assigned when a status type automatically changes.

Operations

You can control the operations allowed per contract status and category. For each user status, you can define the operations that are supported. For example, you can specify that draft documents can be updated on line, but contracts pending approval must go through a change request process. You can also state that once a contract becomes active it cannot be updated at all.

Allowed operations are defined by status, and not by status type. If you define a new status for the status type Active, then you must also specifically allow operations for the new status, such as on-line update. If you create a new status without specifying any allowed operations, then that status allows read only access.

To set up new statuses and define the allowable operations, see Setting up Status and Operations.

Allowed Status Changes

In the following table, *auto* means that the concurrent program Status Change automatically changes the status when the conditions for a status change are met.

Status to From	Entered	Signed	Active	Hold	Expired	Terminated	Canceled
Entered	Yes	Yes/ through approval	Yes/ through approval	No	No	No	Yes
Signed	No	Yes	Yes/auto	Yes	Yes/ auto	Yes	No
Active	No	No	Yes	Yes	Yes/ auto	Yes	No
Hold	No	Yes	Yes	Yes	Yes	No	No
Expired	No	No	Yes	No	Yes	No	No
Terminated	No	No	No	No	No	Yes	No
Canceled	No	No	No	No	No	No	Yes

Access to Contracts and Security

You can fine tune access to contracts, such as granting a specific user rights to see a contract, but not to modify it. The category definition includes granting access levels to responsibilities. For each category you can define whether a given responsibility has read only or modify access.

This section covers the following subjects:

- Contract Groups
- Status and Operations
- Access
- Multi-Organization Considerations
- Restricted Operations
- Access troubleshooting Checklist

Contract Groups

Your first level of access to contracts is via the contract groups listed in the Contract Navigation window. Although you have access to a contract group, you do not necessarily see all the contracts in the contract group because other levels of security may be in place.

You can assign one or more contract groups to each contract. There are two kinds of contract groups: public and private. A private group is created by a user and only the user creating the group has visibility and access to that group. A public group is visible to all users.

Public groups can only be created by users that have the system profile option: OKC Public Group Creator enabled. If this system profile option is not set or is disabled, then the user can only create private contract groups.

Everybody with access to a contract group who is assigned to the contract group has full read and write access to the contract, as defined in the Status and Operations window.

Use contract groups to navigate to contracts through different navigation paths. A contract is referenced in contract groups, not copied. Assign contract groups in the authoring window in the Details subtab of the Summary tab. As a contract administrator, you can use the Contracts Groupings window to assign and unassign contracts to groups.

Status and Operations

The status and operations window defines by category and status which operations can be executed against a contract. The allowed operations are defined by status and not by status type.

In other words, if you define a new status for the status type Active, then you have to make sure that you specifically allow operations such as on-line update. If you create a new status without specifying any allowed operations, you implicitly allow no operations for this contract status.

Access

You grant access to contracts using the Responsibilities tab in the Category Definitions window. The category definition includes access levels, such as read only or modify, by responsibility.

Within a contract, you can grant access to one or many contract groups.

There are a couple of operations and statuses that prevent you from changing a contract. For example, while a contract is being changed by somebody else, you cannot access it. While a change request is pending approval, you can submit another change request, but your change request has to wait for the outcome of the pending change request.

You are restricted in the status changes you can perform on a contract. For example, you cannot terminate a contract that was never active. You have to cancel it. You cannot cancel a contract that was active.

For a complete list of allowed status changes, see Understanding Status and Operations.

Multi-Organization Considerations

When you log on to Oracle Applications, you automatically sign on within your organization. By default, your actions are restricted to read and modify contracts for your organization only. For some business uses, where contracts need to be accessed 24 hours a day from any organization around the world, contracts need to be accessible outside your organization.

If you need to give access to contracts across organizations, you can use the Administration subtab in the Summary tab of the authoring window. You grant read only or modify access to resources independent of the organization the user is in.

To create a resource, you must choose the CRM Resource Manager responsibility. A resource in this context refers to the resource definition of CRM, for example, customers, suppliers, employees, and so on.

If you want to use persons or roles already declared somewhere else in the application setup, for example an employee, then you must import these sources using the Import Resources function in the CRM Resource Manager responsibility. Make sure you assign contract execution rights in the Define Resource menu.

Note: Make sure that you have defined the role completely before importing the resource, for example: Define the employee and tie that employee to FND_USER, before you import the resource. You can import only once.

Restricted Operations

Although you have modify access to a contract via your responsibility or access group, you may still be restricted from performing certain operations against the contract. You can only perform operations that are specified for the status.

If your organization uses changes requests to update a contract and another user is in the process of applying a change request, then you will not be allowed to update the contract. You can submit another change request while another change request is being applied, but you must wait until the prior change request is completed before you can apply your change request to the contract.

There are also restrictions for changing the status of a contract. For example, you cannot terminate a contract that was never signed, instead it has to be cancelled. You cannot cancel a contract that is active, it can only be terminated. See the table of allowed status changes in Understanding Status and Operations.

Access Troubleshooting Checklist

1. User does not have access to contract group.

Check if contract group is public or private.

2. User cannot see the contract in the contract group.

   Check if the user is in the same organization and has access granted. Check if the user is using a responsibility with access to the category.

3. User can see the contract, but cannot update the contract.

   - User was not given access.
   - User is using a responsibility with read only privilege instead of modify.
   - Operations window excludes that operation or this operation is not included, thus implicitly disallowing that operation.
   - The contract may be in a status change that does not allow any operations.

Line Styles

A contract is composed of lines of text. Control the type of information that can be entered on a particular line by defining a line style. The line style sets input requirements and sets up the lists of values to choose from in a contract line during contract authoring.

This section covers the following information:

- Top Line
- Sublines and Items
- Pricing
- Line Types and Item Source
- Priced Lines and the Pricing Rule
- New Item Entry

Top Line

Each contract has one or more contract lines. A contract line can be a simple priced item line or can consist of several layers of lines and contract items. The most general and least detailed line in a contract is called the top line. You can break a top line into sublines and multiple contract items to provide a rich level of detail.

To include a top line and its details in a contract, you must include it in the category definition. For service contracts, you can define a top line style to be eligible for entitlement and invoicing. You define new top lines using the lookup code OKC: Line Type.

Sublines and Items

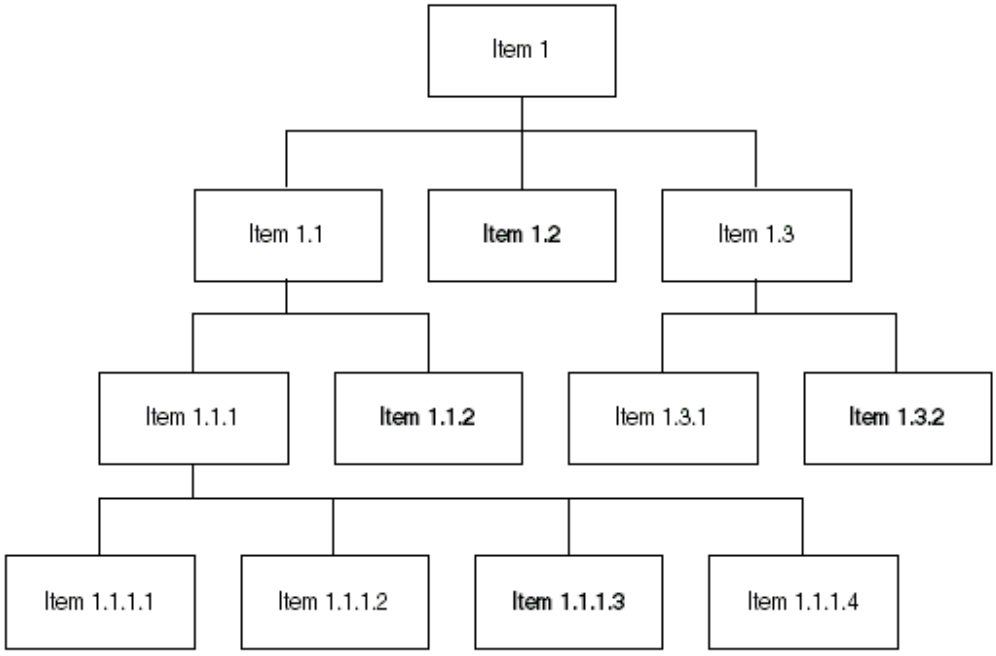
Use lines and sublines to group and categorize goods and services. The specific goods and services are referenced from your inventory or product catalog as contract items. You can use sublines to provide additional detail about the delivery of the contract item. For example, a line for a service product includes a subline that defines the level of service contracted for that product. A line that represents a licensed product includes a subline that identifies where the product is deployed.

Pricing

When you create a line style, you can define it as being a priced line style. You can enter a quantity, unit of measure, and a price for a priced contract line. In the authoring window, the line then calculates the price. You can overwrite the price in the negotiated price column.

For each hierarchical path in a line style, you can only price once. You can have multiple prices on the same level in a hierarchy.

For example, assume a line style hierarchy such as the following:



1. You can mark Items 1.1.13, Item 1.1.2, Item 1.2 and item 1.3.2 as priced items. None of these priced items is a parent or child of another priced item.
2. When you mark the line style for Item 1 at the top line as a priced item, all other dependent lines have their price tag removed.
3. Both Item 1.3.2 and Item 1.3.1 can be priced in this example.

Line Types and Item Source

For each level in the line style hierarchy, you select a line type and the item source. The item source defines the list of values available when authoring a contract that contains the line style.

The line styles that can be selected are based on the contract category and line style hierarchy.

One special line type is called "Free Format" and is used to enter any text description. It does not include an item source. The item source tells you, where you can add values to the selection. For example, the source Consulting refers to items in your inventory items with the item type Consulting. Similarly, you can add items for the source Media Pack, Self SerTraining or Training Line. The source System Item refers to all items in your inventory.

Priced Lines and the Pricing Rule

Before you enter priced lines in your contract, define the pricing rule group for the contract header or contract line. The pricing rule group identifies the price list to be used for your priced items. After you identify the price list and when you add priced lines to your contract; the list price of the item is retrieved. Based on the quantity you enter on the line, the extended price is calculated. This extended price can be overridden in the negotiated price column to account for contractual agreements.

New Item Entry

The inventory items list of values is restricted by the organization that is available to the contract. When a contract is entered, the organization ID is obtained from the multi-org profile option value. The inventory organization value is obtained from the Oracle Order Entry system parameters.

You cannot enter a new item into the individual inventory organizations directly. To enter a new item in Oracle Inventory, you enter the item into the master organization. Then you copy the item to the individual inventory organizations.

Understanding Roles and Contact Sources

A contract is an agreement between at least two parties. Each party assumes a role in relation to the contract, such as the vendor or the customer. The setup of roles differentiates between sell and buy contracts. You can add two or more parties when authoring a contract and you can define the party roles and define the sources from within Oracle Applications for validating parties.

Example

If you sell goods or services, then you are the vendor and your customer may already exist in your customer database (source) as a regular customer.

If you buy goods and services, then you are the customer and your vendor may already exist in your vendor database (source). Even though you are the customer in this contract, you don't want to register as a customer in your customer database just to be available as the customer in the contract.

In both contracts you have a customer and a vendor each. In one contract you are the customer, in the other you are the vendor.

New Roles

In a contract, a role can only be selected if the following conditions are met:

- The category you choose permits the use of that role.
- The role has a valid source. Make sure that the source has a valid date range, the intent (sell or buy) matches your contract, and that the source has values.

Once you have defined a new party role or contact role using Define Categories, you can select them in the Sources window, where you can assign the sources.

A new party role or contact role will only appear in your contract, if the following conditions are all met:

- The role or contact role is defined as a lookup code with a current effective date.
- The role or contact role has a current effective date in the sources window.
- The role or contact role has a source assigned with a current effective date and the right intent (buy or sell).
- The source for the role and intent contains at least one value.
- The party role may be needed (for Rules) as well.

Events

Use events to automatically initiate actions such as creating a service request, sending a notification, and creating a task. Events occur as a result of something happening or before an upcoming event.

In other words, an event acts upon phrases such as:

- When this happens, do something
- At a certain point in time, something happens

The "when something happens" or "at a certain point in time" are *actions,* and "do something" is an *outcome*.

The action can be qualified by conditions. For example, it may not be whenever something happens, but whenever something happens and some conditions are met.

In summary, events are composed of actions, conditions, and outcomes.

Events allows synchronous and asynchronous processing. The asynchronous processing takes advantage of the advanced queuing method. That is, you do not have to wait until an event is evaluated and all the outcomes are executed to continue with your activities. On the other hand, due to the type of business, some applications such as Relationship Plan require an immediate event evaluation while you are using the application.

This section covers the following subjects:

- Types of Events
- Conditions
- Types of Condition Lines
- Types of Conditions
- Outcomes
- Schedule Rules
- Renewal Rules

Types of Events

You can handle action-based events and date-based events.

Action-based events are predefined actions that occur in the application. Examples of action-based events are contract signed, contract terminated, counter updated, and change request initiated.

Date-based events are dates of significance within the application. For example, the contract expiration date.

You define action-based events and date-based events using the Actions window. In this window you also define action attributes that can be used further in conditions and outcomes.

Conditions

You use conditions to define additional criteria for an action-based event or for a date-based event. The conditions also specify the outcomes that need to be initiated when the condition evaluates to true.

Condition lines qualify action and date-based events. They are used to specify that an outcome is to be initiated not only when the action happens, but also when the action happens and certain conditions are satisfied. For example: “when the contract is signed and the amount of the contract is greater than $100,000,” or “30 days before contract expiration and when the contract category is network license agreement.” It is not required to define condition lines for a condition.

Types of Condition Lines

There are three types of condition lines: expressions, counters, and functions.

Using expressions, you can define simple comparisons using the operators such as <, >, =. Expressions are always based on action attributes.

Counter condition lines are applicable for Oracle Contracts for Service and Installed Base. You define conditions for a particular counter reaching a value. Counters can be defined for a product or for a service. For example “when the number of photocopies for a photocopier is greater than 1,000,” or “when the number of calls made by a customer in a gold service agreement is over 10."

Functions can be used for more sophisticated criteria that can be addressed using PL*SQL procedures. A function accepts one or more parameters and returns true or false after its evaluation. Functions don’t need to be based on action attributes.

Types of Conditions

There are two types of conditions:

- Independent conditions
- Conditions attached to contracts

Using condition templates, you can define multiple samples of conditions. From a template you can define either an independent condition or a condition to be attached to a contract.

Independent conditions are not owned by any object. Whenever an action occurs, regardless of the contract, the outcome is initiated if the condition evaluates to true. For example, “whenever any contract is signed create a task to notify the contract administrator.”

Conditions can be specific to contracts. Depending on the action, a condition can be related to a contract line or to a contract as a whole. For example “when the contract is signed” relates to the entire contract, and “when the contract line is terminated” relates to a line of the contract.

Outcomes

An outcome is an automated process that performs a task such as initiating a service request workflow or sending a notification. When the condition is true, the outcomes associated with that condition are performed.

Before an outcome can be associated with a condition, it must be defined using the Process Definition window. There are four types of outcomes: workflow, PL/SQL procedure, alert, and script. Workflow and PL/SQL procedure are outcomes. Alerts are synchronous outcomes.

You cannot use script for outcomes. Script is not referring to a SQL script. Script is only valid for the type of "outcome" to support Relationship Plan functionality as a synchronous event.

Schedule Rules

Schedule rules indicate that some deliverable is due at a future date or time that is not known at the time of authoring the contract. For example, "product" documentation must be provided within 30 days of contract "signing." Rule groups that contain schedule rules also contain a notification rule so that you can create tasks for the schedule rules. You can view schedule rules in the task manager.

You can use the following types of time in a schedule rule:

- Absolute: The point in time is defined by absolute values and is expressed as a date and time value. An example is January 15 at 9 a.m.
- Conditional: The point in time is when something happens or after something happens. Examples include "10 days after contract signing" and "upon contract expiration." A schedule rule can also be based upon contingent events, such as "injury accident" or "earthquake."
- Generic: The point in time is a day of the month or day of the week. Examples include "every first Monday at 10:00 a.m." and "the third Wednesday of June."
- Recurring: The point in time that reoccurs at intervals, such as every 6 months or every 30 days.

You can view scheduled events for a contract in the contract navigator. If you created tasks, then you can also use the administration windows in the Task Manager.

Renewal Rules

You can use the rules to determine the behavior of contracts and lines when a contract is renewed.

At the contract level you can choose between the following options:

- Active contract: The renewed contract is automatically approved.
- Do not renew: The contract cannot be renewed.
- Notify Salesrep: The renewed contract is in entered status and the salesrep is notified, to review the contract and negotiate the terms and conditions before submitting for approval.
- Submit for Approval: The renewed contract is in entered status, the contract is automatically submitted for approval and the salesrep is notified.

On the contract line level, you can determine the line level behavior when a contract is renewed:

- Do not renew: The renewal process does not renew this line when renewing the contract.
- Keep duration: The renewal process does not adjust the effective date range of the line, if the renewed contract has a longer duration than the originating contract.
- Full duration: The renewal process adjusts the line duration to the duration specified in the contract header.

For each line, you can override the renewal rule type. For example, while the entire contract may specify that all lines are adjusted to full duration, a rule on the line level may specify to exclude a certain line from the renewal process.

Understanding Oracle Order Capture Integration

You use Oracle Contracts Core integrated with Oracle Order Capture to perform the following actions:

- Create Quotes, Contracts and Orders
- Renew contracts

Create Quotes, Contracts and Orders

Oracle Order Capture is the CRM interface to Oracle Order Management. Use Oracle Order Capture to create quotes, to generate orders from a contract, and to generate contracts from quotes.

You can use existing contracts as a reference to create quotes in Oracle Order Capture. In addition, you can update the quote, for example, to reflect a new price list.

You can use contract templates to create contracts from a quote. The contract created is in entered status, which allows you to modify the contract before it is submitted for approval. Submit the concurrent program Create Order from a contract to create an order from a contract.

Renew Contracts

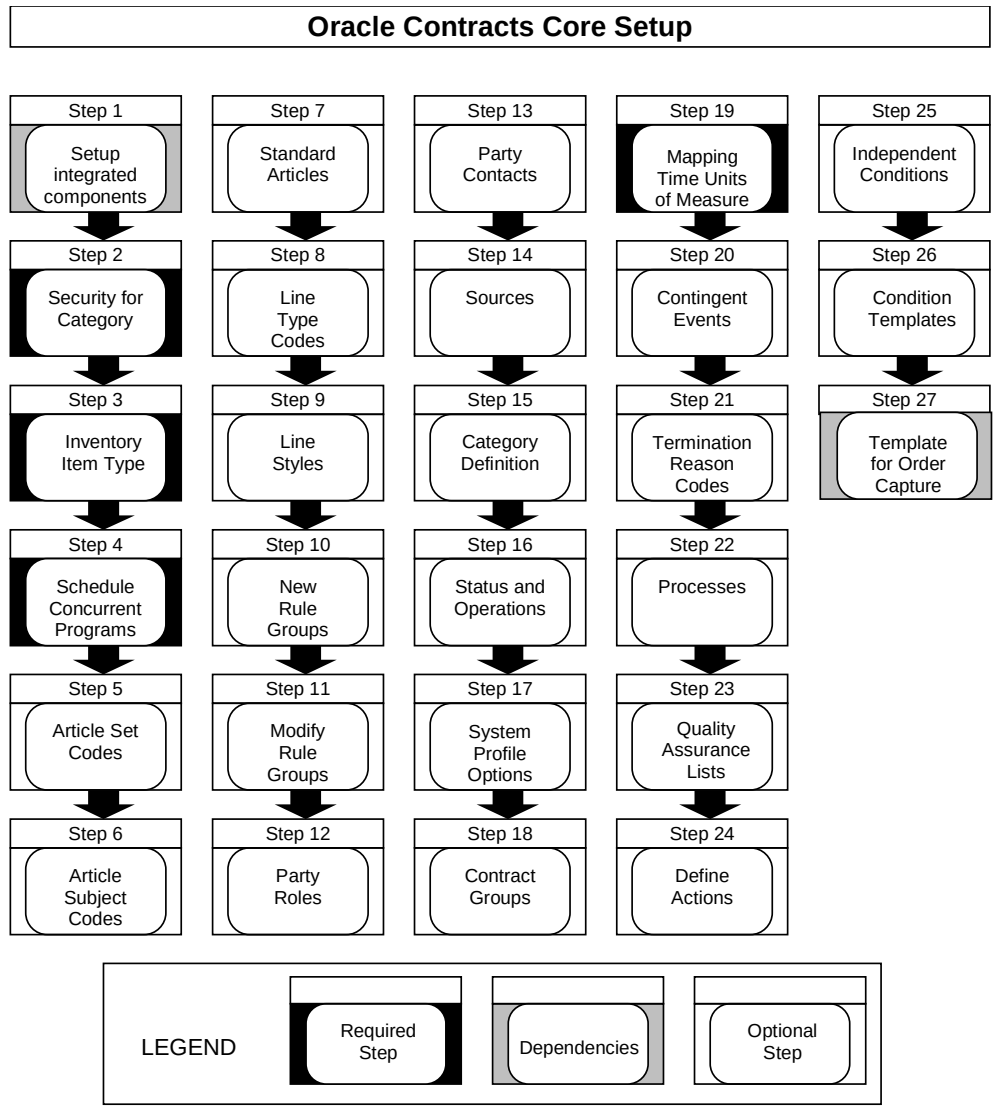
You can use Oracle Contracts Core to automatically create a quote for contracts to be renewed. Using the events component, you define a condition for the date-based action Contract Expiration. Then you specify when you want to create a quote. For example: Define "30 days before the contract expiration" as the condition. The outcome for this condition is "Create a Quote from a contract" for contract renewal.

After the quote is created, you can make changes before creating the order. Once the order is fulfilled the service contract is renewed.

Implementing Oracle Contracts Core

This topic group provides general descriptions of the setup and configuration tasks required to implement the application successfully.

l


Oracle Contracts Core Setup
Step 1
Setup integrated components
Step 2
Security for Category
Step 3
Inventory Item Type
Step 4
Schedule Concurrent Programs
Step 5
Article Set Codes
Step 6
Article Subject Codes
Step 7
Standard Articles
Step 8
Line Type Codes
Step 9
Line Styles
Step 10
New Rule Groups
Step 11
Modify Rule Groups
Step 12
Party Roles
Step 13
Party Contacts
Step 14
Sources
Step 15
Category Definition
Step 16
Status and Operations
Step 17
System Profile Options
Step 18
Contract Groups
Step 19
Mapping Time Units of Measure
Step 20
Contingent Events
Step 21
Termination Reason Codes
Step 22
Processes
Step 23
Quality Assurance Lists
Step 24
Define Actions
Step 25
Independent Conditions
Step 26
Condition Templates
Step 27
Template for Order Capture
LEGEND
Required Step
Dependencies
Optional Step

Following Setup Checklist

After the modules to be integrated are setup, you can start using contracts with few exceptions, that are listed in steps 2, 3 and 4. You can incrementally complete your setup, while starting to use contracts. We suggest that you follow the order presented here. However, for many of the steps, as suggested above, you can change and amend the setup while working with the product.

Setup Checklist

Complete the following steps in the order shown.

Step	Required	Step Title
1.	Optional	Setup Integrated Components
2.	Yes	Security for Category
3.	Yes	Inventory Item Type
4.	Yes	Schedule Concurrent Programs
5.	Optional	Article Set Codes
6.	Optional	Subject Codes
7.	Optional	Standard Articles
8.	Optional	Line Type Codes
9.	Optional	Line Styles
10.	Optional	New Rule Groups
11.	Optional	Modify Rule Groups
12.	Optional	Party Roles
13.	Optional	Party Contacts
14.	Optional	Sources
15.	Optional	Category Definition
16.	Optional	Status and Operations
17.	Optional	System Profile Options
18.	Optional	Contract Groups
19.	Yes	Map Time Unit of Measure
20.	Optional	Contingent Events

Step	Required	Step Title
21.	Optional	Termination Reason Codes
22.	Optional	Processes
23.	Optional	Quality Assurance Lists
24.	Optional	Define Actions
25.	Optional	Independent Conditions
26.	Optional	Condition Templates
27.	Optional	Template for Order Capture

Setup Steps

Step 1: Setup integrated components

The following list shows all modules and forms that Oracle contracts uses for integration:

Foundation: Define Users
Foundation: Define Responsibilities
Oracle Receivables: Define Customer
Oracle Receivables: Define Accounting and Invoice Rules
Oracle Receivables: Define Payment Terms
Oracle Payables: Define Suppliers
Oracle Payables: Define Payment Terms
Oracle Purchasing: Define Buyers
Oracle General Ledger: Define Currencies
Oracle General Ledger: Define Daily Rates
Oracle General Ledger: Define Conversion Types
Human Resources: Define Organizations
Human Resources: Assign Organization Profile Options
Human Resources: Define Locations
Human Resources: Define Persons
Inventory: Define UOM Classes
Inventory: Define UOM
Inventory: Define Item Category
Inventory: Define Master Item
Inventory: Define Item Type Codes
Inventory: Define Category Set
Inventory: Define Organization Items
Pricing: Define Price Lists

Pricing: Define Discounts
Pricing: Define Modifiers
Resources: Resource Role Type
Resources: Resource Role
Resources: Import Resources
Resources: Define Resources

Step 2: Security for Category

Oracle contracts is delivered with default categories you can use. However, you have to make sure that the responsibility you are using is given access to the category. By default, no responsibility has access to any category, unless explicitly assigned. Before you can access a contract of a specific category, you have to include the responsibility of your user in the category setup. See Defining a Category

Step 3: Inventory Item Type

To source line styles, you can use existing views, that are defined on specific inventory system item types. To use some of these integration views, you must define these time types first. See Defining Inventory System Types

Step 4: Schedule Concurrent Programs

Schedule concurrent programs to control the automated status change, the scheduling and dequeueing of events components. See Concurrent Programs

Step 5: Define Article Set Codes

Define article sets group articles. See Setting Up Lookup Codes

Step 6: Define Subject Codes

Define Subject codes to classify articles. See Setting Up Lookup Codes

Step 7: Define Standard Articles

Before defining articles, you may want to define Articles Sets and Subject codes. To define articles. See Defining the Library of Articles

Step 8: Create a new Top Line Style

The line type code defines the top line in the line style definition. See Setting Up Lookup Codes

Step 9: Define Line Styles

Before you can define a new top line style, you must define the line type code. See Defining Line Styles

Step 10: Create New Rule Groups

To create new rule groups. See Setting Up Lookup Codes

Step 11: Modify Rule Groups

There are seeded rule groups you may want to edit or you may want to create new rule groups. See Defining Rule Groups

Step 12: Define Party Roles

In addition to the seeded role like customer or vendor, you can add more roles such as OEM. See Setting Up Lookup Codes

Step 13: Define Contact Roles

In addition to the seeded contacts, you can add more contact roles such as an arbiter or a legal assistant. See Setting Up Lookup Codes

Step 14: Define Sources

The sources form allows you assign different sources for party roles and contact roles. You can modify existing source assignment or add sources to the roles you have defined in the previous two steps. See Defining Sources

Step 15: Define Categories

You can define and modify categories at any time. However, you should carefully plan your party definitions, rule groups, line styles and article definitions before defining categories. See Defining a Category

Step 16: Define Status and Operations

Define new statuses for a status type and define allowed operation for each combination of status type and category status. See Setting up Status and Operations

Step 17: System Profile Options

For a list of all system profile options see, Setting up System Profile Options.

Step 18: Define Contract Groups

Make sure you set the system profile option: OKC: Public group creator if you want create public groups. See Contract Groups

Step 19: Map Time Units of Measure

You must map your own time units defined to the contracts internal time units seeded. See Maintaining Time Units of Measure

Step 20: Define Contingent Event Codes

Contingent events may be used to schedule notifications. See Setting Up Lookup Codes

Step 21: Define Termination Reason Codes

Enter Termination reason codes. See Setting Up Lookup Codes

Step 22: Define Processes

If you want to add packages, procedures, or functions to be used in conditions or Quality Assurance checklists, you have to register them in the process definition screen. If you want to add new workflows use this procedure. See Defining a Process

Step 23: Define Quality Assurance Checklists

You can define your own Quality Assurance list, that will be executed in addition to the default Quality Assurance list. See Defining Quality Assurance Checklist

Step 24: Define Actions

Actions allow you to define relevant changes of a contract to be used for event handling. See Setting up Actions

Step 25: Define Independent Conditions

You can define conditions, that are independent of a contract, thus, conditions that evaluate for every contract. See Events

Step 26: Define Condition Templates

You can create templates for faster entry of conditions in contracts. See Defining Condition Templates

Step 27: Template for Order Capture Integration

To create a contract out of a quote, you need to define a template, from which the contract is created. See Editing a Template

Setting Up Lookup Codes

Use this list to identify lookup codes (QuickCodes) you need to define for your implementation. You can enter them in any order. Please follow the standard procedure outlined in the *Oracle Applications Users Guide*.

	Code	Description
1.	OKC_SUBJECT	Defines subject classification for articles.
2.	OKC_ARTICLE_SET	Defines sets of articles.
3.	OKC_LINE_TYPE	Defines line types.
4.	OKC_CONTACT ROLE	Defines contact roles.
5.	OKC_ROLE	Defines party roles.
6.	OKC_CONTINGENT_EVENTS	Defines contingent events for the schedule tab.
7.	OKC_TERMINATION_REASON	Defines reasons for terminating a contract. Users must specify a reason for terminating a contract.
8.	OKC_RULE_GROUP_DEF	New Codes automatically show in the Rule Group Definition form.

Setting up System Profile Options

Use this list to identify profile options you need to change for your implementation. You can set these profile options in any order you like. To change profile options, please follow the standard procedure outlined in the *Oracle Applications Users Guide*.

Step	Option	Description
1.	OKC: Schedule Rule Alert Window	The number of days before a due task, that the user is notified of upcoming task.
2.	OKC: Change Request Approval	This overrides the workflow approver.
3.	OKC: Contract Approver	This overrides the workflow approver.
4.	OKC: Schedule Rule Escalate	Number of days after a task has missed its due date when escalation begins.

Step	Option	Description
5.	OKC: Status Change Batch Size	Determines the number of records to be updated before they are saved in the database (this parameter should be fine-tuned by the database administrator).
6.	OKC: Renewed Contract Identifier	This identifier will be attached to the renewed contract's contract number.
7.	OKC: Public Group Creator	Privilege to create a public group in Oracle Contracts.
8.	OKC: Time UOM Class	Limits the units you see in the Map Time Units window.
9.	OKC: Update Negotiated Price Flag	This drives the default of the Update Negotiated Price ON/OFF in the Action menu.

Lookup Codes in the Communications Tab

Use this window to add types, action items, or outcomes in the Recording Communications between Parties.

Prerequisites

None

Steps

1. From the file menu, select **Switch Responsibility > Interaction History > Interaction History Administration.**
2. Select the respective tabs of Type, Action Item, or Outcome.
3. Enter your data.
4. Save your work.

Defining the Library of Articles

The Library of Articles is a small database of previously written and established articles. These articles can be referenced and included in a contract. Use this procedure to define a new article, edit an existing article, change an article's release, or view contracts that contain a selected article.

For more information on articles, see Contract Articles.

Prerequisites

Lookups must be defined for article subjects (OKC_SUBJECT) and article sets (OKC_ARTICLE_SET).

Steps

1. From the Navigator, choose **Setup > Contract > Standard Articles.**

   The Standard Articles Library window displays a tree of existing article sets and the articles within each set.

2. Select an article set from the tree.

3. If you want to edit the article set, then edit the description, choose categories, and optionally choose price types to control the value selection when authoring a contract that contains articles from the article set.

4. If you want to edit or review an existing article, then select the article in the set.

5. If you want to add a new article, then from the Actions Menu, choose **New Article**, enter a name for the article, and select an article subject.

   The new article is automatically placed in the selected article set.

6. If you are either entering the first release for a new article or entering a new release for an existing article, then in the Releases tab enter the release name, active date, and description.

7. Click **Release Text** to enter or edit the text.

8. If you want to view contracts that use the release of the existing article, then click **Used in Contracts.** A list of contracts appears.

9. If you want to declare incompatibilities with other articles, then in the Incompatibilities tab select the incompatible articles.

10. If you want to add the article to other article sets, then in the Article Sets tab select the article sets.

Guidelines

The end date of a release is automatically determined by the start date of the next release.

Creating new releases of articles is a way to maintain different versions of an article. Creating a new release does not impact articles in other contracts. To change an article already in use, you must create a new release, which automatically becomes the most current version after the previous release expires.

Articles can be defined to be incompatible with each other. For example, you could make a payment term of 30 days net incompatible with a 10% advance payment. You cannot define an incompatibility before both incompatible articles are defined.

The article set in the Standard Articles Library window has two tabs. On one tab you record the pricing type. On the other tab you add a category.

If you select a standard article, then the tabs change to Price Types and Categories. In the Price Types tab, you can choose price types for your article set. When you author a contract, then you can reduce the article sets available by price type.

When you assign a category to a standard article set, then this article set appears in the list of values when you author a contract that is assigned the category. You can then drill down to articles from that set. However, the authoring window does not limit the articles you select to sets that are assigned to a category. You can still access articles directly when authoring, even though their article set is not assigned to the current category.

Defining a Process

Use this procedure to name the Oracle Workflow processes or procedures and packages you have set up so that they can be understood and used by others to author or modify a contract.

Prerequisites

You must create the Oracle Workflow processes or procedures and packages first.

Steps

1. From the Navigator, choose **Setup > Contract > Process Definition.**
2. Enter a unique name and a description.
3. Enter the usage.
4. Select the type of process.
5. Enter information appropriate for the process you are defining.
6. Save your work.

Guidelines

The description appears in the list of values in windows where you need to select a defined process.

You can select one of the following usages:

- API (Application Programming Interface)
- Contract approval process
- Change request process
- Function, for condition evaluations
- Outcome, for Events
- Quality Assurance

When entering a workflow or package and procedure names, make sure the spelling is accurate. The Process Definitions window does not validate the existence or the valid status of the referred objects.

When defining outcomes for events you cannot use the type script. Script is not referring to a SQL script, but is referring to an internal scripting tool used for Oracle Relationship Planner.

When adding parameters, make sure that they are entered in the correct order.

Maintaining Time Units of Measure

Oracle Contracts Core defines unit of measure conversion for time differently from Oracle Applications. This helps ensure that the scheduling is more accurate than a simple conversion such as 1 month = 30 days, which is only correct for 5 out of the 12 months of a year.

If you want to define your own time unit conversions for extending a contract or for scheduling, then you must define your own time unit conversions. For example, lets say you want to define the 4 time units in the Units of Measure menu: Day, Month, Quarter, and Year. The UOM Class is Time. The system profile option should be set: OKC: Time UOM Class to the value of Time. When you use the Map Time Units window, the list of values will only show the time units you assigned to the conversion class Time. When you map your own time units to the internal time units:

- Day: 1 Day
- Month: 1 Month
- Quarter: 3 Months
- Year: 1 Year

With this setup, your will only be able to select the values listed above from the LOV when defining time components. You will not have access to other time units such as hour or minute.

Prerequisites

Define your units of measure classes and units of measure using Oracle Inventory.

Choose the unit of measure class to be used for contracts, and select it for the profile option OKC: Time UOM Class. The units of measure within the chosen class appear in the list of values for User Unit in the Map Time Units window.

Steps

1. From the Navigator, choose **Setup > Contract > Units of Measure > Time Units of Measure.**
2. Select a user unit of measure from the list of units of measure.
3. Select the base unit of measure that equals the user unit of measure.
4. If needed, enter conversion information.
5. Optionally, enter a description.
6. Save your work.

Guidelines

There are six internal time units: minutes, hours, days, weeks, months, and years. Make sure to map each time unit you want to use in Oracle Contracts Core. An example of a mapping: Day (your definition) = 1 day (base definition)

Setting Up Status and Operations

You can control the operations (such as update on line and delete contract) that can be performed on a contract depending upon the category assigned to the contract when it is created and the status of the contract (such as active or terminated). Use this procedure to add statuses and to define by category and status which operations can be executed against a contract.

For more information on status and operations, see Understanding Status and Operations and Access to Contracts and Security.

Steps

1. From the Navigator, choose **Setup > Contract > Status and Operations**
2. Select a status type.

3. Optionally, enter additional statuses for the status type, enter the text to display in application windows in the Meanings field, and select the default status for the contract when it first reaches the stage of the selected status type.
4. For each status, select every category, operation, and level (header or line level) combination you want to relate to the status.
5. For each line you created in the Allowed Operations by Category section, select Allowed to allow the operation. Clear Allowed to prohibit the operation.
6. Save your work.

   The information you set up is used by the application to control operations for contracts.

Guidelines

If you define a new status for the status type Active, then you have to make sure that you specifically allow operations such as on-line update. If you create a new status without specifying any allowed operations, then you implicitly allow no operations for this contract status.

In order for the concurrent program Status Change to automatically update contract status you must define a default status for each status type.

Defining Quality Assurance Checklist

Use this procedure to define a new Quality Assurance (QA) checklist. Oracle Contracts Core validates a contract before you can submit it for approval using a QA checklist. Each checklist consists of one or more Oracle Workflow processes.

Prerequisites

You must define the Oracle Workflow processes before adding them to the checklist.

Steps

1. From the Navigator, choose **Setup > Contract > Quality Assurance.**
2. Enter a name and a description.
3. In the Processes region, select the process that will become a part of the QA checklist.
4. Make sure the Active box is selected.
5. From the Severity list, select one of the following levels:
   - **Warning**: The contract passes Approval.

- **Stop**: The contract does not pass Quality Assurance if this process fails

6. Optionally, override the default values for the parameters.
7. Save your work.

Guidelines

The default Quality Assurance (QA) checklist is executed automatically for any contract, even if you create another checklist for the contract. You cannot modify or update the default checklist.

Defining a Category

A category is a type of contract, such as a license agreement or warranty contract. Use this procedure to define a category and the components of that category to be available to someone authoring a contract within that category.

Prerequisites

None

Steps

1. From the Navigator, choose **Setup** > **Contract** > **Categories** > **Define Categories.**
2. Enter the category name. The class you select determines what authoring form is used, contract or service contract. For example:

- Corporate: Oracle Core Contracts
- Master Agreement: Oracle Core Contracts
- License: Oracle Core Contracts
- Project: Oracle Project Contracts
- Service: Oracle Service Contracts

Note: For now there are no differences between using Corporate, Master Agreement or License. Corporate is the recommended default to use.

3. In the Party Roles tab, select roles that can be included in the contract and enter effective dates.
4. In the Rules tab, select rule groups you want to have access to on any of the three levels: contract, line, or articles.

5. Select a rule party role and qualify the role as object or subject to restrict the use of the role in the rule in a contract.

   This is needed because there could be multiple vendors or customers in a contract, but you only want to pay or invoice one of them. Enter the role, even if your contract has only one vendor or customer.

6. In the Line Styles tab, select top line styles.

   The tree structure of the entire selected line style is displayed. For every line of a line style, the roles entered in the Party Roles tab are shown in the Party Roles subtab and the rule groups entered in the Rules tab of the category are shown in the Rules subtab of the Line Style tab.

7. If you want a party role to be selectable on the contract line level, then in the Party Roles subtab of the Line Style tab, choose Select.

8. If you want a rule group to be selectable on the contract line level, then in the Rules subtab of the Line Style tab, choose Select.

9. In the Responsibilities tab, select at least one responsibility and assign access level rights to the responsibility along with effective dates.

10. Save your work.

Guidelines

You cannot delete a component from a category. To remove a component from the category, you have to expire it. Adding components to a category do not make the components mandatory during contract authoring.

Roles for rules is used for service contracts. Following is an example of qualifying a role for a rule: Qualify the customer role in the billing rule as the object of the billing role, that is, as the party to which the bill is sent. Or, you can determine that your billing rules identify the seller as the subject, in other words, the seller would send the invoice. The rule edit would then choose the right list of values when you enter rule information.

In the Rule tab, the optional check box determines if the qualified role may be omitted for a rule to pass Quality Assurance.

A responsibility not selected on the Responsibilities tab is prohibited access to the contract.

To access a rule from the authoring window, you must include the rule group in the category definition. You can include more than one rule group to a category.

References

Understanding Roles and Contact Sources

Understanding Contract Rules

Understanding Contract Rules

Setting up the Approval Process

Use this procedure to define the approver for both change requests and contracts if you do not have an Oracle Workflow process defined to handle approvals.

Prerequisites

You do not have an approval process defined using Oracle Workflow.

Steps

1. To set up an approver for change requests, enter the names in the system profile option: OKC: Change Request Approver. This profile options overrides a workflow setup and uses the approver, if the workflow is not yet defined.
2. To set up contract approvers, enter the names in the system profile option: OKC: Contract Approver. This profile options overrides a workflow setup and uses the approver, if the workflow is not yet defined.

Defining Line Styles

A contract is composed of lines of text. Control the type of information that can be entered on a particular line by defining a line style. The line style sets input requirements and sets up the lists of values to choose from in a contract line during contract authoring.

The most general and least detailed line in a contract is called the top line. Oracle Contracts Core delivers a set of line styles you can use, or you can create new line styles following the delivered set as a guideline. You can add line styles in a hierarchy below the top line style to control detailed information within the top line.

Prerequisites

Top Line styles must be defined as lookup codes (OKC_LINE_TYPE)

Steps

1. From the Navigator, choose **Setup > Contract > Categories > Define Line Styles.**

   The Line Styles window displays existing line styles in a tree format.

2. If you want to add a line style below an existing line style in the hierarchy, then select the parent line style.
3. Click **New** in the toolbar. This is the button with a green plus sign.

   A new node will appear in the line style tree with a generated name.
4. Enter a line style name.
5. Choose a line type to classify your line style.
6. If the line style contains a priced item, then select Priced. The tree structure to the left will mark priced items by displaying an arrow.
7. Choose one or more sources for the line style information and enter start and end dates.
8. Save your work.

   The line style is saved as part of the line style hierarchy.

Guidelines

There is one special line type: "Free Format". This line type allows you to enter ad hoc definitions that do not rely on existing sources like inventory items.

You can create a price line style at any level in the line style hierarchy. However, you can only price one line in the hierarchy chain.

For all line styles, you can determine the source of the line style. In the authoring window, lines with the price flag disabled will not show a list of values if you select system items (Inventory) as the source. If a system item line style is not priced, then the item source list of values is retrieved in the item sub tab of the line.

If a line style is for a top line for a service contract type, then the Valid Operations tab appears. You can see whether or not entitlement and invoicing are allowed for the line styles delivered for oracle services.

Entitlement and Invoicing is used for Oracle Contracts for Service only.

References

Line Styles

Defining Rule Groups

A rule group is a collection of one or many rules. Rules collect one or many related pieces of information. Rule groups control which rules are used in a contract.

Your organization can assemble rule groups to enforce business processes and company standards. Group rules so that rules with similar characteristics can be easily referenced and accessed. For example, create a rule group for billing that would group together rules for bill to address, billing schedule, and payment terms. Some rules in the group can be defined as optional.

You can use rule groups to make certain rules optional entries for one rule group, while leaving it a mandatory rule for another rule group. Assign rule groups to your contracts according to your business procedures. You can save incomplete data, but all non optional rules have to be complete to pass Quality Assurance.

Prerequisites

A rule group must be defined as a lookup (OKC_RULE_GROUP_DEF).

Steps

1. From the Navigator, choose **Setup > Contracts > Rule Groups.**

   The Rule Group Definition window appears.

2. If you want to edit a rule group, then select the rule group.
3. Add or change rule types and indicate whether or not the rule type is optional.
4. Add or change rules for each rule type.
5. Save your work.

Guidelines

When you select a rule group in your contract, all rules of that group are automatically copied into your contract. Each rule may be optional or mandatory.

For some rules, sources are provided for each intent (buy or sell). You can modify the rule type source. You must not remove an existing source definition. Adding source definitions for a new object number does not have an effect.

To access a rule from the authoring window, you must include the rule group in the category definition. You can include more than one rule group to a category.

You may have rules where the concept of object and subject is applicable. For more information see Object and Subject.

Concurrent Programs

There are two concurrent programs that need to run continuously. Make sure that the following two processes are restarted if the database or the concurrent managers have been restarted:

- Listener for Events Queue
- Listener for Outcome Queue

When you start the programs, you should accept the default parameter values, unless you are a system administrator. The parameters p_wait and p_sleep refer to technical concepts. A system administrator with experience in advanced queueing can use them to improve the system performance.

Use the concurrent request set “Oracle Contracts Core Listeners Request Set” to start both programs.

The concurrent program "Date Assembler" should run once every day, preferably later in the evening. This program is used for tasks such as automatically renewing contracts that are eligible according to the contract definition.

To process Notifications, the request set “Oracle Contracts Tasks” needs to be started. This includes four programs:

- Send Task Notifications
- First Escalation of Incomplete Tasks
- Second Escalation of Incomplete Tasks
- Action Assembler for Scheduled Planned Date

To understand how to create your own actions, functions or outcomes, see Customizing Events.

Contract Status Change

This concurrent program updates the status of contracts, like: Expiring contracts, realizing a future dated termination and activating a signed contract, See Understanding Status and Operations. You should schedule to run this concurrent program very early in the morning, so that the contract status is always synchronized with their effective dates.

Setting up Actions

Use this window to define or modify actions. Examples of actions include contract signed and Counter Group XX updated. For more information see Events.

Prerequisites

None

Steps

1. From the Navigator, choose **Contract Events > Define Action.**
2. Enter a unique name, select action type, and enter a description.
3. Enter the Correlation. Correlation is a term used in Advanced Queueing to uniquely identify the type of message in the queue.
4. If you want to enable the counter tab for this action when defining conditions, then select Counter Action.
5. If you want on line action handling, then select Allow Synchronous Outcomes. If you want asynchronous, or batch mode action handling, then clear the check box.
6. In the Basic tab, enter attributes. You must enter the name, element name or column, and a data type. These attributes can be accessed in the condition definition.
7. In the Advanced tab you can enter the following:
   - Min/Max Value: Validates the range of entries in conditions.
   - Object Name: Defines the source for the list of values in the Condition Template window.
   - Column Name: Derived from the object name.
   - Date of Interest: Can be used for action based actions only. Select an attribute when you want the condition to evaluate to true, for example, the entered signing date. If you leave the Date of Interest clear, then the signing date refers to the date when the contract signing date was entered, ignoring the value of the date entered.
   - Used in Conditions: Clear this check box if the attribute should not be used in conditions.
8. Save your work.

Guidelines

The description of the action appears in the Conditions window.

Select Enabled when you are ready to use the condition, or clear Enabled to disable a condition.

Defining Condition Templates

Use condition templates to define multiple samples of conditions. The templates can then be used to define independent conditions or conditions attached to a contract. Use this procedure to define a condition template.

For more information, see Events.

Prerequisites

Before an outcome can be assigned to a condition template, it must be defined using Setup > Contract > Process Definition.

Steps

1. From the Navigator, choose **Contract Events > Define Condition Template.**

   The Condition Template window appears**.**

2. Enter a name and a description.
3. If you want to create a task for the schedule tab in the execution overview:
   a. Check the **Create a Task** check box.
   b. Enter the task owner.
4. Select Action or Date.
5. If you are creating an action condition, then select an action.
6. If you are creating a date condition, then enter the date information.
7. Build your condition lines.

Note: When using the "LIKE" operator, you must not use quotes. For example, when building a condition, where contract_number like "%-2000%", your right value must be %-2000 and not "%-2000".

8. If you want to enter fixed values for parameters for a function, then click **Parameters** and enter the information.
9. Click **Show Condition** to display all condition lines and check their syntactical validity.

   If the condition has validated successfully, then Condition Valid is automatically selected.

10. Select **Outcomes.**

11. If you want to assign fixed values or action attribute values to the outcome parameters, then click **Parameters** and enter the information.

12. Save your work.

Guidelines

You can save your template at any time. You are not required to complete all information in the window.

An independent condition does not need to be tied to a contract and can run independent of a contract. Therefore, an independent condition has to validate before you can save it.

For condition of type action, choose if the event is evaluated only once or each time the condition is met. For example: Select Evaluate Once Only if you want to evaluate the condition to true only if the counter exceeds 1,000 for the first time. Clear the check box if you want the condition to evaluate to true each time the counter reaches 1,000.

Each condition can consist of one or more condition lines. The sequence determines the order in which the lines are processed. You can enter three types of lines.

- **Expressions**: The left value only allows you to enter attributes defined in the action attributes.
- **Counter**: Select Product or Service and choose a description of the service or product (The Counter tab does not show for Contracts Core).

  Note: If you select a counter group, then your left value lists counter group types. If you choose a counter group master, then your left value shows the related counter groups.
- **Functions**: Select Functions declared in the Process Definitions Window. To enter values for parameters, select the **Parameter** button.

Defining Sources

Use this procedure to determine the different lists of values that will appear as sources for your roles. For example, in a contract where you are buying services, you the customer should show in the list of values for the roleof the customer, though you have not defined yourself as a customer in your Oracle Receivables customer list. In this case, you define the customer for a buy contract to select an organization.

Steps

1. From the Navigator, choose **Setup > Contract > Sources.** The Role Sources window appears.

2. Select a party role.
3. In the Party Source tab, select a source for the party, select buy or sell, and enter effective dates.
4. In the Contact Sources tab, click New from the toolbar to add a new contact. A new row appears.
5. Select a contact, a source for the contact, select buy or sell, and enter effective dates. The constrained flag indicates that a relation from the contact to the party is enforced through the party ID.
6. Save your work.

Guidelines

You cannot delete a source from a role, once you have defined it. To change your source assignment, you expire the current source or contact source assignment and add the new source assignment with a start date following the expiration date of the prior assignment.

The sources you can choose from are predefined objects.

Defining Inventory System Types

Oracle Contracts allows you to use different sources for line styles. The underlying views for the linestyle sources have been defined. For example, if you want to add an item to the linestyle source "Consulting", you would have to create an inventory item with the inventory type "Consulting".

Currently the item types for the items that are be used by the views have not been defined. You need to define inventory lookup codes to define items specifically for all of the following line styles.

From an inventory responsibility within the setup menu, choose **Setup > Items > Items Type** and enter the lookup types listed:

Lookup Type	Name
CONSULTING	Consulting
SW LIC	Software License
K	Kit
MEDIA	Media Pack
TRAIN	Training

Lookup Type	Name
EDU	Education

Customizing Events

If you customize events, you are doing so at your own risk. Oracle support will not help you create or debug actions, functions or outcomes you have created. If you encounter problems with customizations and wish to obtain support, then you must recreate the problem using standard Oracle objects.

Coding Action Assemblers

In contracts business process, whenever a defined action occurs, the necessary information about the attributes needs to be assembled and placed into the Events queue. You need to code an action assembler procedure for new actions.

Following are two examples of how you code an action assembler:

Example 1 - When All the Attribute Information is Readily Available

This example accepts all attribute values as parameters. It builds the message table with the attribute name (element_name) and value (element_value). It also creates the record containing correlation for the action.

Then it calls the contracts advanced queuing procedure SEND_MESSAGE to queue the message.

This example fits when the action occurs in a form where the form contains all the required information.

Pseudo Code

```
/*==========================================================+
             Copyright (c) 1999 Oracle Corporation
               Redwood Shores, California,USA
                All rights reserved.

==========================================================*/
```

```
SET VERIFY OFF

WHENEVER SQLERROR EXIT FAILURE ROLLBACK;

CREATE OR REPLACE PACKAGE OKC_MY_ASMBLR_PVT AS

g_pkg_name CONSTANT varchar2(100) := 'OKC_MY_ASMBLR_PVT';


PROCEDURE acn_assemble(
  /* mandatory parameters do not change */
  p_api_version       IN NUMBER,
  p_init_msg_list     IN VARCHAR2 DEFAULT OKC_API.G_FALSE,
  x_return_status     OUT NOCOPY VARCHAR2,
  x_msg_count         OUT NOCOPY NUMBER,
  x_msg_data          OUT NOCOPY VARCHAR2,

 /* parameters specific to this action assembler to be added here */
  p_attribute1_value    IN DATATYPE,
  p_attribute2_value    IN DATATYPE,
  . . .
  . . .                           );

END OKC_MY_ASMBLR_PVT;
/

COMMIT;
EXIT;



/*=========================================================+
              Copyright (c) 1999 Oracle Corporation
                Redwood Shores, California,USA
                 All rights reserved.

==========================================================*/

SET VERIFY OFF

WHENEVER SQLERROR EXIT FAILURE ROLLBACK;

CREATE OR REPLACE PACKAGE BODY OKC_MY_ASMBLR_PVT AS

PROCEDURE acn_assemble(
```

```
  p_api_version       IN NUMBER,
  p_init_msg_list     IN VARCHAR2 DEFAULT OKC_API.G_FALSE,
  x_return_status     OUT NOCOPY VARCHAR2,
  x_msg_count         OUT NOCOPY NUMBER,
  x_msg_data          OUT NOCOPY VARCHAR2,

/* parameters specific to this action assembler to be added here */
  p_attribute1_value    IN/OUT DATATYPE,
  p_attribute2_value    IN/OUT DATATYPE,
  p_attribute3_value    IN/OUT DATATYPE,
  . . .
  . . .                               )  IS

 l_api_name     CONSTANT VARCHAR2(30) := 'acn_assemble';
 l_api_version   CONSTANT NUMBER := 1.0;
 i               NUMBER := 1;
 l_corrid_rec    okc_aq_pvt.corrid_rec_typ;
 l_msg_tbl       okc_aq_pvt.msg_tab_typ;
 l_msg_count     number;
 l_msg_data      varchar2(1000);
 l_return_status varchar2(1);

 /* 'MY_ACTION_CORRELATION' is the correlation of the new action for
 which this assembler is being coded */

 CURSOR cur_corr_csr IS
 SELECT aae.element_name,
        aae.format_mask format_mask
   FROM okc_actions_b acn,
        okc_action_attributes_b aae
  WHERE acn.id = aae.acn_id
    AND acn.correlation = 'MY_ACTION_CORRELATION' ;

BEGIN

 l_return_status := OKC_API.START_ACTIVITY
                    (l_api_name
                    ,p_init_msg_list
                    ,'_PROCESS'
                    ,x_return_status);

  IF l_return_status = OKC_API.G_RET_STS_UNEXP_ERROR THEN
     RAISE OKC_API.G_EXCEPTION_UNEXPECTED_ERROR;
  ELSIF l_return_status = OKC_API.G_RET_STS_ERROR THEN
     RAISE OKC_API.G_EXCEPTION_ERROR;
```

```
  END IF;

l_msg_tbl := okc_aq_pvt.msg_tab_typ();
 FOR corr_rec IN cur_corr_csr
 LOOP
 /*   'ATTRIBUTE1_ELEMENT_NAME' is the 'element name' of the action
attribute for the newly defined action */
 IF corr_rec.element_name = 'ATTRIBUTE1_ELEMENT_NAME' THEN
  l_msg_tbl.extend;
  l_msg_tbl(i).element_name     := corr_rec.element_name;
  l_msg_tbl(i).element_value    := p_attribute1_value;
 ELSIF corr_rec.element_name = 'ATTRIBUTE2_ELEMENT_NAME' THEN
   l_msg_tbl.extend;
  l_msg_tbl(i).element_name    := corr_rec.element_name;
  l_msg_tbl(i).element_value   := p_attribute2_value;

/* If the attribute is of date or number datatype and the datatype is specified
while defining the attribute then do the following */

 ELSIF corr_rec.element_name = 'ATTRIBUTE3_ELEMENT_NAME' THEN
   l_msg_tbl.extend;
   l_msg_tbl(i).element_name :=  corr_rec.element_name;
   IF corr_rec.format_mask IS NOT NULL THEN
     l_msg_tbl(i).element_value := to_char(p_attribute3_value,
                                     corr_rec.format_mask);
   ELSE
     l_msg_tbl(i).element_value := p_attribute3_value;
   END IF;

 ELSIF . . .
  . . .
  . . .
 END IF;

  i := i + 1;
 END LOOP;

 l_corrid_rec.corrid := 'MY_ACTION_CORRELATION' ;

/* This procedure puts the message on the events queue */
  OKC_AQ_PUB.send_message(p_api_version     =>'1.0'
                         ,x_msg_count       => l_msg_count
                         ,x_msg_data        => l_msg_data
                         ,x_return_status   => l_return_status
                         ,p_corrid_rec      => l_corrid_rec
```

```
                        ,p_msg_tab         => l_msg_tbl
                        ,p_queue_name      =>
                        okc_aq_pvt.g_event_queue_name);

    IF l_return_status = OKC_API.G_RET_STS_UNEXP_ERROR THEN
        RAISE OKC_API.G_EXCEPTION_UNEXPECTED_ERROR;
    ELSIF l_return_status = OKC_API.G_RET_STS_ERROR THEN
        RAISE OKC_API.G_EXCEPTION_ERROR;
    END IF;

 OKC_API.END_ACTIVITY(x_msg_count, x_msg_data);

EXCEPTION
 WHEN OKC_API.G_EXCEPTION_ERROR THEN
  x_return_status := OKC_API.HANDLE_EXCEPTIONS
                      (l_api_name,
                       G_PKG_NAME,
                      'OKC_API.G_RET_STS_ERROR',
                       x_msg_count,
                       x_msg_data,
                       '_PROCESS');
 WHEN OKC_API.G_EXCEPTION_UNEXPECTED_ERROR THEN
  x_return_status := OKC_API.HANDLE_EXCEPTIONS
                      (l_api_name,
                       G_PKG_NAME,
                      'OKC_API.G_RET_STS_UNEXP_ERROR',
                       x_msg_count,
                       x_msg_data,
                       '_PROCESS');
 WHEN OTHERS THEN
  x_return_status := OKC_API.HANDLE_EXCEPTIONS
                      (l_api_name,
                       G_PKG_NAME,
                       'OTHERS',
                       x_msg_count,
                       x_msg_data,
                       '_PROCESS');
END acn_assemble;


END OKC_MY_ACTION_ASMBLR_PVT;
/

COMMIT;
EXIT;
```

Example 2 - When Attribute Information Needs to Be Assembled

This example accepts key information about the action, for example contract_id, and collects the values for all the attributes from the database. It builds the message table with the attribute element name (element_name) and value (element_value). It also creates the record containing correlation for the action.

Then it calls the contracts advanced queuing procedure SEND_MESSAGE to queue the message.

This example fits where the action occurs in the middle of a workflow process or in a where information is not readily available.

Pseudo Code

```
/*=========================================================+
|               Copyright (c) 1999 Oracle Corporation
|                  Redwood Shores, California, USA
|                       All rights reserved
+=========================================================*/
SET VERIFY OFF
WHENEVER SQLERROR EXIT FAILURE ROLLBACK;

CREATE OR REPLACE PACKAGE OKC_MY_ACTION_ASMBLR_PVT AS
g_pkg_name     CONSTANT varchar2(100) := 'OKC_MY_ACTION_ASMBLR_PVT';

---------------------------------------------------------------------
-- PROCEDURE acn_assemble
---------------------------------------------------------------------
  PROCEDURE acn_assemble(
/* mandatory parameters do not change */
    p_api_version                 IN NUMBER,
    p_init_msg_list              IN VARCHAR2 DEFAULT OKC_API.G_FALSE,
    x_return_status               OUT NOCOPY VARCHAR2,
    x_msg_count                   OUT NOCOPY NUMBER,
    x_msg_data                      OUT NOCOPY VARCHAR2,
/* parameters essential to gather attribute values for this action to be added
here */
    p_my_param                      IN/OUT DATATYPE,
    . . .);
END OKC_MY_ACTION_ASMBLR_PVT;
/
commit;
exit;
```

```
/*=========================================================+
|               Copyright (c) 1999 Oracle Corporation
|                  Redwood Shores, California, USA
|                       All rights reserved.
+=========================================================*/

SET VERIFY OFF
WHENEVER SQLERROR EXIT FAILURE ROLLBACK;

CREATE OR REPLACE PACKAGE BODY OKC_MY_ACTION_ASMBLR_PVT AS
  g_pkg_name    CONSTANT varchar2(100) := 'OKC_MY_ACTION_ASMBLR_PVT';

  PROCEDURE acn_assemble(
/* mandatory parameters do not change */
    p_api_version                  IN NUMBER,
    p_init_msg_list                IN VARCHAR2 DEFAULT OKC_API.G_FALSE,
    x_return_status                OUT NOCOPY VARCHAR2,
    x_msg_count                    OUT NOCOPY NUMBER,
    x_msg_data                     OUT NOCOPY VARCHAR2,
 /* parameters essential to gather attribute values for this action to be added
here */
    p_my_param                      IN/OUT DATATYPE
     . . .) IS
--
    l_api_name           CONSTANT VARCHAR2(30) := 'ACN_ASSEMBLE';
    l_api_version        NUMBER := 1.0;
    l_init_msg_list      VARCHAR2(1) DEFAULT OKC_API.G_FALSE;
    l_return_status varchar2(1) := OKC_API.G_RET_STS_SUCCESS;
--
  /*Declare cursor or cursors to gather attribute values using
   p_my_param */
    CURSOR my_cur IS
    SELECT attribute1_value,
            attribute2_value,
           . . .,
           . . .
    FROM  table1,
          table2,
          . . .
    WHERE table1.id = p_my_param
    AND   . . .;
    my_rec my_cur%ROWTYPE;

    /* Declare cursor to get all attributes element_names, format masks(for
date,number datatypes) for the newly created action*/
```

```
/* 'MY_ACTION_CORRELATION' is the correlation of the new action for
 which this assembler is being coded */

    CURSOR acn_cur IS
    SELECT aae.element_name element_name,
           aae.format_mask format_mask
    FROM   okc_actions_b acn,
           okc_action_attributes_b aae
    WHERE  acn.id = aae.acn_id
   AND    acn.correlation = 'MY_ACTION_CORRELATION';
    acn_rec acn_cur%ROWTYPE;

--
   l_rec okc_aq_pvt.corrid_rec_typ;
   l_tbl okc_aq_pvt.msg_tab_typ;
   i     NUMBER := 1;
--
    BEGIN
    -- call start_activity to create savepoint, check comptability
    -- and initialize message list
       l_return_status := OKC_API.START_ACTIVITY(l_api_name
                                                ,l_init_msg_list
                                                ,'_PVT'
                                                ,x_return_status
                                                );
    -- check if activity started successfully
       IF l_return_status = OKC_API.G_RET_STS_UNEXP_ERROR THEN
          RAISE OKC_API.G_EXCEPTION_UNEXPECTED_ERROR;
       ELSIF l_return_status = OKC_API.G_RET_STS_ERROR THEN
             RAISE OKC_API.G_EXCEPTION_ERROR;
       END IF;

   l_rec.corrid := 'MY_ACTION_CORRELATION';
    l_tbl := okc_aq_pvt.msg_tab_typ();

  /*   'ATTRIBUTE1_ELEMENT_NAME' is the 'element name' of the action
  attribute1 for the newly defined action */

    FOR acn_rec IN acn_cur LOOP
      OPEN my_cur;
      FETCH my_cur INTO my_rec;
      IF acn_rec.element_name = 'ATTRIBUTE1_ELEMENT_NAME' THEN
          l_tbl.extend;
          l_tbl(i).element_name  :=  acn_rec.element_name;
       l_tbl(i).element_value := my_rec.attribute1_value;
```

```
    ELSIF acn_rec.element_name = 'ATTRIBUTE2_ELEMENT_NAME' THEN
          l_tbl.extend;
         l_tbl(i).element_name  :=  acn_rec.element_name;
          l_tbl(i).element_value := my_rec.attribute2_value;
 /* If the attribute is of date or number datatype and the datatype is specified
while defining the attribute then do the following */
        ELSIF acn_rec.element_name = 'ATTRIBUTE3_ELEMENT_NAME' THEN
          l_tbl.extend;
          l_tbl(i).element_name :=  acn_rec.element_name;
          IF acn_rec.format_mask IS NOT NULL THEN
            l_tbl(i).element_value := to_char(my_rec.attribute3_value,
                                              acn_rec.format_mask);
          ELSE
            l_tbl(i).element_value := my_rec.attribute3_value;
          END IF;

        ELSIF . . .
          . . .
        END IF;
        i := i+1;
      CLOSE my_cur;
    END LOOP;

/* This procedure puts the message on the events queue */
OKC_aq_pvt.send_message(p_api_version   =>  l_api_version
                          ,x_msg_count     => x_msg_count
                          ,x_msg_data      => x_msg_data
                          ,x_return_status => x_return_status
                          ,p_corrid_rec    => l_rec
                          ,p_msg_tab       => l_tbl
                          ,p_queue_name    =>
                           okc_aq_pvt.g_event_queue_name
                          );
    -- check if activity started successfully
       IF l_return_status = OKC_API.G_RET_STS_UNEXP_ERROR THEN
          RAISE OKC_API.G_EXCEPTION_UNEXPECTED_ERROR;
       ELSIF l_return_status = OKC_API.G_RET_STS_ERROR THEN
          RAISE OKC_API.G_EXCEPTION_ERROR;
       END IF;

    OKC_API.END_ACTIVITY(x_msg_count, x_msg_data);

    EXCEPTION
       WHEN OKC_API.G_EXCEPTION_ERROR THEN
       x_return_status := OKC_API.HANDLE_EXCEPTIONS
```

```
        (l_api_name,
         G_PKG_NAME,
         'OKC_API.G_RET_STS_ERROR',
         x_msg_count,
         x_msg_data,
         '_PVT');
        WHEN OKC_API.G_EXCEPTION_UNEXPECTED_ERROR THEN
        x_return_status := OKC_API.HANDLE_EXCEPTIONS
        (l_api_name,
         G_PKG_NAME,
         'OKC_API.G_RET_STS_UNEXP_ERROR',
         x_msg_count,
         x_msg_data,
         '_PVT');
        WHEN OTHERS THEN
        x_return_status := OKC_API.HANDLE_EXCEPTIONS
        (l_api_name,
         G_PKG_NAME,
         'OTHERS',
         x_msg_count,
         x_msg_data,
         '_PVT');
    END acn_assemble;

END OKC_K_MY_ACTION_PVT;
/
commit;
exit;
```

Coding Functions for Function Expressions in Condition Lines

The functions used in the Function tab of the Conditions window should always return T or F as a return value. The following example shows how a function expression can be used to determine if a contract belongs to a particular customer. While entering the parameters for the function expression in the Conditions window, the user can enter a value instead of selecting attributes for parameters p_role and p_customer_name. This way users can check if the contract belongs to a customer they are interested in. The condition evaluator executes this function with the user entered values at run time and returns T or F, based on which certain outcomes can be executed.

```
FUNCTION customer_exists(p_kid IN NUMBER,
                         p_customer_name IN VARCHAR2,
                         p_role    IN  VARCHAR2)
RETURN VARCHAR2 IS
```

```
CURSOR cust_cur IS
select 'X'
from  okc_k_party_roles_v role,
      okx_parties_v party
where role.object1_id1 = party.id1
and   role.object1_id2 = party.id2
and   role.jtot_object1_code in ('OKX_PARTY','OKX_OPERUNIT')
and   upper(role.rle_code) = upper(p_role)
and   role.chr_id = p_kid
and   party.name = p_customer_name;
cust_rec  cust_cur%ROWTYPE;

BEGIN
   OPEN cust_cur;
   FETCH cust_cur INTO cust_rec;
     IF cust_cur%FOUND THEN
       RETURN('T');
     ELSE
       RETURN('F');
     END IF;
   CLOSE cust_cur;
EXCEPTION
  WHEN others THEN
  RETURN('F');
END customer_exists;
```

Coding Outcomes for Conditions

All PL/SQL Outcomes that do not have the standard set of parameters listed below should have a wrapper API which provides these parameters and default values. All outcomes should have p_init_msg_list set to OKC_API.G_TRUE to initialize message stack.

p_api_version	IN NUMBER,
p_init_msg_list	IN VARCHAR2 DEFAULT OKC_API.G_TRUE,
x_return_status	OUT NOCOPY VARCHAR2,
x_msg_count	OUT NOCOPY NUMBER,
x_msg_data	OUT NOCOPY VARCHAR2,

Note, that it is not necessary to define these standard parameters in the Process Definitions form, as the Outcome Initiator will always include these parameters in the procedure call. The Outcome must not contain any non-standard OUT parameters. If these are included in an outcome it will fail on execution with a 'PLS-00363: expression '<expression>' cannot be used as an assignment target' error.

If there is a non-standard OUT parameter in the required outcome - a wrapper API will need to be written which includes a local variable where the OUT parameter can be assigned to.

```
EXCEPTION
       WHEN OKC_API.G_EXCEPTION_ERROR THEN
       x_return_status := OKC_API.HANDLE_EXCEPTIONS
       (l_api_name,
        G_PKG_NAME,
        'OKC_API.G_RET_STS_ERROR',
        x_msg_count,
        x_msg_data,
        '_PVT');
       WHEN OKC_API.G_EXCEPTION_UNEXPECTED_ERROR THEN
       x_return_status := OKC_API.HANDLE_EXCEPTIONS
       (l_api_name,
        G_PKG_NAME,
        'OKC_API.G_RET_STS_UNEXP_ERROR',
        x_msg_count,
        x_msg_data,
        '_PVT');
       WHEN OTHERS THEN
       x_return_status := OKC_API.HANDLE_EXCEPTIONS
       (l_api_name,
        G_PKG_NAME,
        'OTHERS',
        x_msg_count,
        x_msg_data,
        '_PVT');
    END;
```

Any messages specific to the outcome should be created in the FND_NEW_MESSAGES table. If the Outcome is a workflow, then it need not have any out parameters. The errors in workflow will be handled by standard workflow functionality.

Coding Date Based Action Assemblers

Users can define actions that are not triggered by originating actions, but are related to a certain point in time, such as the date when contracts are about to expire, or a customer's anniversary date within the specified period, and so on. To process all the date based actions, a concurrent manager is scheduled to run once everyday to run the concurrent program "Date Assembler" which checks for all records that satisfy the date condition and builds a message table with all the attributes and values which is put on the queue for evaluation.

Following is an example to code a date based action assembler for contracts that are about to expire.

- All these actions need to be defined along with correlation ex: KEXPIRE and attributes that can be derived from the information provided by the action, such as condition header id, contract category, contract class, contract expiry date, contract id, contract number, contract number modifier, contract status, estimated amount, and so on using the Define Action form.
- Using the Define Condition form you define whether the event is a date based event. The exact point in time must be indicated: For example “45 days before contract expiration date”.

Refer to Oracle Applications Developer’s Guide for information on Concurrent programs. The Concurrent program date assembler scheduled to run once everyday selects a contract or all contracts (for independent conditions) expiring before or after the given number of days from the current date.

Pseudo Code

Date action assembler pub

```
/*=========================================================+ Copyright (c)
1999 Oracle Corporation
Redwood Shores, California, USA
All rights reserved.
+=========================================================*/
SET VERIFY OFF
WHENEVER SQLERROR EXIT FAILURE ROLLBACK;
CREATE OR REPLACE PACKAGE BODY MY_DATE_ASMBLR_PUB AS
PROCEDURE conc_mgr(errbuf  OUT VARCHAR2,
                                          retcode OUT VARCHAR2) IS

  l_api_name        CONSTANT VARCHAR2(30) := 'conc_mgr';
  l_api_version     CONSTANT VARCHAR2(30) := 1.0;
  l_return_status   VARCHAR2(1)  := OKC_API.G_RET_STS_SUCCESS;
  x_return_status   VARCHAR2(1)  := OKC_API.G_RET_STS_SUCCESS;
  l_msg_count       NUMBER;
  l_msg_data        VARCHAR2(1000);
  l_init_msg_list   VARCHAR2(3) := 'F';

  BEGIN
       -- call start_activity to create savepoint, check comptability
       -- and initialize message list
       l_return_status := OKC_API.START_ACTIVITY(l_api_name
```

```
                                                    ,l_init_msg_list
                                                    ,'_PUB'
                                                    ,x_return_status
                                                    );
    --Initialize the return code
    retcode := 0;
MY_DATE_ASMBLR_PUB.date_expire_asmblr(
          p_api_version    => l_api_version
                ,p_init_msg_list  => l_init_msg_list
                ,x_return_status  => l_return_status
                ,x_msg_count     => l_msg_count
     ,x_msg_data     => l_msg_data);

IF l_return_status = OKC_API.G_RET_STS_UNEXP_ERROR THEN
RAISE OKC_API.G_EXCEPTION_UNEXPECTED_ERROR;
ELSIF l_return_status = OKC_API.G_RET_STS_ERROR THEN
RAISE OKC_API.G_EXCEPTION_ERROR;
ELSIF l_return_status = OKC_API.G_RET_STS_SUCCESS THEN
               commit;
       END IF;
OKC_API.END_ACTIVITY(l_msg_count, l_msg_data);
EXCEPTION
     WHEN OKC_API.G_EXCEPTION_ERROR THEN
             retcode := 2;
              errbuf  := substr(sqlerrm,1,200);
       l_return_status := OKC_API.HANDLE_EXCEPTIONS
       (l_api_name,
        G_PKG_NAME,
        'OKC_API.G_RET_STS_ERROR',
        l_msg_count,
        l_msg_data,
        '_PUB');
       WHEN OKC_API.G_EXCEPTION_UNEXPECTED_ERROR THEN
retcode := 2;
errbuf  := substr(sqlerrm,1,200);
       l_return_status := OKC_API.HANDLE_EXCEPTIONS
       (l_api_name,
        G_PKG_NAME,
        'OKC_API.G_RET_STS_UNEXP_ERROR',
        l_msg_count,
        l_msg_data,
        '_PUB');
       WHEN OTHERS THEN
retcode := 2;
errbuf  := substr(sqlerrm,1,200);
```

```
    l_return_status := OKC_API.HANDLE_EXCEPTIONS
    (l_api_name,
     G_PKG_NAME,
     'OTHERS',
     l_msg_count,
     l_msg_data,
     '_PUB');
END conc_mgr;

PROCEDURE date_expire_asmblr(
  p_api_version                  IN NUMBER DEFAULT 1.0,
  p_init_msg_list                IN VARCHAR2 DEFAULT OKC_API.G_FALSE,
  x_return_status                OUT NOCOPY VARCHAR2,
  x_msg_count                    OUT NOCOPY NUMBER,
  x_msg_data                     OUT NOCOPY VARCHAR2) IS

  l_api_name            CONSTANT VARCHAR2(30) := 'date_assemble';
 l_return_status     VARCHAR2(1) := OKC_API.G_RET_STS_SUCCESS;

  --Get all the action and condition details for a date based event
  CURSOR acn_csr IS
  SELECT acn.correlation,
         acn.acn_type,
         cnh.id,
         cnh.dnz_chr_id,
         cnh.cnh_variance,
         cnh.before_after
  FROM okc_condition_headers_v cnh,
       okc_actions_v acn
  WHERE cnh.acn_id = acn.id
  AND acn.acn_type = 'DBA'
  AND cnh.condition_valid_yn = 'Y'
  AND cnh.template_yn = 'N';

  acn_recacn_csr%ROWTYPE;

  BEGIN
  -- call start_activity to create savepoint, check comptability
  -- and initialize message list
     l_return_status := OKC_API.START_ACTIVITY(l_api_name
                                              ,p_init_msg_list
                                              ,'_PUB'
                                              ,x_return_status
                                              );
     -- check if activity started successfully
```

```
       IF l_return_status = OKC_API.G_RET_STS_UNEXP_ERROR THEN
          RAISE OKC_API.G_EXCEPTION_UNEXPECTED_ERROR;
       ELSIF l_return_status = OKC_API.G_RET_STS_ERROR THEN
             RAISE OKC_API.G_EXCEPTION_ERROR;
       END IF;

--If the action type is date based action and the correlation is Contract
expiry date
--then call the date action assembler to process all the contracts expiring
before or after
--the given number of days
       FOR acn_rec IN acn_csr LOOP
             IF acn_rec.acn_type = 'DBA' and acn_rec.correlation = 'KEXPIRE'
THEN
--Call the date assembler for contract expiry date
    MY_EXP_DATE_ASMBLR_PVT.contract_exp_date_asmblr(
              p_api_version   => 1
                 ,p_init_msg_list        => 'F'
                 ,x_return_status        => x_return_status
                 ,x_msg_count           => x_msg_count
                 ,x_msg_data             => x_msg_data
                 ,p_cnh_id                 => acn_rec.id
                 ,p_dnz_chr_id          => acn_rec.dnz_chr_id
                 ,p_cnh_variance       => acn_rec.cnh_variance
                 ,p_before_after         => acn_rec.before_after);
       IF x_return_status = OKC_API.G_RET_STS_UNEXP_ERROR THEN
          RAISE OKC_API.G_EXCEPTION_UNEXPECTED_ERROR;
       ELSIF x_return_status = OKC_API.G_RET_STS_ERROR THEN
RAISE OKC_API.G_EXCEPTION_ERROR;
ELSIF x_return_status = OKC_API.G_RET_STS_SUCCESS THEN
commit;
       END IF;
ELSIF  acn_rec.acn_type = 'DBA' and acn_rec.correlation = 'CORRELATION' THEN
--Call the  date assembler for --------------------
  END IF;
      END LOOP;
    OKC_API.END_ACTIVITY(x_msg_count, x_msg_data);

    EXCEPTION
       WHEN OKC_API.G_EXCEPTION_ERROR THEN
       x_return_status := OKC_API.HANDLE_EXCEPTIONS
       (l_api_name,
        G_PKG_NAME,
        'OKC_API.G_RET_STS_ERROR',
        x_msg_count,
```

```
        x_msg_data,
        '_PUB');
       WHEN OKC_API.G_EXCEPTION_UNEXPECTED_ERROR THEN
       x_return_status := OKC_API.HANDLE_EXCEPTIONS
       (l_api_name,
        G_PKG_NAME,
        'OKC_API.G_RET_STS_UNEXP_ERROR',
        x_msg_count,
        x_msg_data,
        '_PUB');
       WHEN OTHERS THEN
       x_return_status := OKC_API.HANDLE_EXCEPTIONS
       (l_api_name,
        G_PKG_NAME,
        'OTHERS',
        x_msg_count,
        x_msg_data,
        '_PUB');
    END date_expire_asmblr;
END MY_DATE_ASMBLR_PUB;
/
commit;
exit;
```

Date Action Assembler

For coding MY_EXP_DATE_ASMBLR_PVT Refer to Coding Action assemblers, example 2 when attribute information needs to be assembled.

```
/*=========================================================+
|       Copyright (c) 1999 Oracle Corporation
|         Redwood Shores, California, USA
|            All rights reserved.
+=========================================================*/

SET VERIFY OFF

WHENEVER SQLERROR EXIT FAILURE ROLLBACK;

CREATE OR REPLACE PACKAGE BODY MY_EXP_DATE_ASMBLR_PVT AS

  PROCEDURE date_assemble(
    p_api_version IN NUMBER,
```

```
    p_init_msg_list     IN VARCHAR2 DEFAULT OKC_API.G_FALSE,
    x_return_status     OUT NOCOPY VARCHAR2,
    x_msg_count         OUT NOCOPY NUMBER,
    x_msg_data          OUT NOCOPY VARCHAR2,
    p_cnh_id               IN NUMBER,
    p_dnz_chr_id        IN IN NUMBER,
    p_cnh_variance     IN IN NUMBER,
    p_before_after       IN VARCHAR2) IS


    l_api_name            CONSTANT VARCHAR2(30) := 'date_assemble';
    l_return_status       VARCHAR2(1) := OKC_API.G_RET_STS_SUCCESS;
    k                     NUMBER := 1;
    l_last_rundate DATE;
    l_variance                 okc_condition_headers_b.cnh_variance%TYPE;
    l_corrid_rec      okc_aq_pub.corrid_rec_typ;
    l_msg_tbl         okc_aq_pub.msg_tab_typ;

    --Get the last rundate of the date assembler concurrent program
    CURSOR last_rundate_csr IS
    SELECT nvl(max(req.actual_completion_date), (sysdate - 1)) last_rundate
    FROM   fnd_concurrent_programs prg,fnd_concurrent_requests req
    WHERE  prg.application_id = req.program_application_id
    AND    prg.concurrent_program_id = req.concurrent_program_id
    AND    prg.concurrent_program_name = 'CONCURRENT_PROGRAM_NAME'
    AND    req.status_code <> 'E';

    --Get all the contracts that are expiring from the last rundate
    --of the date assembler concurrent program for a given condition
    CURSOR attribute_value_csr(p_chr_id IN NUMBER, p_variance IN NUMBER, p_last_
rundate IN DATE) is
    SELECT a.attr_value1,
           a.attr_value2,
           a.attr_value3,
               -------------
    FROM   table1 a, table2 b
    WHERE (a.id = p_chr_id OR p_chr_id IS NULL)
    AND ------------------------------
    AND trunc(a.end_date) BETWEEN
    (p_variance + trunc(p_last_rundate) + 1)
    AND (trunc(sysdate) + p_variance);

    CURSOR elements_csr IS
    SELECT aae.element_name
    FROM   okc_actions_v acn,okc_action_attributes_v aae
```

```
    WHERE  acn.id = aae.acn_id
    AND    acn.correlation = 'CORRELATION_NAME';

    elements_rec elements_csr%ROWTYPE;

    BEGIN
    -- call start_activity to create savepoint, check comptability
    -- and initialize message list
       l_return_status := OKC_API.START_ACTIVITY(l_api_name
                                                 ,p_init_msg_list
                                                 ,'_PVT'
                                                 ,x_return_status
                                                 );
    -- check if activity started successfully
       IF l_return_status = OKC_API.G_RET_STS_UNEXP_ERROR THEN
          RAISE OKC_API.G_EXCEPTION_UNEXPECTED_ERROR;
       ELSIF l_return_status = OKC_API.G_RET_STS_ERROR THEN
             RAISE OKC_API.G_EXCEPTION_ERROR;
       END IF;

    --Get the correlation
    SELECT acn.correlation INTO l_corrid_rec FROM okc_actions_v acn
    WHERE acn.correlation = 'CORRELATION_NAME';

    --Fetch the last rundate of the date assembler concurrent program
    OPEN last_rundate_csr;
    FETCH last_rundate_csr into l_last_rundate;
    CLOSE last_rundate_csr;

        --Check if the variance is positive or negative
IF p_before_after = 'A' THEN
l_variance := p_cnh_variance * -1;
ELSIF p_before_after = 'B' THEN
l_variance := p_cnh_variance;
END IF;

      --get contract details
      FOR attribute_value_rec in attribute_value_csr(p_chr_id       => p_dnz_
chr_id,
                           p_variance     => l_variance,
                           p_last_rundate => l_last_rundate) LOOP
          k := 1;
   l_contract_number := k1_rec.k_number;
   --Initialize the table
   l_msg_tbl   := okc_aq_pvt.msg_tab_typ();
```

```
          FOR elements_rec IN elements_csr LOOP
--Build the elements table
    IF elements_rec.element_name = 'ATTRIBUTE1_ELEMENT_NAME' THEN
    l_msg_tbl.extend;
       l_msg_tbl(k).element_name  := elements_rec.element_name;
      l_msg_tbl(k).element_value := attribute_value_rec.attr_val1;
  ELSIF elements_rec.element_name = 'ATTRIBUTE2_ELEMENT_NAME' THEN
    l_msg_tbl.extend;
       l_msg_tbl(k).element_name  := elements_rec.element_name;
       l_msg_tbl(k).element_value := attribute_value_rec.attr_val2;
  ELSIF elements_rec.element_name = 'ATTRIBUTE3_ELEMENT_NAME' THEN
    l_msg_tbl.extend;
       l_msg_tbl(k).element_name  := elements_rec.element_name;
       l_msg_tbl(k).element_value := attribute_value_rec.attr_val3;
    ELSIF ------------------------------------------------------------
   --Increment the counter
k := k + 1;
END LOOP;

--Call the Advanced queue procedure to put the message on the queue
OKC_AQ_PUB.send_message(p_api_version      =>'1.0'
             ,p_init_msg_list    => 'F'
                                    ,x_msg_count       => x_msg_count
                                   ,x_msg_data         => x_msg_data
                                   ,x_return_status    => l_return_status
                                   ,p_corrid_rec        => l_corrid_rec
                                  ,p_msg_tab           => l_msg_tbl
                                   ,p_queue_name     => okc_aq_pvt.g_event_
queue_name);
IF l_return_status = OKC_API.G_RET_STS_UNEXP_ERROR THEN
          RAISE OKC_API.G_EXCEPTION_UNEXPECTED_ERROR;
      ELSIF l_return_status = OKC_API.G_RET_STS_ERROR THEN
            RAISE OKC_API.G_EXCEPTION_ERROR;
ELSIF l_return_status = OKC_API.G_RET_STS_SUCCESS THEN
commit;
       END IF;
    END LOOP;

    OKC_API.END_ACTIVITY(x_msg_count, x_msg_data);

    EXCEPTION
       WHEN OKC_API.G_EXCEPTION_ERROR THEN
       x_return_status := OKC_API.HANDLE_EXCEPTIONS
       (l_api_name,
        G_PKG_NAME,
```

```
        'OKC_API.G_RET_STS_ERROR',
        x_msg_count,
        x_msg_data,
        '_PVT');
      WHEN OKC_API.G_EXCEPTION_UNEXPECTED_ERROR THEN
      x_return_status := OKC_API.HANDLE_EXCEPTIONS
      (l_api_name,
       G_PKG_NAME,
       'OKC_API.G_RET_STS_UNEXP_ERROR',
       x_msg_count,
       x_msg_data,
       '_PVT');
      WHEN OTHERS THEN
      x_return_status := OKC_API.HANDLE_EXCEPTIONS
      (l_api_name,
       G_PKG_NAME,
       'OTHERS',
       x_msg_count,
       x_msg_data,
       '_PVT');
    END date_assemble;

END MY_EXP_DATE_ASMBLR_PVT;
/
commit;
exit;
```

Rules For Creating Actions

1. All Actions defined should have at least one attribute.
2. Counter based Actions should have COUNTER_GROUP_ID and COUNTER_ GROUP_LOG_ID as mandatory attributes.

Customizing Workflows

Oracle Workflow enables you to automate business processes, routing information of any type according to customizable business rules. Workflow automates two procedures in Oracle Contracts:

- Contracts approval workflow
- Contracts change requests workflow

You can use Oracle Workflow Builder to customize the two workflows.

Contracts Approval Workflow

The Approval Workflow OKCAUKAP is using the process: K_APPROVAL_PROCESS.

The default approval workflow routes the approval to the user SYSADMIN, see Setting up the Approval Process. After the user SYSADMIN has approved the workflow, the initiator of the workflow receives a notification to sign the contract.

The recommendation is to modify the workflow, so that after the last approval is obtained, the signature is requested. After the contract is signed, it may get the status Active or Signed, depending on the effectivity dates of the contract.

Contracts Change Requests Workflow

The Change Request Workflow OKCADCRQ is using the process: CHK_APPROVAL_PROCESS. The default approval workflow routes the approval to the user SYSADMIN, see Setting up the Approval Process.

Registering a new Source as a JTF Object

Oracle Contracts Core allows you to define your own list of values in several forms. These list of values are called sources. You then can use these sources when defining Line Style sources or Role Sources.

The process of making any kind of data available as sources comprises two steps:

1. Define a view.

2. Integrate the view into as a JTF object.

3. Assign the usage, that is should this source be used for line style definitions, role definitions, and so on.

To explain how to register a new source as a JTF object, an example will be shown. Assume that you need a line style source that qualifies frequencies:

Value	Description
Never	Never
Seldom	1 to 4 times
Often	Ranging from 5 to not always
Always	Without a doubt, always happens

Define a View

To define a view that can be integrated into a JTF object, the view must have the format outlined below:

Name	Null	Type
ID1	NOT NULL	NUMBER
ID2	NOT NULL	NUMBER
NAME	NOT NULL	VARCHAR2(11)
DESCRIPTION	NOT NULL	VARCHAR2(31)
PRIMARY_UOM_CODE	NOT NULL	VARCHAR2(0)
STATUS	NOT NULL	CHAR(1)
START_DATE_ACTIVE		DATE
END_DATE_ACTIVE		DATE

ID1 and ID2 ensure the uniqueness of the rows. If ID1 is selective enough to be unique, you can fill in a dummy character like '#' as ID2.

To define the view OKX_FREQ_V, following the mandatory format, run the following script:

```
CREATE OR REPLACE VIEW okx_freq_v AS
SELECT 1 ID1, '#' ID2, 'Often' NAME, 'ranging from 5 to not always' DESCRIPTION,
'A' STATUS, sysdate START_DATE_ACTIVE, sysdate END_DATE_ACTIVE,
null PRIMARY_UOM_CODE
FROM DUAL
UNION SELECT 2 ID1, '#' ID2, 'Never' NAME, 'Never' DESCRIPTION,
'A' STATUS, sysdate START_DATE_ACTIVE, sysdate END_DATE_ACTIVE,
null PRIMARY_UOM_CODE
FROM DUAL
UNION SELECT 3 ID1, '#' ID2, 'Hardly ever' NAME, '1 to 4 times' DESCRIPTION,
'A' STATUS, sysdate START_DATE_ACTIVE, sysdate END_DATE_ACTIVE,
null PRIMARY_UOM_CODE
FROM DUAL
UNION SELECT 4 ID1, '#' ID2, 'Always' NAME,
'Without a doubt, always happens' DESCRIPTION,
'A' STATUS, sysdate START_DATE_ACTIVE, sysdate END_DATE_ACTIVE,
null PRIMARY_UOM_CODE
```

```
FROM DUAL
```

Note: For line items, you need the column primary_uom_code only if you want to price the line.

Integrate the View as a JTF Object

Use the 'CRM Task Manager' responsibility to integrate the view.

Open the Form: Task Setup: Object Types through the menu Setup: Objects Metadata.

Note: You should consult the JTF user manuals to find out more about the fields of the form. However, this section gives you an example for the typical case of registering a view as a JTF object.

1. Enter a name and description.
2. Enter a unique object code, which will be used for internal identification, in this example: OKX_FREQ.
3. Enter a start date.
4. In the 'Select Statement Details' section, enter two values, separated by a blank into the FROM field:

- The name of the view (i.e. OKX_FREQ_V)
- The name of the object code (i.e. OKX_FREQ)

5. In the order by field, enter the object code name and column name you want to order by, (i.e. OKX_FREQ.name).

Field	Field Values
Name	Frequency
Description	Frequency in non quantitative terms
Start Date	Today's date
Object Code	OKX_FREQ
From (in the Select Statement Details)	OKX_FREQ_V_OKX_FREQ
Order by (in the Select Statement Details)	OKX_FREQ.name

Assign the Usage

In the Form: **Task Setup: Object Types** you now can assign usages. The following usages have been defined for Oracle Contracts Core:

Object User	Description
OK	Contracts
OKX	Contract source
OKX_B_O_ROLE	Object constrained for buy
OKX_B_S_ROLE	Subject constrained for buy
OKX_CONTACTS	Contract contact source
OKX_CURRENCY	Constraint by currency
OKX_LINES	Contract line style source
OKX_ROLES	Contract role source
OKX_RULES	Contract rule source
OKX_S_O_ROLE	Object constrained for sell
OKX_S_S_ROLE	Subject constrained for sell

Note: Despite the fact that you can add more Object users, adding Object users beyond the users listed here will not benefit you. Only the seeded values support the standard forms and functionalities.

If you want include your new List of Values in the linestyles, you have to register the line style as an object user. Enter the OKX_LINES as the object user.

Using Contracts Core

This topic group provides process-oriented, task-based procedures for using the application to perform essential business tasks.

Working with the Contract Launchpad and Navigator

Use the Launchpad to view contracts in your inbox, recently modified contracts, or bookmarked contracts. See Understanding the Contract Launchpad and Navigator for details about the Contract Launchpad.

Use the Launchpad and the Contract Navigator tabs to access a variety of functions related to a contract. You can access most of these functions from the Contract Navigator by selecting a contract and pressing the right mouse button.

The available functions include:

- Finding a Contract
- Creating a New Contract Manually
- Creating a Contract from a Template
- Opening a Contract
- Creating a New Version of a Contract
- Viewing the History of a Contract
- Creating a Change Request
- Applying Changes and Closing the Change Request
- Copying a Contract
- Creating a Subcontract
- Renewing a Contract
- Changing the Status of a Contract
- Extending a Contract
- Extending a Contract Line
- Terminating a Contract

- Terminating Contract Lines
- Viewing the Schedule
- Recording Communications Between Parties
- Authoring Contracts
- Editing a Template

Finding Contracts

Use this procedure to find a contract when you do not know which group the contract is associated with. When the Search Templates and Contracts window appear, you can select a contract from the Results region and right click to open the contract.

Steps

1. From the Navigator, choose **Launch Contracts.** The Oracle Contracts window appears.
2. From the View menu, select Find. The Search Templates and Contracts window appears.
3. Enter your search criteria and click the Search button.
4. Select a contract from the Results region and right click to open the contract.

Creating a New Contract Manually

Use this procedure to create a new contract manually. To create a contract using a template follow the procedure outlined in Creating a Contract from a Template.

Prerequisites

A contract template must exist for the contract category.

Steps

1. From the Navigator, choose **Launch Contracts.** The Oracle Contracts window appears.
2. In the Contract Navigator tab, right-click anywhere and choose **New.** The Create New Contract window appears.
3. Select **Create a new Contract Manually.**
4. Select a category.
5. Click **Create**.

The Contract Authoring window appears. You are now ready to author the contract. Go to Authoring Contracts.

Creating a Contract from a Template

Use this procedure to create a contract from a template. To create a contract without a template follow the procedure outlined in Creating a New Contract Manually.

Prerequisites

None

Steps

1. From the Navigator, choose **Launch Contracts.** The Oracle Contracts window appears.
2. In the Contract Navigator tab, right-click anywhere and choose **New.** The Create New Contract window appears.
3. Select **Create a Contract from a Template.**
4. Select a template.
5. If you need to search for a template, then:
   a. Click **Find.**
   b. In the Search Templates Window enter the search criteria.
   c. Select a template.
   d. Click **Open**.
6. Click **Create**. The Contract Authoring window appears. You are now ready to author the contract. Go to Authoring Contracts.

Opening a Contract

Use this procedure to open an existing contract.

Prerequisites

None

Steps

1. From the Navigator, choose **Launch Contracts.** The Oracle Contracts window appears.

2. In the Contract Navigator tab, select a contract group.
3. Right-click a contract and choose **Open.** The Contract Authoring window appears. You are now ready to author the contract. Go to Authoring Contracts.

Creating a New Version

Use this procedure to create a new version of a contract.

Prerequisites

None

Steps

1. From the Navigator, choose **Launch Contracts.** The Oracle Contracts window appears.
2. In the left window of the Contract Navigator tab, select a contract group.
3. Right-click a contract and choose **Create New Version.** This creates a new version of your contract.

Guidelines

When you create a new version, the digit in front of the period (the main version), is increased by one. The minor version is reset to zero.

Viewing the History of a Contract

Use this procedure to view older versions of a contract.

Prerequisites

None

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select a contract group.
3. Double-click the contract for which you want to see a previous version. A summary window appears.
4. Select the History tab.

5. In the Select list, select **Contract Versions** to see all versions. All major versions are displayed.
6. Select the version you want to open.
7. Click **Open**. You now can view the version of the contract.

Creating a Change Request

Change Requests are a formal way to document changes to a contract. A request is approved by the approver defined in the workflow selected and submitted for the change request. Once your change request is approved, the administrator entered in the change request receives a notification that the contract can be changed.

An internal key gives the change request administrator update access to the contract. When the administrator changes, it is locked for other change requests until the change request is closed. See Applying Changes and Closing the Change Request.

Prerequisites

The status and operations must allow changes for that category and status.

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select the contract you want to change.
3. Right-click on the contract and choose **Create Change Request.** The Create a Change Request window opens.
4. Enter the request name and a description.
5. Enter the requester's name. This person will receive a workflow notification when changes have been approved and can be applied to the contract.
6. Enter the request date and effective dates.
7. If you want to version the changed contract, then select Version.
8. If you want to have a signature required, then select Signature Required.
9. Record the lines where you are requesting a change and the exact nature of the change.
10. Select the Admin tab.
11. Select the Approval Workflow.
12. Select Party Approvals.

13. Click the submit Submit for Approval button.

Applying Changes and Closing the Change Request

When you receive a notification to apply changes, open the contract and apply the changes as described in the change request notification. Once you start editing the contract, you automatically lock the contract for other users with an approved change request.

Prerequisites

In order to apply changes, the change request notification must still be open in your inbox. No other change request must be in process.

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Launchpad tab, select the approved change request notice in your inbox and click Open. The Contract Authoring window appears.
3. Edit the contract according to the approved changes.
4. In the Contract Navigator tab of the Oracle Contracts window, select the contract you changed.
5. Double-click the contract. The contract window appears.
6. In the History tab, select the change request.
7. Optionally, right-click the contract and select **Change Request.** This opens the change request menu where you can then query for a change request.
8. Enter the current date in the Apply Date field and save your entry. You closed the change request and released the lock on the contract.

Copying a Contract

There are several ways to copy a contract, parts of a contract, or multiple contracts.

Prerequisites

None

Steps

1. Navigate to the Copy window using one of the following ways:

- In the Contract Navigator, select a contract group, right-click a contract, and choose **Copy.**
- Open an existing contract, and in the Authoring window, choose **Copy from Contract/Template** from the Action menu.

2. In the Copy window, enter the new contract name, and optionally, a contract modifier.
3. From the left side, select the contract or part of the contract you want to copy.
4. Click the double right-arrow button to move the existing contract or parts on the left side to the right side. Anything on the right side is considered part of the new contract.
5. Optionally, click the double left arrow button to move a new contract or parts of a contract back to the existing contract area on the left side.
6. Click **Create**.

   Your new contract with copied contract information is now saved. You can then make changes to the newly created contract in the same way that you would when authoring a contract.

Creating a Subcontract

In order to fulfill obligations of an existing contract, a party can establish a new contract with a third party to provide some or all of the items (such as materials or deliverables) required in the original contract. This is called creating a subcontract. You can copy a contract or elements of a contract into a subcontract. Use this procedure to create a subcontract.

Prerequisites

A finalized, active contract must be in place.

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select a contract group.
3. Right-click a contract and choose **Subcontract.**
4. On the right side, enter the name and optionally a modifier of the new contract.
5. Select a category for the new subcontract.
6. Optionally correct the intent and enter the perspective of the contract.

7. If you want to reference to the subcontracting contract instead of copying components, then select Keep References. You cannot change referenced components in the new subcontracted contract.
8. Highlight the desired components in left navigator, click the forward arrow button and click **Create** in the subcontract components selection popup.

You can make changes to the new contract in the Contract Authoring window.

References

Defining Sources

Authoring Contracts

Renewing a Contract

Renewing a contract does not affect the current contract. It creates a new contract by copying the current contract. You can renew a contract only once. For more information see Renewal Rules.

For example: You can renew contract A100-2000 and create the renewed contract A100-2001. Then you cannot renew the contract A100-2000 again, but you can renew contract A100-2001 into contract A100-2002.

Prerequisites

The contract must be in active, signed, or expired status.

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select a contract group.
3. Select one or more contracts, right-click, and choose **Renew.**
4. To keep the new contract distinguishable, the modifier is defaulted to the value you set in the profile option OKC: Renewed Contract Identifier, with the current date appended. You can change the modifier to another value.
5. The new start date is defaulted to the date following the end date of the current contract. Either select an end date for the renewed contract or enter values for duration and period. For example, to renew a contract for six months, enter 6 in the duration and select Month in the period field.
6. Click **Renew.**

A copy of the current contract is saved.

Changing the Status of a Contract

Use this procedure to change the status or status type of a contract. The contract status determines the operations allowed in the contract and status type.

Prerequisites

None

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select a contract group.
3. Right-click a contract and choose **Change Status.**
4. Select the new status from the list of values.
5. Save your work.

References

Understanding Status and Operations

Extending a Contract

Use this procedure to extend a contract. Extending a contract changes only the date a contract is scheduled to expire and possibly changes the status of the contract and lines from expired to active.

Prerequisites

The contract must have a status of Active or Expired.

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select a contract group.
3. Select one or more contracts, right-click, and choose **Extend.** The Extend window appears.

4. Either select a new end date for the contract or enter values for duration and period. For example, to extent a contract for a further six months, enter 6 in the duration and select Month in the period field.
5. Click **Extend**.

Extending a Contract Line

Use this procedure to extend a line on an active contract. Extending a line may change the end date of a contract. When the extended line then is scheduled to expire on the new end date and if applicable, adjusts the end date of the contract.

Prerequisites

The contract line must have a status of Active or Expired.

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select a contract group.
3. Double-click a contract to see the overview.
4. Select the line or lines you want to extend.
5. Right-click and choose **Extend.** The Extend window opens.
6. Either select a new end date for the line or enter values for duration and period. For example, to extent a contract line for further six months, enter 6 in the duration and select Month in the period field.
7. Click **Extend.**

Guidelines

If you extend a contract line beyond the end date of a contract, the end date of the contract is automatically updated to the end date of the extended line.

Terminating a Contract

Use this procedure to terminate a contract that is active or effective and on hold.

Prerequisites

The contract line must have a status of Signed, Active, or Hold.

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select a contract group.
3. Select one or more contracts, right-click, and choose **Terminate.** The **Terminate** window appears.
4. Enter a termination date and a termination reason.
5. Click **Terminate.**

Guidelines

If you want to add more reasons for a termination, your system administrator can add more options in the Lookup Codes.

Terminating Contract Lines

Use this procedure to terminate a contract line in an active contract. If you want to add more reasons for a termination, your system administrator can add more options in the Lookup Codes.

Prerequisites

The contract status is Active and the contract lines have a status of Active or Hold.

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select a contract group.
3. Double-click a contract to see the overview.
4. Select the line or lines you want to terminate.
5. Right-click and choose **Terminate.** The Terminate window appears.
6. Enter a termination date and a termination reason.
7. Click **Terminate.**

Viewing the Schedule

Use this procedure to get an overview of scheduled events.

Prerequisites

None

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select the contract for which you want to see scheduled tasks or completed outcomes.
3. Double-click the contract and then select the Schedule tab.

You can use this window to record contingent events and the completion of tasks.

Recording Communications Between Parties

The Communications tab allows you to record and view communications between the parties within a contract. Such communications can range from phone calls and e-mails to formal letters sent. You can use the attachments feature to attach files or e-mails to communications. To define lookup values for the communications tab, see Lookup Codes in the Communications Tab.

Prerequisites

None

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select the contract for which you want to see or enter communications.
3. Double-click the contract and select the Communications tab.

Authoring Contracts

The Contracts Authoring window is where you create and modify contracts. You can perform the following functions:

- Entering General Contract Information
- Entering Summary Information
- Entering Contract Parties and their Contacts
- Specifying Contract Line Item Information

- Entering Article Information in a Contract
- Entering Rules
- Checking if the Contract Passes Quality Assurance
- Submitting for Approval
- Using Conditions

Entering General Contract Information

Use this procedure to enter the general information about a contract. You must enter this information before using any of the tabs in the Contracts Authoring window.

Prerequisites

None

Steps

1. Navigate to the Contracts Authoring window**.**
2. Enter the contract number.
3. If you have entered a contract number that already exists for another number, then select a modifier to distinguish this contract.
4. Select a Buy or Sell for intent.
5. Enter an end date.
6. In the Details subtab, enter at least one group name.

You are now ready to enter the rest of the contract information. Go to Entering Summary Information.

Guidelines

At any time you can save an incomplete contract as a template, using Save as Template from the Tools menu.

Editing a Template

Use this procedure to edit a contract template.

Prerequisites

None

Steps

1. From the Navigator, choose **Launch Contracts.**
2. In the Contract Navigator tab, select a contract group.
3. Right-click a template and choose **Open.**

The Contract Authoring window appears. Instead of contract number, the template name is displayed. You are now ready to modify the template. Go to Authoring Contracts.

Entering Summary Information

Use this procedure to enter summary information for a contract.

Prerequisites

None

Steps

1. Navigate to the Contracts Authoring window**.**
2. In the Effective region of the Summary tab, enter this information:
   - Intent (buy or sell)
   - Perspective
   - Start and end dates
3. In the Details tab, enter the approval workflow information.
   a. Choose QA Checklist from the list of values. Contracts automatically executes the default checklist in addition to your own checklist.
   b. Optionally, select **Auto Renew.** You must also provide the number of days before the contract is automatically renewed by a concurrent program.
   c. Select a contract group. You must assign the contract to at least one group.
4. In the Administration subtab, select your approval workflow.
5. You can give assign access to groups or individual users in the Access region.

You are now ready to enter the rest of the contract information. Go to Entering Contract Parties and their Contacts.

Guidelines

You can enter condition types and actions, quality check a contract or submit it for approval using the buttons at the bottom of the window.

At any time, you can check if your contract passes Quality Assurance. The result of the on-line Quality Assurance check is displayed as soon as it completes. There is no need to run the Quality Assurance before you submit the contract for approval, since the Quality Assurance is automatically run before submitting for approval.

Entering Contract Parties and their Contacts

In the Parties tab, you can identify the legal parties for the contract. You can also enter individuals involved with the creation and management of the contract as contacts for the parties. Sales contacts can also be entered. The information you enter depends on the category and intent.

Prerequisites

None

Steps

1. Navigate to the Contracts Authoring window.
2. In the Parties tab, select the party's role.
3. Select the party's name. You can also add the alias the party uses for the contract and the Also Known As identifiers for the party.
4. In the Party Contacts region, enter a contact, sequence, and a name of the contact. The sequence determines the order the contacts should appear.

The affirmative action fields are automatically filled. You are now ready to enter the rest of the contract information. Go to Specifying Contract Line Item Information.

Guidelines

You cannot enter new customers from the Authoring window. If you enter a contract and you need to add a new customer or a supplier, then you must first define the new customer or create a supplier.

References

Access to Contracts and Security

Specifying Contract Line Item Information

In the Line Items tab, you can detail all the contract line items, which are the contract deliverables and relevant items. The Line Items tab has three subtabs: Sublines, Rules, and, Parties. You can use the subtabs to add and exclude data specific to a line. Use this procedure to enter details for a line.

Prerequisites

If you want to use price lists, then you must enter the price list in the pricing rule in the contract Rules tab.

Steps

1. Navigate to the Contracts Authoring window.
2. In the Line Items tab, enter the line style and a number for the display sequence.
3. Depending on the line style definition, you may enter the description or select a description from the list of values.

Note: If the your line item is priced at the top level, you can enter quantity, unit of measure and price information. You can override the contract level price list by choosing a price list in the Rules subtab.

4. Enter the unit of measure.
5. If a list price is defined for the item in the price list, then the unit price is retrieved and displayed. Otherwise, you can enter the unit price. The extended price is calculated from the quantity times the unit price. The negotiated price allows you to override the extended price.

Note: The Action menu has a menu item "Update Negotiated Price". When the menu item displays Update Negotiated Price ON, the negotiated price will default to the extended price and any further changes in the extended price will reflect in the negotiated price. The user can override the negotiated price later. When the menu item shows Update Negotiated Price OFF the negotiated price will not change if the extended price changes. See Setting up System Profile Options.

6. Optionally, depending on the line type you selected, use the Sublines tab to enter more detail in the sublines.

You can enter sublines for those lines defined to contain sublines in the line style hierarchy. This window allows you to view and edit the hierarchy at any level. Navigate through levels by using the Drill Up and Drill Down buttons.

7. Enter rules on lines the same way rules are entered on the contract.
8. Use the Parties tab to add line specific information that is not valid for the entire contract. Entering a party in a subline is similar to entering party information for the contract.

You are now ready to enter the rest of the contract information. Go to Entering Article Information in a Contract.

References

Entering Rules

Line Styles

Entering Article Information in a Contract

An article is the text that describes and details the terms and conditions that are attached to a contract.You can enter standard and nonstandard articles while authoring a contract. Whenever you use a standard article within a contract, all rules that have been associated with that article are copied into that contract. See Contract Articles.

You can perform the following actions:

- Enter a Standard Article
- Enter a Non-Standard Article by copying a Standard Article
- Enter a Non-Standard Article

You can define rules groups during the initial setup of the standard article set. Whenever you use a standard article set, the rule groups defined for that set (compatible to your contract category) are copied into the contract where they can be modified. Rule groups are not copied when individual articles are selected for reference or copy.

Entering a Standard Article

Use this procedure to reference a standard article or article set while authoring a contract.

Prerequisites

None

Steps

1. Navigate to the Contracts Authoring window.
2. In the Articles tab, leave the Standard check box selected.
3. In the name field, select an article set.
4. Select the article. Optionally, double-click the article set to select only one article. If you know your article, then you can type in fragments of the article to reduce the list of values matching your search criteria. This second method bypasses the navigation through article sets.
5. Click **Reference.**
6. Optionally, click **Show Text** to modify the text.
7. Save your work.

Guidelines

When selecting an article set, the rule groups defined for that set are copied into the contract. When selecting an individual article, rules are not copied.

Entering a Non-Standard Article by Copying a Standard Article

Use this procedure to copy an existing standard article and modify it.

Prerequisites

None

Steps

1. Navigate to the Contracts Authoring window.
2. In the Articles tab, leave the Standard check box selected.
3. In the name field, select an article set.
4. Select the article. Optionally, double-click the article set to select only one article. If you know your article, then you can type in fragments of the article to reduce the list of values matching your search criteria. This second method bypasses the navigation through article sets.
5. Click **Copy.** The standard flag is automatically unchecked when you choose to copy an article instead of referencing it.

6. Optionally, click **Show Text** to modify the text.
7. Save your work.

Entering a Non-Standard Article

Use this procedure to create a non-standard article.

Prerequisites

A contract must be created, even if it is only in its initial authoring stages.

Steps

1. Navigate to the Contracts Authoring window.
2. In the Articles tab, clear the Standard check box.
3. Enter a name to describe your non-standard article.
4. Click **Show Text** and enter the article text.
5. Save your work.

Entering Rules

Use this procedure to add, edit, and delete rules at the contract header or line level while authoring a contract.

Prerequisites

None

Steps

1. Navigate to the Contracts Authoring window and select Rules tab.
2. Select a Rule Group. Rules associated with a selected Rule Group appear in the Rule region (below Rule Group). To distinguish rule groups that have the same name, you can enter a comment or description for each rule group. Depending on the rule group, you may have to enter an object and/or subject. See Object and Subject.
3. Select a Rule. If the Required check box (read only) is selected, this indicates the rule is required for that Rule Group. If the Entered check box (read only) is selected, this indicates Rule Detail Values were entered for that rule.
4. Enter a Value for the Rule Detail when the Required check box (read only) is selected.

5. Enter values as needed for other Rule Details.
6. Save your work.

References

Understanding Contract Rules

Defining Rule Groups

Using the Sections Tab

If you are integrating iStore, then you can use the Sections tab to format the way articles are displayed. You can use this tab to enter sections. For each section, you enter subsection headings and articles under a section. You can only enter articles, that have been entered in the articles tab. If you want to display articles within a subsection, then you must use the tree on the left side to change the display of the right side. When the subsection selected is displayed as a section, then you can add articles to the level now displayed in the section on the right side.

Checking If the Contract Passes Quality Assurance

Use this procedure at any time to check if your contract passes Quality Assurance. The results are displayed as soon as the check completes. This procedure is not required before submitting a contract for approval because a check is automatically made during the approval process. You can use this check to verify the completeness of your contract.

Prerequisites

You must set up Quality Assurance checks first.

Steps

1. Navigate to the Authoring window.
2. Select or author a contract.
3. Select the **Check Contract** button**.** A window displays Quality Assurance comments.

Guidelines

There is no need to run the Quality Assurance before you can submit the contract for approval.

Submitting a Contract for Approval

Use this procedure to submit a contract for approval.

Prerequisites

None

Steps

1. Navigate to the Contracts Authoring window.
2. From the Actions menu, choose **Submit for Approval,** or select the **Submit for Approval** button from the Contracts Authoring window. The Quality Assurance check executes automatically. The contract is submitted only if it passes Quality Assurance.
3. If the contract does not pass the Quality Assurance check, then you can choose to continue with the submission.

Using Conditions

You now can enter conditions in one of two ways:

- Copy condition templates into your contract.
- Create a new contract-specific condition.

References

For more information on conditions, see Events.

For more information on combining rules and conditions, see Using Conditions and Rules.

Copying from a Condition Template

Use this procedure to copy a condition template into your contract.

Prerequisites

None

Steps

1. Navigate to the Contracts Authoring window.
2. Click the **Conditions** button**.**

3. Select **Copy From Template** from the Actions Menu.
4. Use the Condition Template list of values (LOV) to enter the condition template name.
5. Modify or amend the condition for the current contract.

Creating a Contract-Specific Condition

Use this procedure to create a contract-specific condition for your contract.

Prerequisites

None

Steps

1. From the Navigator, choose **Contract Events > Define Independent Condition.** The Independent Condition window appears.
2. Enter a name and a description.
3. Select the **Create a Task** check box if you want to create a task for the Schedule tab in the contract overview. You have access to the tasks through the CRM Task Manager as well. Creating a task requires entering a task owner.
4. Select the condition type. For Action you must select an action from the Action Based Conditions pop-up window. For Date you must enter the number of days, qualify before or after, and the triggering date.
5. For condition of type action choose if the event is evaluated only once or each time the condition is met. For example: Select the Evaluate Once Only check box if you want to evaluate the condition to true only if the counter exceeds 1,000 for the first time. Do not select the check box, if you want the condition to evaluate to true each time the counter reaches 1,000.
6. Enter information to create condition lines.
7. Display the condition lines and check validity.

   Select the **Show Condition** button to display all condition lines and check their syntactical validity on-line. The Valid field is display only and gives you a quick overview if the condition has validated successfully.
8. Select Outcomes previously defined in the Process Definitions Window.
9. Select the **Parameter** button to assign fixed values or action attribute values to the outcome parameters.

10. Save your work.

References

Defining Condition Templates

Reviewing Errors from Asynchronous Processes

This window displays all error messages of Advanced Queueing processes that have failed. Usually this window is used by your support representative to resolve issues.