ORACLE
PeopleSoft

PeopleSoft 9.2: Active Analytics Framework

February 2021

ORACLE

PeopleSoft 9.2: Active Analytics Framework

Documentation Accessibility

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

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Contents

Preface

Understanding the PeopleSoft Online Help and PeopleBooks

The PeopleSoft Online Help is a website that enables you to view all help content for PeopleSoft applications and PeopleTools. The help provides standard navigation and full-text searching, as well as context-sensitive online help for PeopleSoft users.

Hosted PeopleSoft Online Help

You can access the hosted PeopleSoft Online Help on the Oracle Help Center. The hosted PeopleSoft Online Help is updated on a regular schedule, ensuring that you have access to the most current documentation. This reduces the need to view separate documentation posts for application maintenance on My Oracle Support. The hosted PeopleSoft Online Help is available in English only.

To configure the context-sensitive help for your PeopleSoft applications to use the Oracle Help Center, see Configuring Context-Sensitive Help Using the Hosted Online Help Website.

Locally Installed Help

If you're setting up an on-premise PeopleSoft environment, and your organization has firewall restrictions that prevent you from using the hosted PeopleSoft Online Help, you can install the online help locally. See Configuring Context-Sensitive Help Using a Locally Installed Online Help Website.

Downloadable PeopleBook PDF Files

You can access downloadable PDF versions of the help content in the traditional PeopleBook format on the Oracle Help Center. The content in the PeopleBook PDFs is the same as the content in the PeopleSoft Online Help, but it has a different structure and it does not include the interactive navigation features that are available in the online help.

Common Help Documentation

Common help documentation contains information that applies to multiple applications. The two main types of common help are:

- Application Fundamentals
- Using PeopleSoft Applications

Most product families provide a set of application fundamentals help topics that discuss essential information about the setup and design of your system. This information applies to many or all applications in the PeopleSoft product family. Whether you are implementing a single application, some combination of applications within the product family, or the entire product family, you should be familiar with the contents of the appropriate application fundamentals help. They provide the starting points for fundamental implementation tasks.

In addition, the *PeopleTools: Applications User's Guide* introduces you to the various elements of the PeopleSoft Pure Internet Architecture. It also explains how to use the navigational hierarchy, components, and pages to perform basic functions as you navigate through the system. While your application or implementation may differ, the topics in this user's guide provide general information about using PeopleSoft applications.

Field and Control Definitions

PeopleSoft documentation includes definitions for most fields and controls that appear on application pages. These definitions describe how to use a field or control, where populated values come from, the effects of selecting certain values, and so on. If a field or control is not defined, then it either requires no additional explanation or is documented in a common elements section earlier in the documentation. For example, the Date field rarely requires additional explanation and may not be defined in the documentation for some pages.

Typographical Conventions

The following table describes the typographical conventions that are used in the online help.

Typographical Convention	***Description***
Key+Key	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For Alt+W, hold down the Alt key while you press the W key.
. . . (ellipses)	Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax.
{ } (curly braces)	Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe (|).
[ ] (square brackets)	Indicate optional items in PeopleCode syntax.
& (ampersand)	When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object. Ampersands also precede all PeopleCode variables.
⇒	This continuation character has been inserted at the end of a line of code that has been wrapped at the page margin. The code should be viewed or entered as a single, continuous line of code without the continuation character.

ISO Country and Currency Codes

PeopleSoft Online Help topics use International Organization for Standardization (ISO) country and currency codes to identify country-specific information and monetary amounts.

ISO country codes may appear as country identifiers, and ISO currency codes may appear as currency identifiers in your PeopleSoft documentation. Reference to an ISO country code in your documentation

does not imply that your application includes every ISO country code. The following example is a country-specific heading: "(FRA) Hiring an Employee."

The PeopleSoft Currency Code table (CURRENCY_CD_TBL) contains sample currency code data. The Currency Code table is based on ISO Standard 4217, "Codes for the representation of currencies," and also relies on ISO country codes in the Country table (COUNTRY_TBL). The navigation to the pages where you maintain currency code and country information depends on which PeopleSoft applications you are using. To access the pages for maintaining the Currency Code and Country tables, consult the online help for your applications for more information.

Region and Industry Identifiers

Information that applies only to a specific region or industry is preceded by a standard identifier in parentheses. This identifier typically appears at the beginning of a section heading, but it may also appear at the beginning of a note or other text.

Example of a region-specific heading: "(Latin America) Setting Up Depreciation"

Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in the PeopleSoft Online Help:

- Asia Pacific
- Europe
- Latin America
- North America

Industry Identifiers

Industries are identified by the industry name or by an abbreviation for that industry. The following industry identifiers may appear in the PeopleSoft Online Help:

- USF (U.S. Federal)
- E&G (Education and Government)

Translations and Embedded Help

PeopleSoft 9.2 software applications include translated embedded help. With the 9.2 release, PeopleSoft aligns with the other Oracle applications by focusing our translation efforts on embedded help. We are not planning to translate our traditional online help and PeopleBooks documentation. Instead we offer very direct translated help at crucial spots within our application through our embedded help widgets. Additionally, we have a one-to-one mapping of application and help translations, meaning that the software and embedded help translation footprint is identical—something we were never able to accomplish in the past.

Using and Managing the PeopleSoft Online Help

Select About This Help in the left navigation panel on any page in the PeopleSoft Online Help to see information on the following topics:

- Using the PeopleSoft Online Help
- Managing Hosted online help
- Managing locally installed PeopleSoft Online Help

PeopleSoft Enterprise Components Related Links

PeopleSoft Information Portal

My Oracle Support

PeopleSoft Training from Oracle University

Contact Us

Send your suggestions to psoft-infodev_us@oracle.com. Please include the applications update image or PeopleTools release that you're using.

Follow Us

YouTube

Twitter@PeopleSoft_Info.

PeopleSoft Blogs

LinkedIn

Chapter 1

Understanding PeopleSoft Active Analytics Framework

Understanding PeopleSoft Active Analytics Framework

PeopleSoft Active Analytics Framework is a suite of tools that make up a closed-loop decision-making system in which business-intelligent applications or transactions can respond when conditions are met and specific actions are recommended, for example:

- Giving a priority service or a better discount for high-value customers.
- Sending pertinent email messages or notifications.
- Displaying alerts and warning messages.

PeopleSoft Active Analytics Framework provides components for setting up the analytic framework that enable you to manage the data library, build policies, and manage trigger points and actions. These components provide a way to define flexible business rules, called policies, that can be altered without modifying application code. Business analysts and other functional users define policies with an intuitive user interface.

Functional users can create policies that use data elements of various forms and shapes residing in different sources such as the transactional environment, data warehouses, legacy systems, and so on. The data elements are exposed to the business user as terms defined in the PeopleSoft Active Analytics Framework data library.

The extensible action framework supports the definition and execution of consequential actions. Application developers can create customized action types within product lines to accommodate their functional needs. PeopleSoft Active Analytics Framework also delivers a built-in action type for displaying alerts to the user.

Understanding Policies and Trigger Points

Policies are the result of combining trigger points, conditions, and defined actions to complete a desired business request. The framework includes components for building policies.

Application developers and functional business analysts use a wizard-like interface to build, manage, and associate trigger points with policies. Business analysts can create interactive policies that react to customer behavior of particular interest to them. For example:

- If a banking customer deposits more than ten thousand dollars, a banking analyst can create a regulatory IRS notification.

- If a high-value customer has logged three or more critical support issues with a call center within a week, a business executive could send a letter of apology.

Policies supplement, but do not alter, normal transactional processing. Therefore, a policy cannot be used to stop a particular behavior due to some specified restriction. If the condition portion of a policy evaluates to true, the policy causes an action to be performed. If the specified condition evaluates to false, then no consequential actions occur. Policies are evaluated independently from each other. Therefore, if more than one policy is to be evaluated at the same time, the consequential actions of one policy cannot alter or influence the actions of another. Likewise, the sequence of policy execution cannot affect the results of another policy.

You construct a policy by defining one or more conditions, specifying one or more actions, and associating them with a trigger point. A policy cannot be activated without defining at least one condition and action. You can reuse defined policies with multiple trigger points if the elements of the policy agree with the contexts of the trigger points.

Trigger points are events from which the analytic decision engine is invoked by the application. The framework supports the registration of new trigger points, when needed. Examples of trigger points are:

- When a customer is identified.
- When a product is selected.
- After you sign in to a self-service application.

Registration of a trigger point involves specifying:

- The type of actions to be invoked.
- The *context* in which the associated policies are to be carried out.

During runtime, policies are triggered by specific trigger points within application components, resulting in defined actions being taken.

Note: Registration of a trigger point also involves introducing necessary code in the application to request the decision engine to evaluate the policies pertaining to a trigger point.

Understanding the Data Library

The data library is a repository for information within the PeopleSoft Active Analytics Framework. Each element in the data library is exposed by way of a *term,* which is a pointer to a unit of data within the PeopleSoft system. This data may reside in a relational database, or it may be derived at runtime.

Terms in the data library are organized hierarchically into functional categories called *subject areas.* Terms can be assigned to more than one subject area at a time; for example, if a term represents a customer it could be located both in a marketing subject area and in a financial subject area.

The data library enables functional users to:

- Access data (terms) residing in different sources such as an operational CRM environment, data warehouses, legacy systems, and so on.

- Use the data in variety of contexts, such as input in rule applications, tokens in correspondence templates, or placeholders for customized text in questions or for presenting disparate pieces of information in different screen applications.

PeopleSoft Active Analytics Framework includes components to define new terms in the data library and can automatically create terms for data elements in a component.

Understanding the Action Framework

The action framework is a suite of components that are designed to manage actions and to invoke actions at runtime. The primary purpose of the action framework is to enable functional users to specify and configure the actions to be performed when policy conditions evaluate to true.

Also, Oracle designed the action framework to enable application developers and IT personnel to introduce new action types for use in policies and to invoke them at runtime.

The action framework components:

- Register and maintain *action types.* An action type is metadata pertaining to a class of actions that can be performed at runtime.
- Register and maintain *action bundles.* Action bundles are groups of combinable action types.
- Embed and configure consequential actions within policies.
- Provide a display alert action type for use with all product lines.

Understanding Application Data Sets

The Application Data Set functionality includes the Data Set Designer component (PTADSMMGR) and the Data Migration Workbench component (PTADSDMW):

1. Data Set Designer - Authorized administrators use the Data Set Designer to create data set definitions (ADS definition) as a hierarchical structure of records and their collective properties. A data set definition, with its group of records, constitutes a data set. Both record definitions and data set definitions are metadata that define the shape of the migration data.
2. Data Migration Workbench - Authorized administrators can then use the Data Migration Workbench to insert data set instances (data content) into projects that represent a unit of work as a data migration project. Data migration projects are like managed object projects: a collection of data set instances with various data set definitions. The Data Migration Workbench enables administrators to copy and compare projects containing data sets as well as view compare reports and validation reports.

   You can also integrate the Enterprise Components Approval Framework to provide administrative control of the Project Copy from File process. Employ enhanced security to ensure that the Data Set definitions are suitable for copying data, to enable user security for the PIA data set pages, and assign access to copy and compare the data. PeopleSoft delivers the MigrateData process ID for enabling data migration Approval Framework.

The following Data Sets are available in Enterprise Components:

Data Set Definition Name	*Description*
EOCF_ACTION_BUNDLE	EOCF Action Bundle
EOCF_ACTION_DEFN	EOCF Action Definition
EOCF_ACTION_EVTYPE	EOCF Action Event Type
EOCF_ACTION_OBJECT	EOCF Action Object
EOCF_ACTION_SET	EOCF Action Set
EOCF_ACTION_TYPE	EOCF Action Type
EOCF_BUNDLE	EOCF Bundle
EOCF_BUS_DOMAIN	EOCF BUS DOMAIN
EOCF_CONTEXT	EOCF Context
EOCF_ELEM_SUB	EOCF Subject
EOCF_EVENT_DEFN	EOCF Event Definition
EOCF_EVENT_RULE	EOCF Event Rule
EOCF_EVENT_TYPE	EOCF Event Type
EOCF_IMPL_DEFINITION	EOCF Implementation Definition
EOCF_INTEL_ACTION	EOCF INTEL ACTION
EOCF_NATIVE_POLICY	EOCF Native Policy
EOCF_OPER_DEFINITION	EOCF Operator Definition
EOCF_OPER_SET	EOCF Operator Set
EOCF_PHRASE_POLICY	EOCF Phrase Policy
EOCF_POLICY_DEFINITION	EOCF Policy Definition
EOCF_RES_AUDIENCE	EOCF Res Audience
EOCF_SUBJECT	EOCF Subject
EOCF_SUBJ_TREE	EOCF Subject Tree
EOCF_TERM	EOCF Term
EOCF_TERM_CFG	EOCF Term Config

Chapter 2

Building and Managing Policies

Building Policies

You build and manage policies with a wizard-like interface called a policy builder. During the creation of policies, you associate them with trigger points. At runtime, policies are evaluated by specific trigger points in application components, resulting in defined actions being taken.

The policy builder enables business analysts to change conditions, actions, or both in policies to enable a business process change in an application component without having to modify application code or needing the help of IT personnel.

Note: Policies cannot be shared among different setIDs.

Pages Used to Build, Edit, and Activate Policies

Page Name	*Definition Name*	*Usage*
Manage Policies - Search Page	EOCF_RULE_CFGSRCH	Build and manage policies.
Manage Trigger Point Page	EOCF_MANAGE_TP	Manage trigger points.

Prerequisites to Building Policies

Before you build a policy:

- Define trigger points.
- Define data library terms.
- Define action types, categories, and action objectives.

See Registering Trigger Types and Trigger Points, Understanding the Data Library, Understanding the Action Framework.

Manage Policies - Search Page

Use the Manage Policies - Search page (EOCF_RULE_CFGSRCH) to build and manage policies.

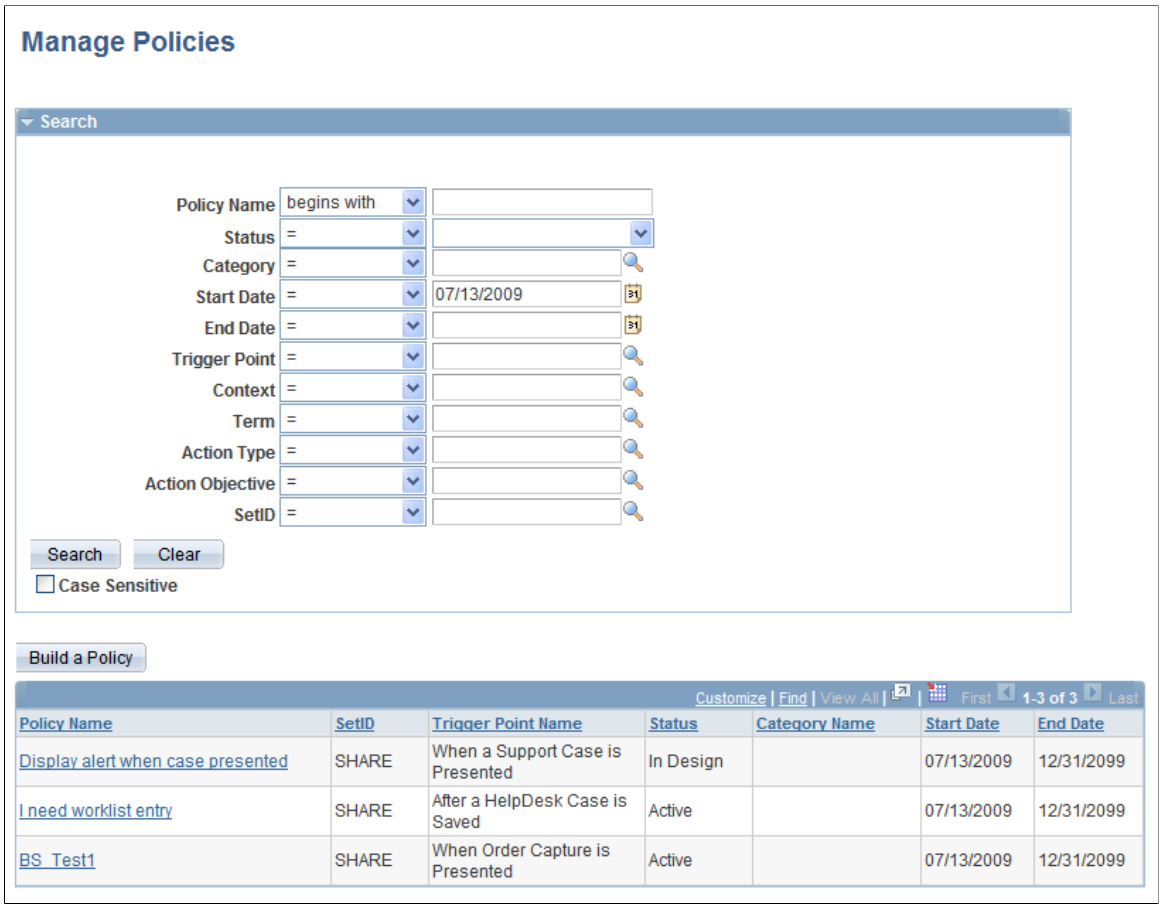
Navigation

Enterprise Components > Active Analytics Framework > Policies > Manage Policies

Use the Manage Policy (EOCF_RULE_CFGSRCH) page to build a new policy or edit an existing policy.

Image: Manage Policies - Search page

This example illustrates the fields and controls on the Manage Policies - Search page. You can find definitions for the fields and controls later on this page.



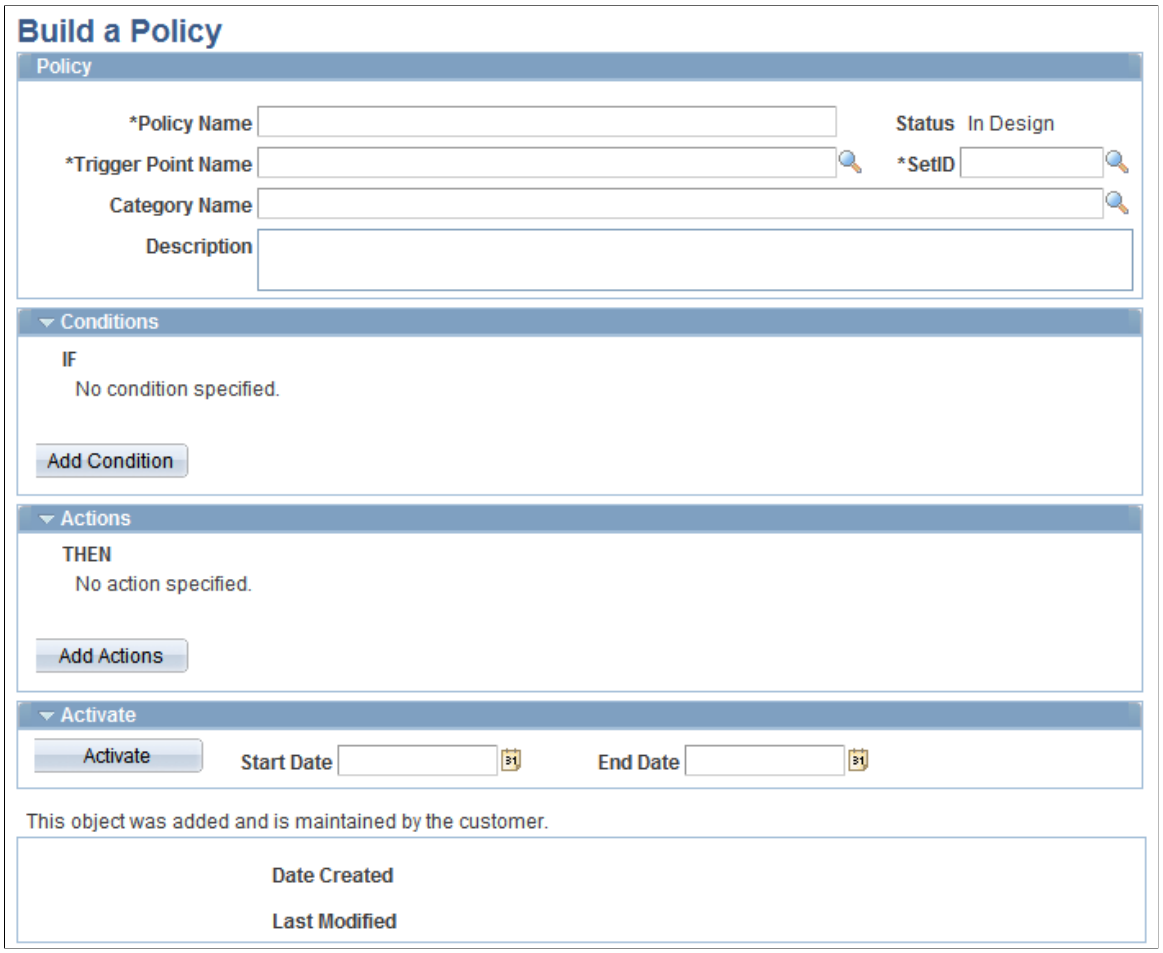
If you want to edit an existing policy, use the search criteria to find the desired policy, then click the policy name on the results grid to open the policy definition.

Note: If you select a trigger point name as a search criterion, only policies directly associated with the trigger point are retrieved. To retrieve policies associated with *policy groups* of a trigger point, specify search criteria other than the trigger point name.

To create a new policy, click the Build a Policy button to access the Build a Policy (EOCF_RULE_DEFN) page.

Image: Build a Policy page

This example illustrates the fields and controls on the Build a Policy page. You can find definitions for the fields and controls later on this page.



Policy Name Enter a unique and descriptive name for the policy.

Trigger Point Name Select a trigger point from the drop-down list box.

A policy is always associated with at least one trigger point.

Category Name Select a policy category name from the drop-down list box.

This name is used to functionally classify policies and aid in searching for policies.

SetID Select from a list of set IDs that are defined within the PeopleSoft system.

Specify a setID and a trigger point for every policy. This value is used to select the valid set of policies associated with a trigger point at runtime. This value also constrains the prompt list for the right-hand side values entered in conditions by performing a setID to setID indirection.

Status The default value for a new policy is *In Design.*

	When you click the Activate button and activate the policy, the status changes to *Active*.
Start Date and **End Date**	Enter the start and end dates. These dates define the validity of a policy at runtime.

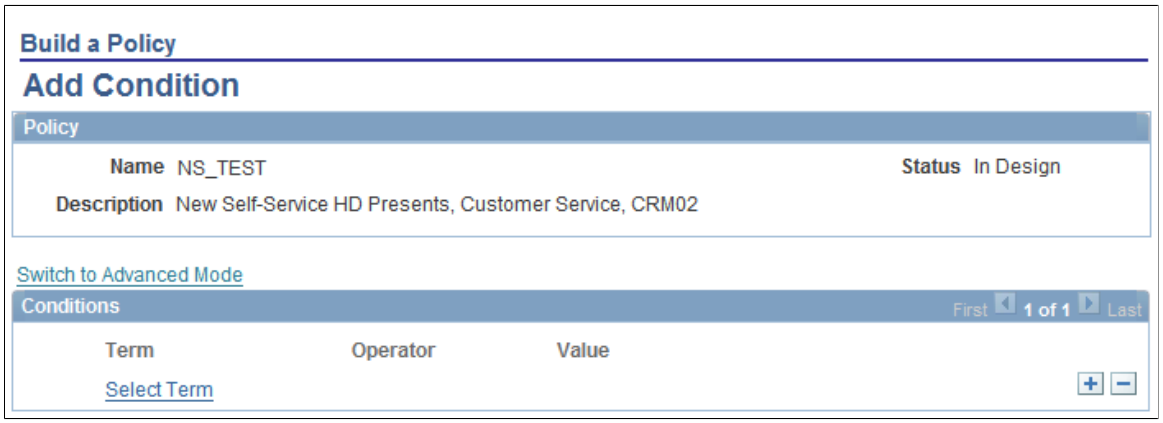
Complete the appropriate fields and click Add Conditions.

Adding and Editing Conditions

Click the Add Condition button on the Build a Policy page to access the Add Condition page.

Image: Add Condition page

This example illustrates the fields and controls on the Add Condition page. You can find definitions for the fields and controls later on this page.



Two modes are available for specifying conditions:

- Basic.

  This is the default mode; the default logical operator is AND.

- Advanced.

  You can group condition rows using parentheses, specify logical operators (AND, OR), and specify terms as values in the right-hand side.

To add a condition row:

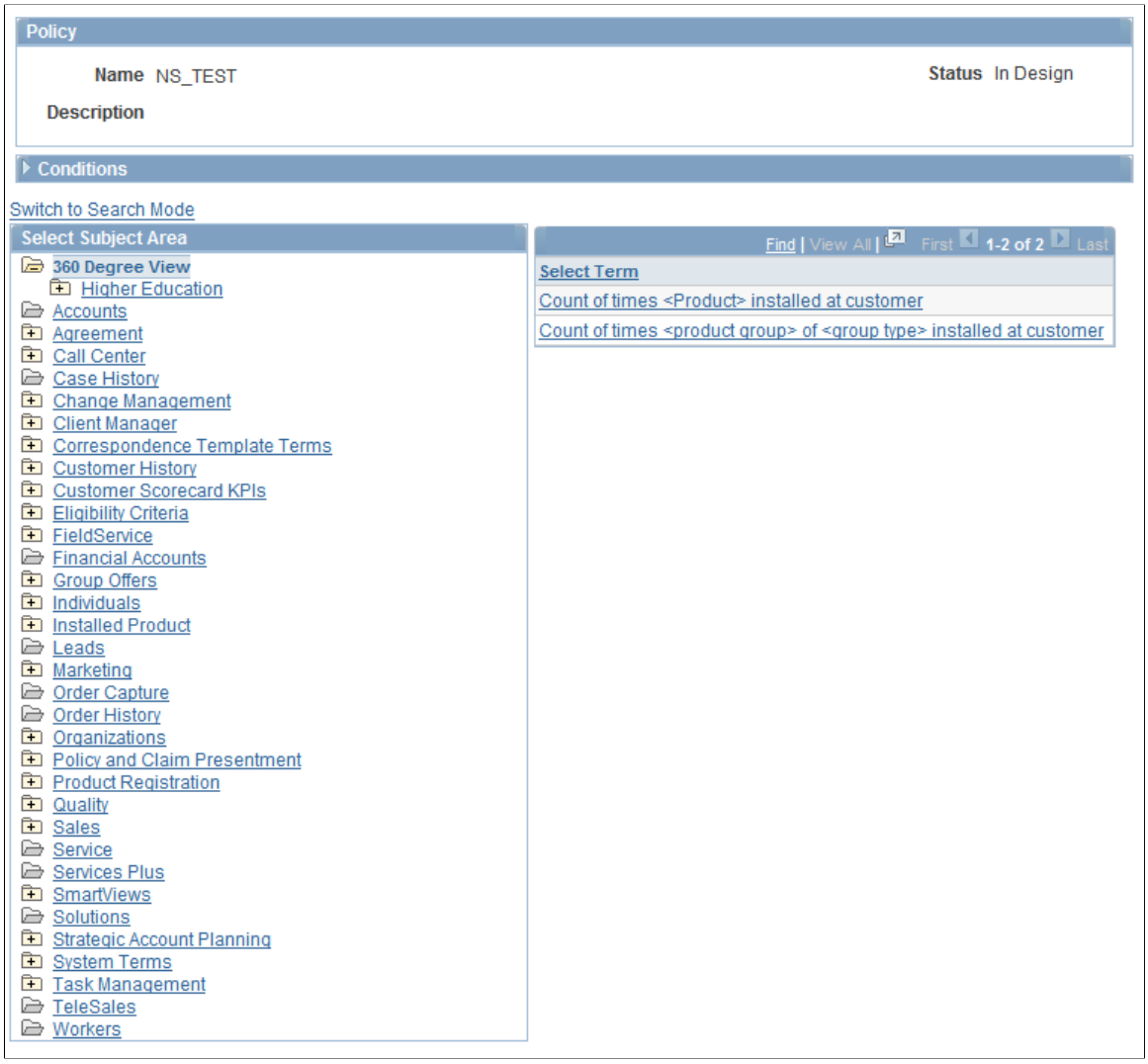
1. Select a term. Configure the term if it is linked.
2. Select an operator.
3. Enter or select one or more values on the right-hand side.

Selecting Available Terms

Click Select Term on the Add Condition page to access the Term Selection page.

Image: Term Selection page

This example illustrates the fields and controls on the Term Selection page. You can find definitions for the fields and controls later on this page.



The Term Selection page has two modes by which to search for and select terms for a condition: browsing by subject area and searching by entering search criteria. Terms that appear are limited to those that can be resolved by the trigger point. Therefore, all terms having contextual implementations for the context pertaining to this trigger point and all terms having generic implementations in which binds can be supplied by the context are available.

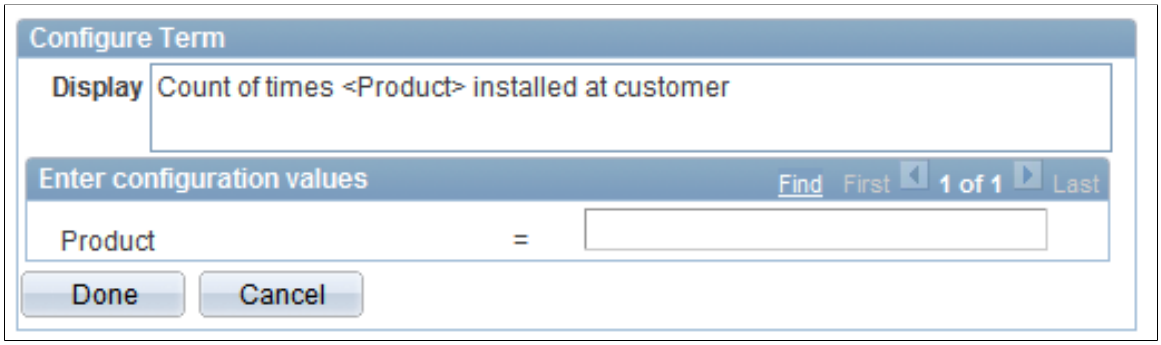
Terms returning multiple rows are not available for use in conditions. Terms that retrieve data from detail rows in the component buffer cause the decision engine to evaluate the condition once for each row. The decision engine generates actions for every row for which the condition is true.

Configuring a Term

If you select a term that is configurable— it has design time binds—it must be configured when you build the condition. Configurable terms are displayed as links. Subsequently, while building the condition, click the link of the term to access the Configure Term page and configure the term.

Image: Configure Term section

This example illustrates the fields and controls on the Configure Term section. You can find definitions for the fields and controls later on this page.



The prompt of valid values displayed for configurable terms relies on the prompt configuration specified in the Manage Terms component.

See Creating Terms.

Selecting an Operator

The list of operators available when you select a term depends on the return data type defined for that term and the context of the selected trigger point.

Each operator defined in the Register Operators page supports certain data types; this determines which operators are available when you select a term in the condition builder. Furthermore, the selection of an operator determines how many fields are required on the right-hand side for entering values.

For example, if you select the is between operator, two right-hand side fields appear to enter values.

See Registering Operators and Operator Sets.

Entering Right-Hand Side Values of a Condition

Prompt options that you specify when defining a term determine the right-hand side field type. The field may be a prompt, drop-down list box, or multiselect prompt. SetID-based prompt tables specified in the term definition use the setID value specified on the policy definition page to perform a setID to setID indirection, thus constraining the prompt list.

Note: Use a semicolon as a separator in multiselect prompts.

In advanced mode, you can specify a term on the right-hand side of a condition—this term is resolved at runtime and the resolved value used as the right-hand side value.

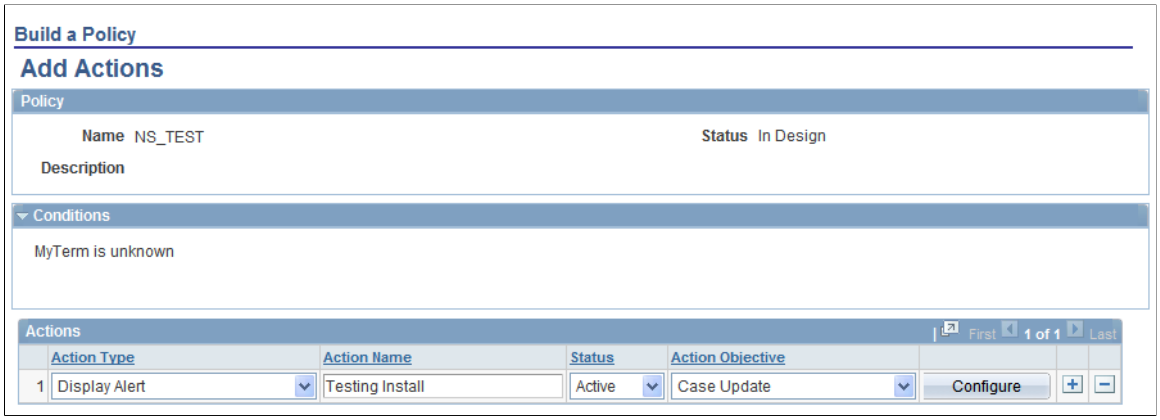
After you enter the condition, click Done to save the condition. Before a successful save, the system validates that right-hand side values are present and are of the appropriate data types; that parentheses match; and that configurable terms have been configured.

Adding, Editing, and Configuring Actions

Click Add Actions or Edit Actions on the Build a Policy page to access the Add or Edit Actions page.

Image: Add Actions page

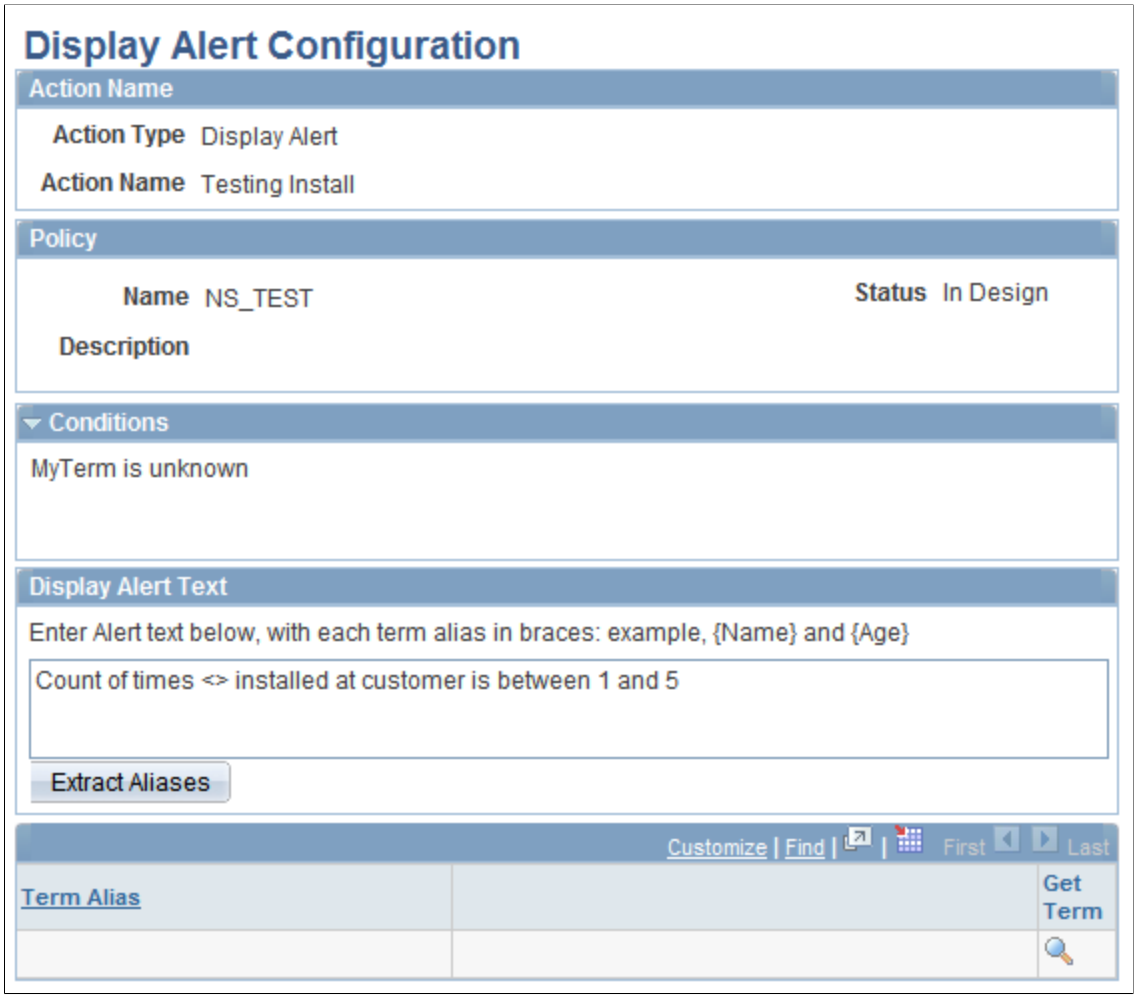
This example illustrates the fields and controls on the Add Actions page. You can find definitions for the fields and controls later on this page.



If the action type is configurable, the Configure button is enabled on the page. Individual action types have specific configuration pages.

Image: Display Alert Configuration page

This example illustrates the fields and controls on the Display Alert Configuration page. You can find definitions for the fields and controls later on this page.



Note: PeopleSoft Active Analytics Framework delivers a display alert action type. Other action types may be delivered by individual product lines. Please refer to the appropriate product line documentation to get more information about delivered action types.

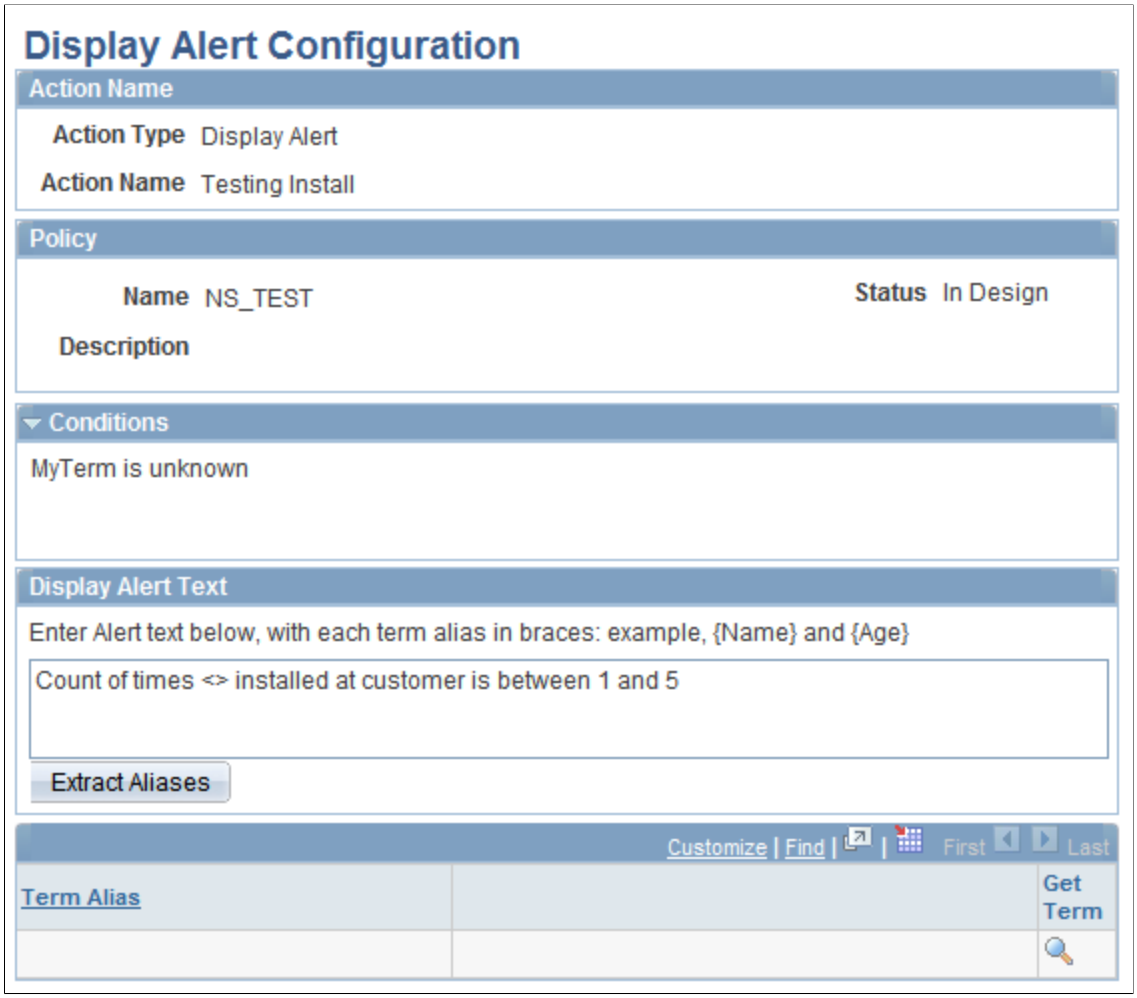
See Registering Action Types and Action Type Bundles.

Configuring the Display Alert Action

Access the Display Alert page by selecting *Display Alert* action type.

Image: Display Alert Configuration page

This example illustrates the fields and controls on the Display Alert Configuration page. You can find definitions for the fields and controls later on this page.



1. Click Configure.

   The Display Alert Configuration page appears.

2. Enter text in the Display Alert Text field and include any term aliases in braces.

   Term aliases are placeholders for dynamic content to be merged in the alert text at runtime.

3. Click Extract Aliases to extract term aliases to populate the grid, thus enabling you to map each alias to a term.

4. Click Get Term to map a term for each term alias in the grid.

   Only terms that return a single value can be used within the display text. Return data types of record or rowset, or terms with *Many* rows specified are not allowed.

5. Click OK or Apply to save the display alert configuration.

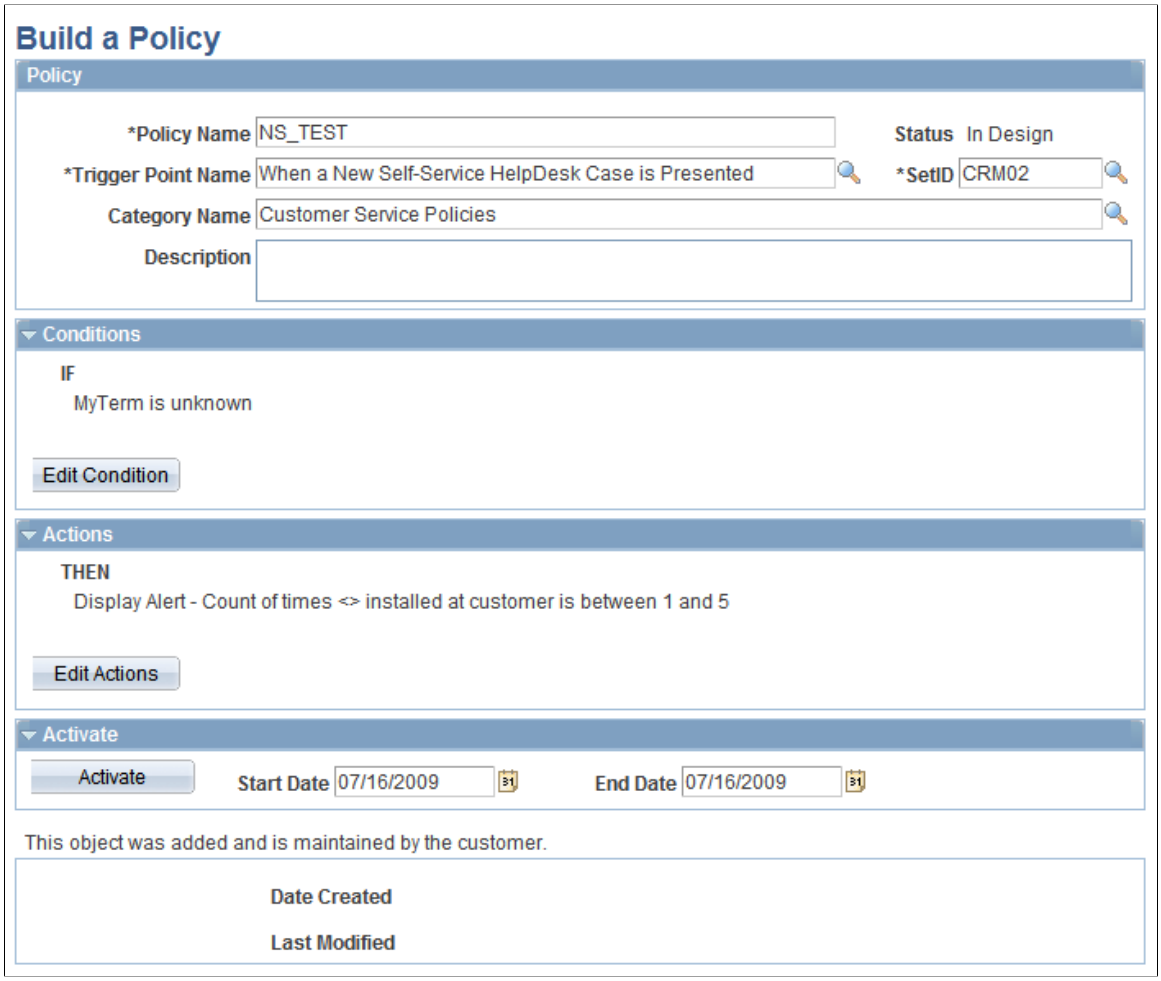
   This configuration is retrieved at runtime to generate the alert text and display it in a popup box.

Activating a Policy

Access the Build a Policy page (Enterprise Components, Active Analytics Framework, Policies, Manage Policies).

Image: Build a Policy page

This example illustrates the fields and controls on the Build a Policy page. You can find definitions for the fields and controls later on this page.



On the Build a Policy page, click the Activate button.

The system sets the status to active after executing validations. Activating a policy disables field editing; however, the policy can still be copied and associated with another trigger point.

Note: Upon activation of the policy, any modifications made are in effect only in new user sessions. Therefore, you must sign out and sign in again to see any changes made to the policy.

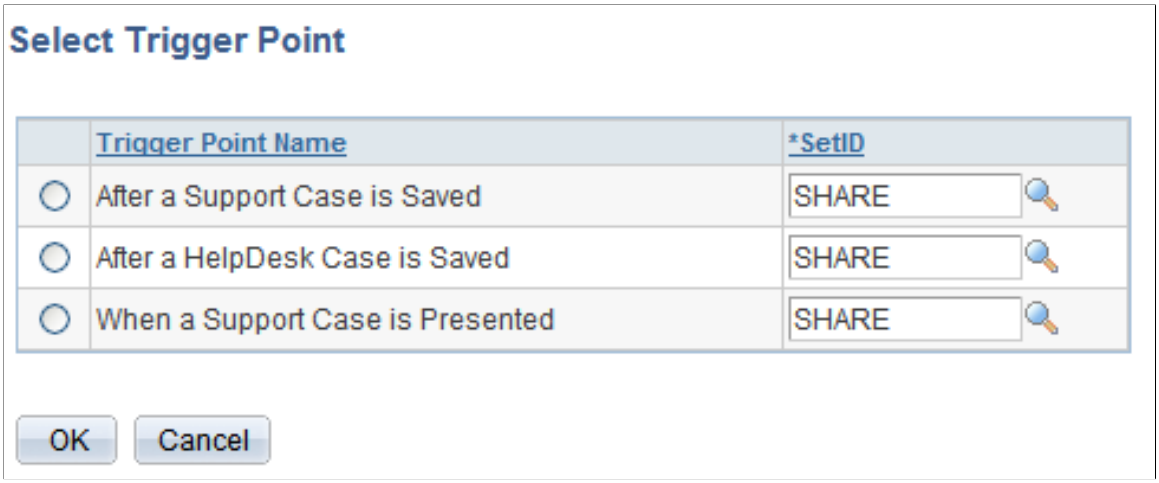
Associating a Policy to Another Trigger Point

A policy can be associated with more than one trigger point within the same setID. Do this by:

1. Adding a new row to the Associated Trigger Points grid.
2. Selecting a valid trigger point from the drop-down list box:

Image: Select Trigger Point page

This example illustrates the fields and controls on the Select Trigger Point page. You can find definitions for the fields and controls later on this page.



Note: The trigger points available for selection in the drop-down list box are constrained by the terms used in the policy condition and the actions configured in the policy. Also, a policy cannot be associated with multiple trigger points spanning multiple setIDs.

3. Saving the policy to associate the policy with the new trigger point.

Copying a Policy

Create a new policy by copying an existing one, provided that you can use the same condition and actions. While copying a policy (by clicking the Copy button at the bottom of the Build a Policy page), you'll be prompted for a trigger point and setID. Selecting from the list of valid trigger points results in creating a new policy, which appears on the screen.

Image: Build a Policy page showing Copy button clicked to copy that policy (1 of 2)

This example illustrates the fields and controls on the Build a Policy page showing Copy button clicked to copy that policy (1 of 2). You can find definitions for the fields and controls later on this page.

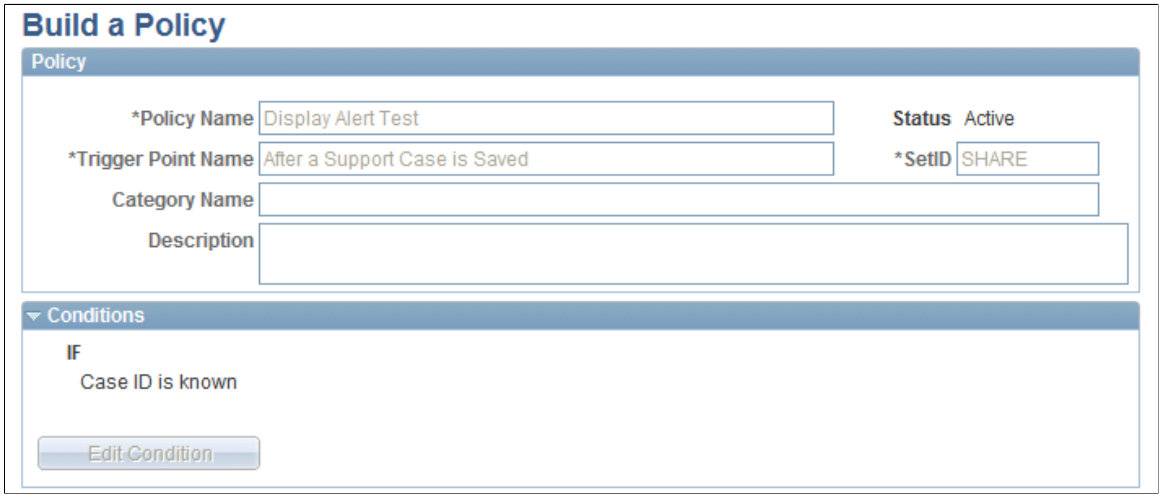
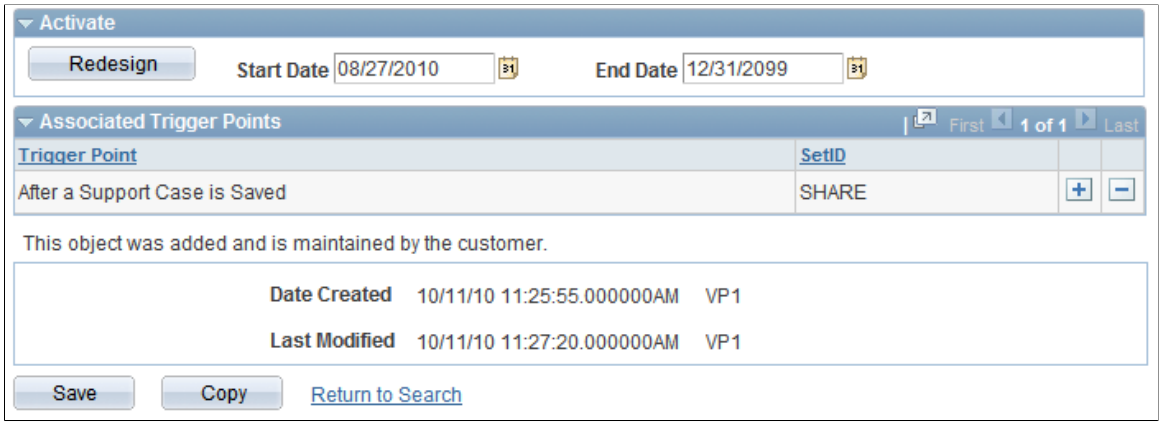


Image: Build a Policy page showing Copy button clicked to copy that policy (2 of 2)

This example illustrates the fields and controls on the Build a Policy page showing Copy button clicked to copy that policy (2 of 2). You can find definitions for the fields and controls later on this page.



Note: When you copy a policy, the condition and actions, but not the action configurations, are copied to the new policy. Therefore, you need to reconfigure the actions by clicking Edit Actions. A reminder message appears when you're transferred to the new policy.

Managing Trigger Points

The Manage Trigger Point page provides a comprehensive view of policies that are associated with a specific trigger point. This page displays policies and policy groups in a hierarchy, with the trigger point as the root and policy groups (if any) as parents of policies.

In addition, this page enables you to assign execution options at the trigger point level and at the policy group level, facilitating policy arbitration and better policy management.

Page Used to Manage Trigger Points

Page Name	*Definition Name*	*Usage*
Manage Trigger Point Page	EOCF_MANAGE_TP	Manage trigger points.

Manage Trigger Point Page

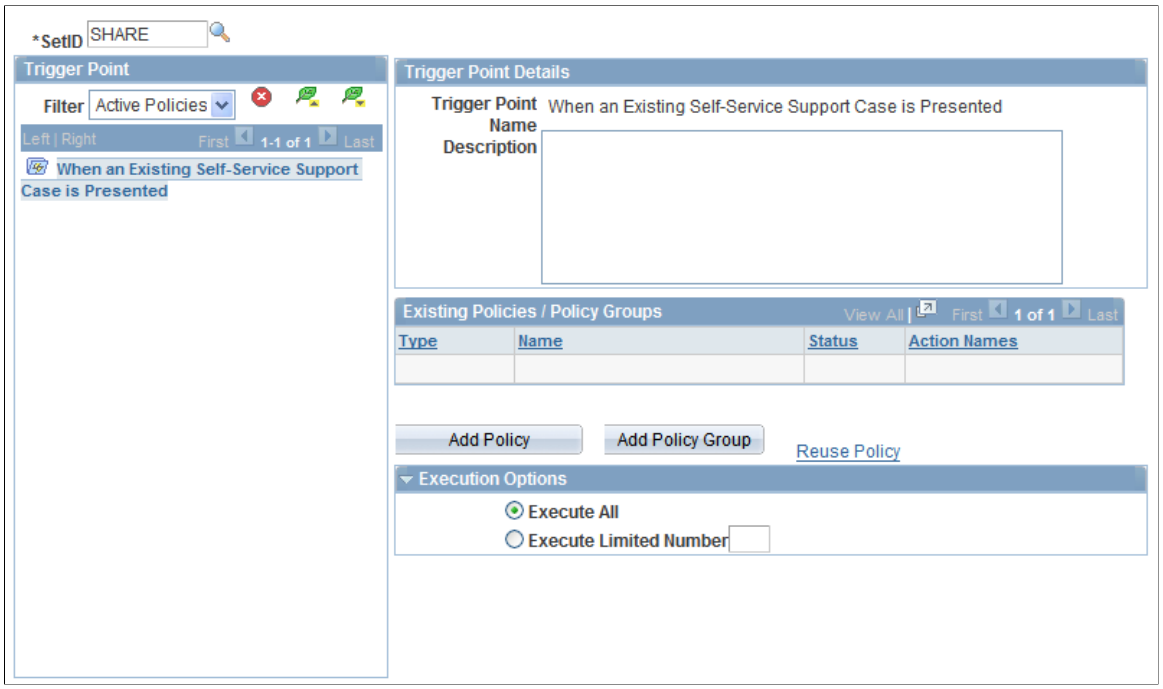
Use the Manage Trigger Point page (EOCF_MANAGE_TP) to manage trigger points.

Navigation

Enterprise Components > Active Analytics Framework > Policies > Manage Trigger Point

Image: Manage Trigger Point page

This example illustrates the fields and controls on the Manage Trigger Point page. You can find definitions for the fields and controls later on this page.



The trigger point hierarchy on the left-hand side of the page displays all the policies and policy groups associated with the selected trigger point. The trigger point appears as the highest-level item in the hierarchy while policy actions appear as the lowest level items.

Filter	This field applies only to policies displayed in the hierarchy and the Existing Policies/ Policy Groups grid. It displays active policies, in-design policies, or all policies depending on the selection.
SetID	Toggle the value in this field to view policies for this setID.

Removing a Policy or Policy Groups

You can remove a policy or policy group from a trigger point by selecting the policy or policy group to be deleted, and then clicking the Delete icon. Removing a policy or policy group from the trigger point disassociates it from the trigger point, but does not delete it from the database.

A policy group is not reusable—once the policy group is disassociated from a trigger point, it can not be referenced.

Any changes made to a trigger point by adding or removing policies or policy groups or by modifying execution options take effect at runtime and only for new user sessions. Therefore, you must sign-out and sign-in again to see any changes made.

Note: A policy that is associated with a single trigger point (either directly or within policy groups) cannot be removed from the trigger point. To disable such a policy, you must edit it and set the status to *In Design*.

Reordering Policy or Policy Groups

To set priorities to policies, you may need to reorder policies or policy groups within a trigger point or a parent policy group. Reorder policies and policy groups by using the Reorder icon.

Adding a Policy or Policy Group

Click Add Policy to create a new policy and associate it with the trigger point. The Build a Policy page appears.

Click Add Policy Group to create a new policy group and to associate it with this trigger point.

A policy group may be used to set policy priority within a group, to deactivate policies, to nest child policy groups, and to assign preconditions.

Reusing Policies

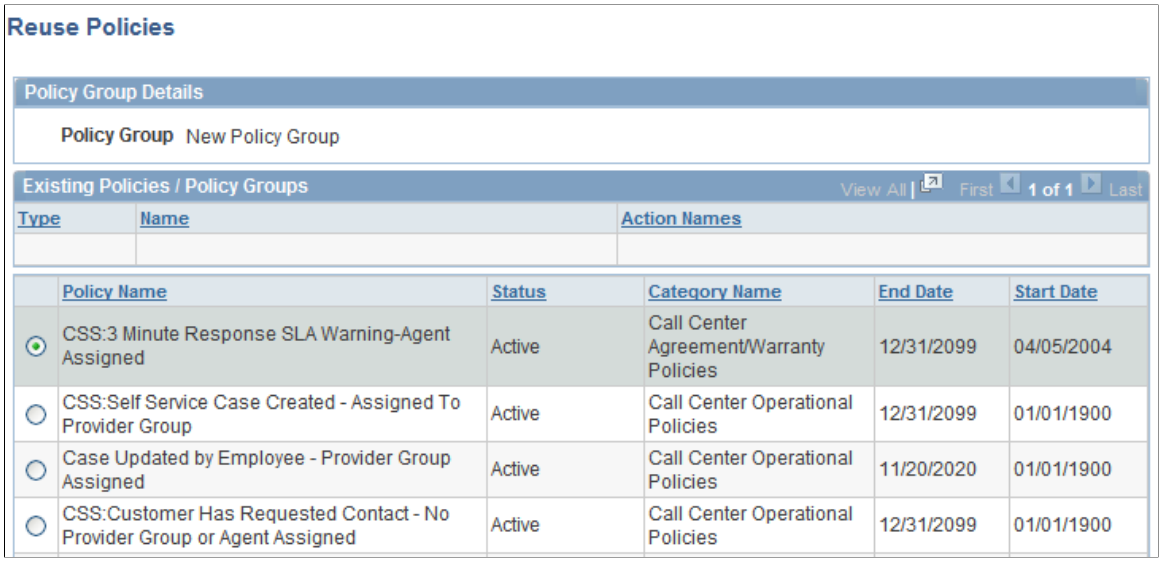
Reuse a policy in multiple trigger points and policy groups if the contexts are compatible.

Click the Reuse Policy link in the Existing Policies/Policy Groups section of the Manage Trigger Points page to reuse an existing policy.

Select one of the policies listed in the grid by selecting the appropriate option. Click OK. The selected policy is associated with the trigger point or policy group.

Image: Reuse Policies page

This example illustrates the fields and controls on the Reuse Policies page. You can find definitions for the fields and controls later on this page.



Note: If a policy is reused within a single trigger point through direct association with the trigger point, or by association with a policy group within the trigger point, you cannot remove this policy from either the trigger point or the policy group within the trigger point. If you want to remove such a policy, you must deactivate the policy by setting its status to *In Design*.

Adding a Precondition

Preconditions are combinations of one or more conditions. They are optional and only policy groups can have preconditions.

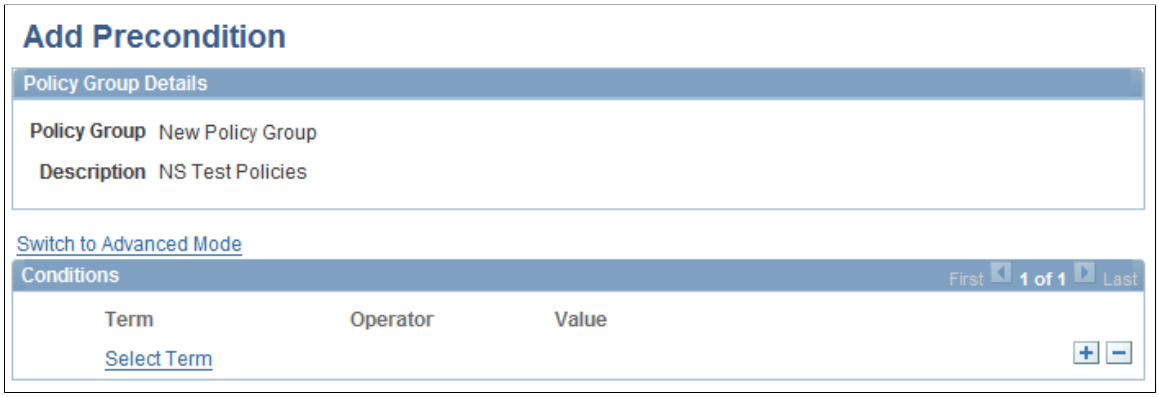
At runtime, the policies within a policy group are not evaluated unless the precondition evaluates to true.

For example, you could have a precondition defined for a self-service policy group, such as “Is this user on the internet?” Consequently, all policies within that policy group would not be executed unless the precondition of being on the internet evaluates to true.

Click Add Precondition to access the Add Precondition page.

Image: Add Precondition page

This example illustrates the fields and controls on the Add Precondition page. You can find definitions for the fields and controls later on this page.



Select terms and operators; specify values on the right-hand side.

Setting Execution Options

The following execution options can be specified for a trigger point or a policy group:

Execute All	This is the default, if specified for a trigger point. This option enables execution of all policies within a trigger point or policy group. During runtime, when executing policies within a policy group, the system adheres to the execution option specified for the policy group.
Execute Limited Number	Enables execution of a maximum (of the number specified) policies that evaluate to true.
Use Parent Execution Options	(Policy groups only). This is the default for policy groups. At runtime, this uses the execution options of either the trigger point or the parent policy group.

Understanding Execution Options

This section describes the execution options and various scenarios of how they can be used.

Execute All

The *Execute All* option tests all policies in a trigger point; that is, all policies in the trigger point can cause actions to occur if their conditions prove true.

For example, if a trigger point has three policies and the execution option is set to Execute All, all three policies are evaluated.

Execute Limited Number

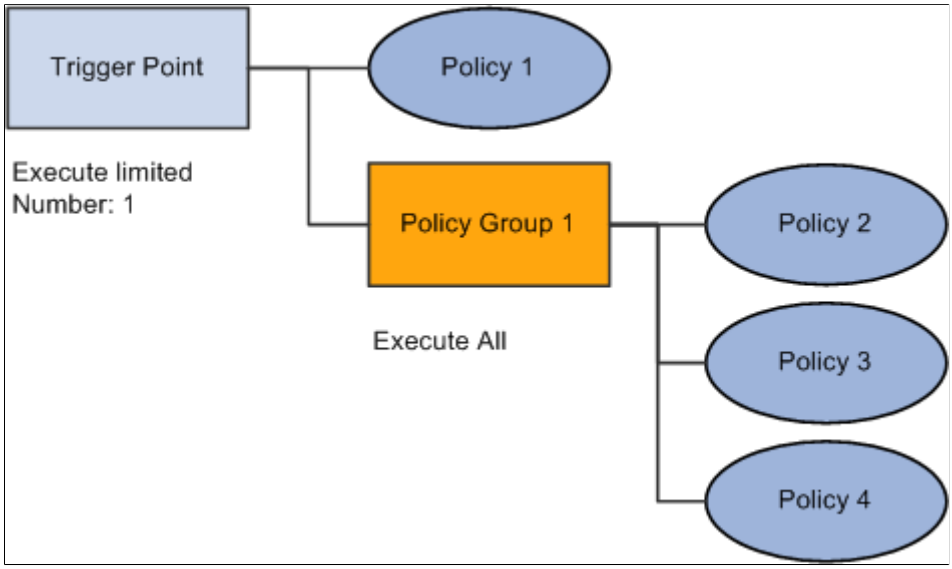
The *Execute Limited Number* option evaluates all policies in the trigger point until one of the policies' conditions evaluates to true. Therefore, suppose that you set Execute Limited Number to 1 for a trigger point that has three policies, where Policy 1 evaluates to false and Policy 2 evaluates to true; Policy 3 will not be considered and Policy 2 is executed.

Various Scenarios of Execution Options

Policy groups may have their own execution options that could affect the option setting. For example, consider the following diagram:

Image: Execution Options: Scenario 1

Diagram showing scenario 1 execution options

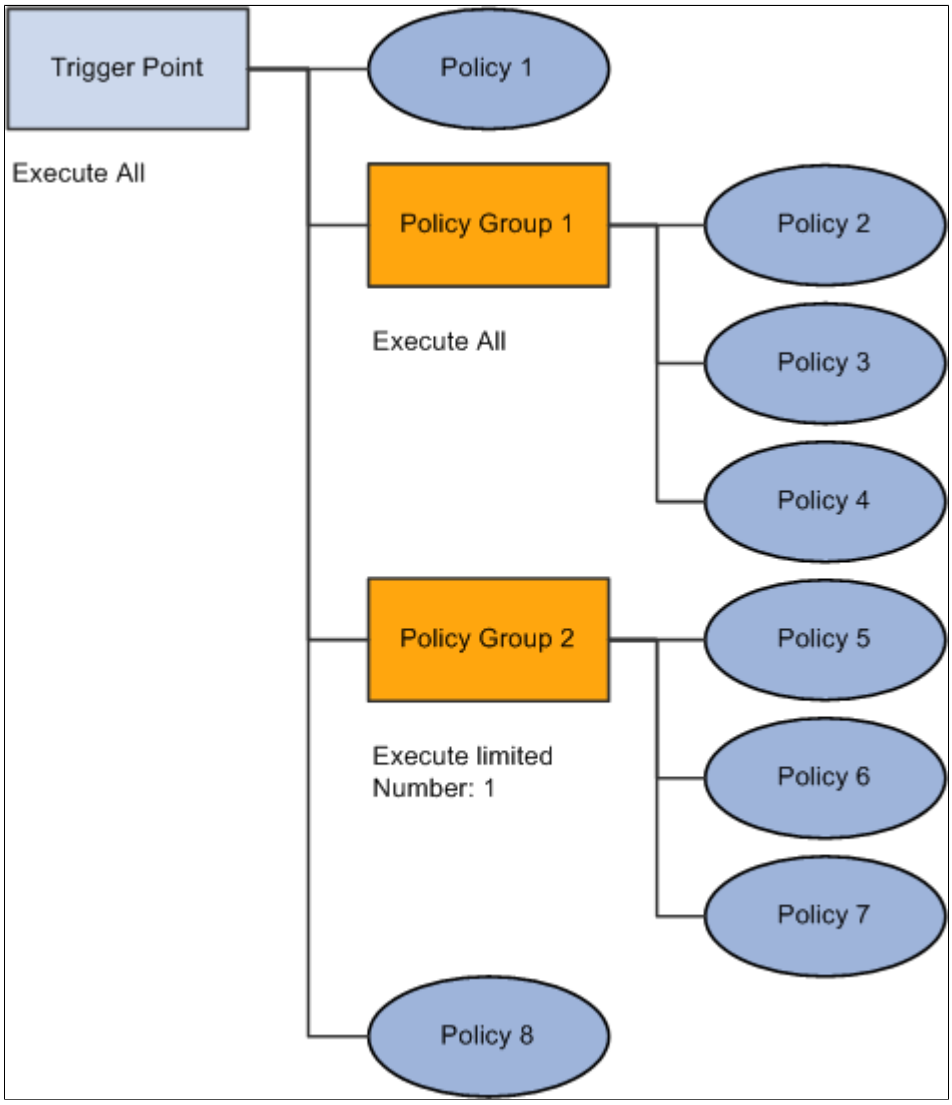


In Scenario 1, if Policy 1 proves false, then all policies in Policy Group 1 are evaluated because its execution option overrides the trigger point's execution option. Therefore, even though the trigger point is set to execute one policy, if Policy 2, 3, and 4 evaluate to true, those three policies' actions will execute.

Consider Scenario 2:

Image: Execution Options: Scenario 2

Diagram showing scenario 2 execution options

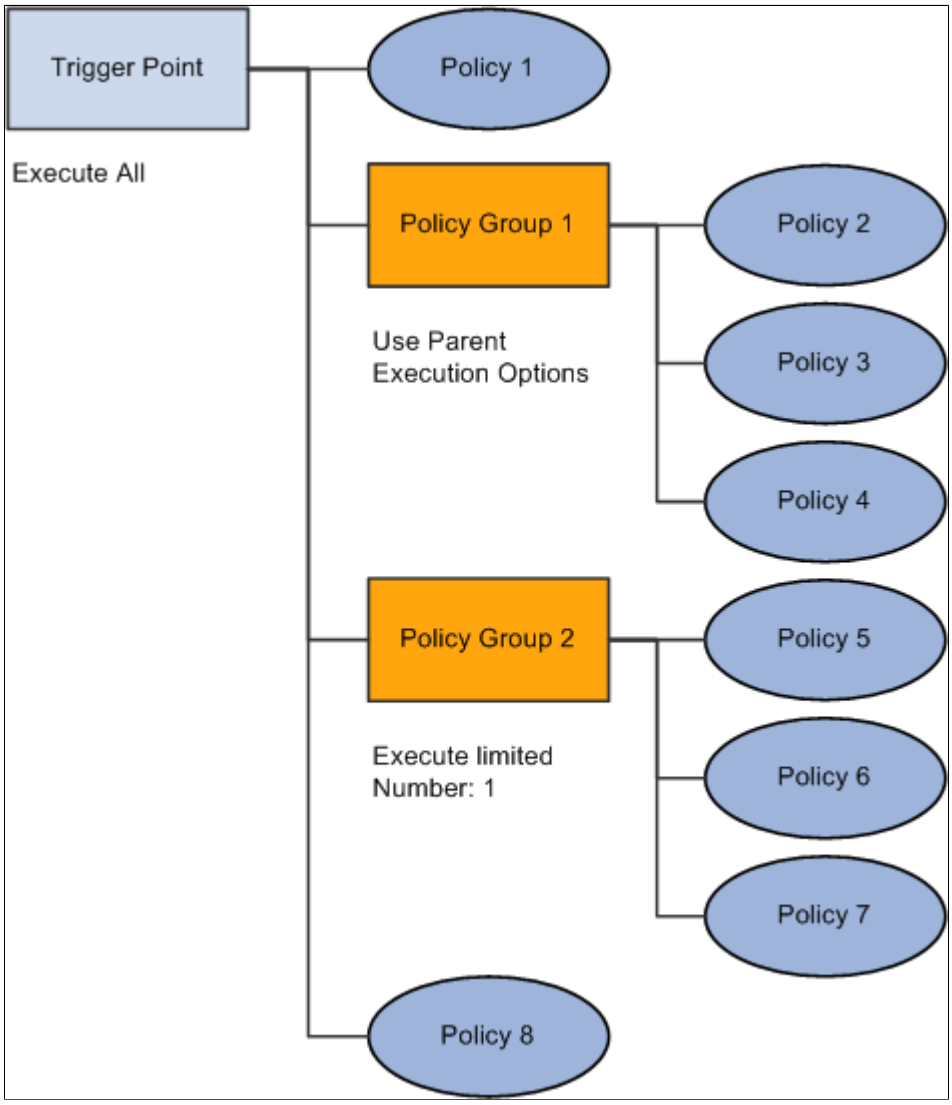


In this scenario, assume that all policy conditions are true; actions from Policy 1, 2, 3, 4, 5, and 8 will execute.

Consider Scenario 3:

Image: Execution Options: Scenario 3

Diagram showing scenario 3 execution options

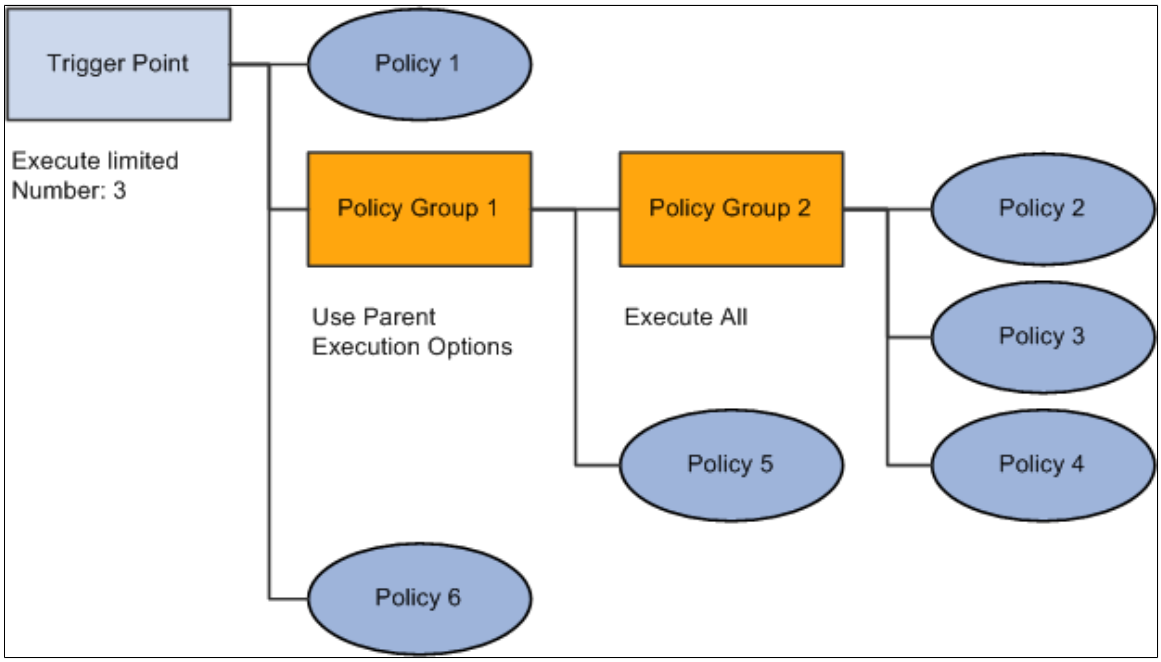


In this scenario, the trigger point's execution option is the default (as is Policy Group 1). Assuming that all policy conditions are true, this trigger point executes exactly as in Scenario 2; that is, actions from Policy 1, 2, 3, 4, 5, and 8 will execute.

Consider Scenario 4:

Image: Execution Options: Scenario 4

Diagram showing scenario 4 execution options



In this scenario, assume that all policy conditions are true. The execution option for Policy Group 2 overrides that for Policy Group 1; therefore, all of the policy actions for Policy Group 2 execute. However, Policy 5 is not considered, nor is Policy 6 because three policy actions have already executed (Policy 2, 3, and 4).

Guidelines for Setting Execution Options

Unless you have a specific reason to set options, Oracle recommends using the default execution options. If only a few conditions could be true for a trigger point, and these conditions are logically exclusive (only one could be true at a time), set the number of policies considered to one (Execute Limited Number =1) to improve performance.

If, for whatever reason, only a few policy options should be considered, and no policy groups exist, setting a limitation for the trigger point is a good practice.

If you have a trigger point containing unrelated policies and a category of possible conditions that may be true, for which only one set of consequent actions should be taken, the best practice is to separate the unrelated policies into a separate policy group with a limitation on the number of policies allowed to fire.

Chapter 3

Setting Up the Data Library

Understanding the Data Library

The data library is a repository for information within the PeopleSoft Active Analytics Framework. Each element in the data library is exposed by way of a *term,* which is a pointer to a unit of data within the PeopleSoft system. This data may reside in a relational database, or it may be derived at runtime.

See Also Understanding PeopleSoft Active Analytics Framework

Creating Implementations

Use the Define Implementations component (EOCF_IMPL_DEFN) to create and define implementations.

An *implementation* refers to the mechanism through which the data is retrieved, derived, or computed. The implementation knows either where the data physically resides or it knows the algorithm for deriving the value. All terms must be associated with an implementation unless the data to which the term refers is present in the component buffer.

An implementation can be associated with more than one term. Conversely, a term may require multiple implementations if it needs to be resolved from multiple contexts. Typically, application developers or IT personnel develop implementations.

Note: Terms that are resolved by accessing data available in the current operating component's buffer do not need implementations to be developed. PeopleSoft Active Analytics Framework provides mechanisms to access data that is available in the component buffer.

Oracle recommends that when multiple related terms will be accessed during a single business event, you create a single implementation to return a rowset containing the data for several terms; then, specify which data element or field position in the rowset or record is to be used for the term.

You develop Implementations using:

- Application class.

  Use application class implementations when retrieval or derivation of data involves writing procedural code, or as a resolution method when data must be retrieved from an external source such as another PeopleSoft database or legacy systems. The application class can return data to the data library in a variety of forms: rowset, record, date, datetime, string, number, date array, datetime array, string array, number array, and array of any. Use PeopleSoft Application Designer to develop the application classes.

Note: Oracle does not recommend: 1) Using an application class to retrieve data from a component buffer; 2) Using an application class to retrieve the values for the binds directly by accessing the component buffer without registering them as implementation binds. Application classes must use Application Programming Interfaces (APIs) to retrieve the values for the implementation binds (input parameters).

- PS Query.

  Use Query Manager in PeopleSoft Application Designer to develop PS Query-based implementations. PS Query implementations are not appropriate for applications that get data from external databases or systems. The data library invokes the appropriate queries based on the information provided when you register the implementation. The data returned is available to the data library in the form of a rowset.

- SQL object.

  Use this implementation when the SQL used needs to be platform-independent and the data need not undergo complex transformations. SQL object implementations are not appropriate for applications that get data from external databases or systems. The data library invokes the appropriate SQL object based on the information provided when you register the implementation. The data returned is available to the data library in the form of a record or an array of any objects. Use PeopleSoft Application Designer to create a SQL object.

- Record.Field.

  Create a Record.Field-based implementation when the data can be retrieved directly from a table without going through complex transformations. The data returned is available to the data library in the form of string, date, datetime, number, string array, date array, datetime array, and number arrays.

Registering an Implementation

With the exception of component buffer implementations, all implementations must be registered in the PeopleSoft Active Analytics Framework. Before you register an implementation, you must define the PeopleSoft Application Designer objects if using application class, PS Query, or SQL Object implementation methods.

Specify the following items in the registration component:

- Functional name.
- Resolution method used for that implementation.
- Values for the parameters needed for invoking the implementation. The list of parameters varies depending upon the resolution method.
- List of binds that are expected by this implementation.

Note: The binds specified for an application class implementation are referenced in the application class object for retrieving the values. Therefore, changing these implementation bind names can have an adverse effect on the term resolution.

For IT users, the list of implementation binds specified are used for two purposes:

- To allow implementations to access these bind values.

  For any implementation, bind values are passed by position regardless of the resolution method used. Application class-based implementations alone have the additional capability to access the bind values by name.

- To allow the data library engine to use these binds to uniquely tag the data in application cache.

  If IT users take a shortcut by retrieving the necessary data by directly accessing the context (by not registering the data as implementation binds), the data library engine may, as a result, tag the data with incomplete key information. This could cause the same cached data to be incorrectly reused for resolving terms for which it is not valid.

Creating Terms

A *term* is a user-friendly name that refers to the data library content. It's essentially a piece of information that could exist in the PeopleSoft system or an external system, or it could be derived. For example, the data could be available in the component buffer; retrieved using a PS Query or an SQL object; or computed using an application class.

Terms are the building blocks in policies. Functional users can build conditions for a policy using terms present in the data library. Terms must be registered in the PeopleSoft Active Analytics Framework before they can be used.

Registering a term is a multistep process that includes:

1. Developing an implementation.
2. Registering the implementation.
3. Defining the term.
4. Associating the term with one or more subject areas.
5. Testing and activating the term.

Defining Term Properties

Defining a term involves specifying:

- Term name, code, and type (constant or variable).
- Data type.

  The data library supports primitive data types of string, number, datetime, date, time; and PeopleSoft-specific data types of record and rowset.

- Number of rows to be returned and whether they are scalar or vector (returning an array).

  Terms that are record or rowset data types have number of rows set to *One*.

  Note: Terms returning a vector (where value of number of rows is many) do not appear in the term list while you are building a condition for a policy.

- User binds.

  These are values that would be supplied either during the construction of a condition or at the time of associating the term with the application. Not all terms will have the binds; however, user binds may make a term more reusable.

- Optionally, details about how the data needs to be captured for user binds: whether a prompt or drop-down list box needs to be shown and how to derive the values.

- Which implementation needs to be used for resolving the term.

- Whether the term can potentially be resolved from any context, or only from specific contexts.

- How the data library needs to extract the data from the content returned by the implementation.

- Prompt details for the term.

  Specifying prompt details for a term is needed only when the term will be used to build a condition. The prompt details convey how the data needs to be captured on the right-hand side for a term participating in a condition.

- Configuration details for prompts.

  The details that you provide are used during the construction of a condition. When a term is selected as an element in a condition, the right-hand side widget will be constructed based on the configuration details specified for the prompts. You can configure the following prompt types:

  - Translate

    Specify a translate field name in which its values appear to the end user on the right-hand side of a condition.

  - Dropdown

    Specify a record name, data field name, and description field name. The record and data field names supply the valid data values to display in the drop-down list box; the description field provides a user-friendly description of the data value.

  - Prompt

    Specify a record name, data field name, and description field name. The record and data field names supply the valid data values to display in the prompt; the description field provide a user-friendly description of the data value.

  - Custom

    Specify a custom application class, a data field name, and a description field name. Valid values are retrieved by execution of the specified application class method and presented as a prompt.

- Scope of each term implementation.

  - When caching is activated for a term, data that is cached is uniquely identified by the implementation ID and all of the implementation bind values for that implementation.

- When scope is specified as the trigger point, after the first invocation of a term, subsequent references to the same term in one or more policies associated with the same trigger point force the data library engine to retrieve the data from the cache, provided that all the values for the implementation binds match those of the ones belonging to the data present in the cache.
- When scope is defined as a component, the longevity of the data is for a specific instance of the transaction.
- When scope is defined as global, the cached data is available for the entire user session.
- When scope is defined as *Do Not Cache,* data is retrieved by invocation of the implementation every time.

- Association of a term with subject areas.

  Subject areas act as file cabinets. You must assign a term to at least one subject area, but you can associate it with more than one.

Note: PeopleSoft Active Analytics Framework does not format the data. The term user or term implementer is responsible for formatting it according to his or her needs. For example, the term Current Date is always resolved using the standard YYYYMMDD format.

Using Generic Implementations

A generic implementation can resolve terms within the requesting context. You define generic implementations for terms when they can be used in various contexts and when any new contexts may want to use that term.

Examples of generic implementations are:

- Customer-specific measures such as customer value, the number of cases reported in a period of time, or the number of telephone interactions with the customer.
- Customer profile information, such as first and last name, email address, and customers within a segment.

Using Contextual Implementations

If the input data needed for invoking an implementation is too specific and cannot be supplied outside of the component, then the implementation must be associated with the component's context. For example, terms such as case status, order creation date, and case description cannot be resolved from components other than those in which they are present.

Terms that have different implementations depending upon their contexts will have an implementation associated with a specific context. For example, the term *Revenue for a customer / segment / segment group* is computed based on the context from which it originates. The implementation specific to the customer context calculates the revenue value from that customer. The implementation specific to a segment context calculates the revenue value generated from all the customers belonging to that segment, and so on for each segment group.

Managing Terms

This section discusses how to manage terms.

Pages Used to Manage Terms

Page Name	*Definition Name*	*Usage*
Term Definition Page	EOCF_TERM_DEFN	Define and manage terms.
Subject Areas Page	EOCF_TERM_SUBAREA	Define the subject area details.
Policy Options Page	EOCF_TERM_INACTION	Define policy options.
Extended Attributes Page	EOCF_TERM_ATTR	Add additional attributes to terms.
Notes Page	EOCF_TERM_NOTES	Add notes to the term.
Define Implementation Page	EOCF_IMPL_DEFN	Define an implementation.
Notes Page	EOCF_IMPL_NOTES	Add notes to the implementation.
Test Term Implementation Page	EOCF_TEST_TERM	Test a term's implementation.

Term Definition Page

Use the Term Definition page (EOCF_TERM_DEFN) to define and manage terms.

Navigation

Enterprise Components > Active Analytics Framework > Data Library > Manage Terms > Term Definition

Image: Term Definition page (1 of 2)

This example illustrates the fields and controls on the Term Definition page (1 of 2). You can find definitions for the fields and controls later on this page.

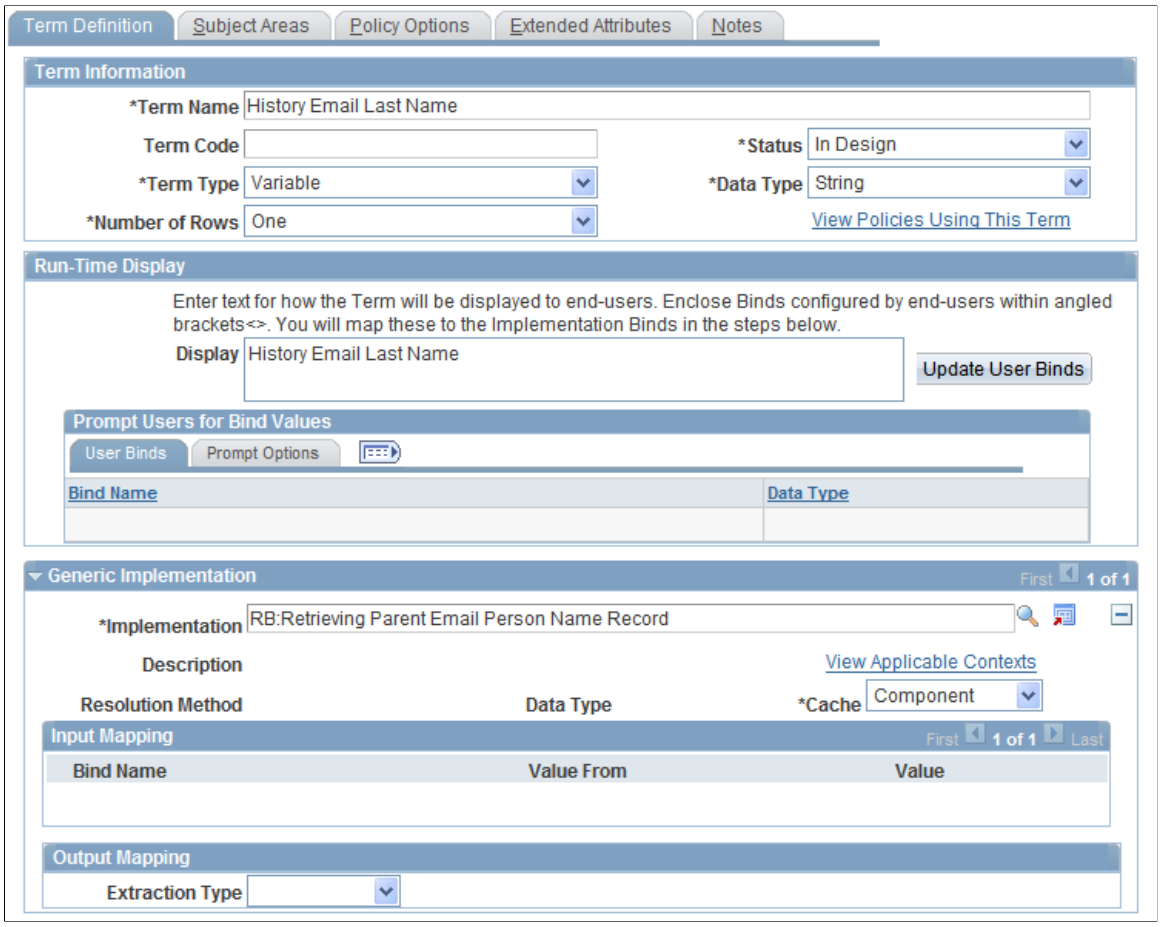
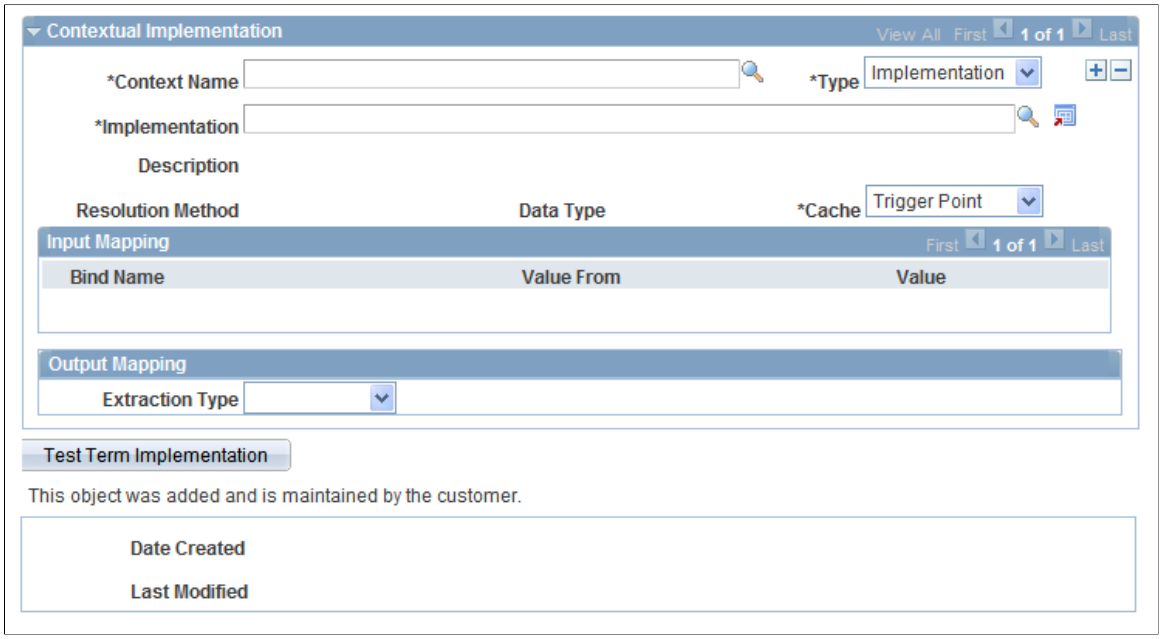


Image: Term Definition page (2 of 2)

This example illustrates the fields and controls on the Manage Terms page (2 of 2). You can find definitions for the fields and controls later on this page.



Term Name The unique identifier of the term. This is the label that will be displayed to the functional users. Though allowed, Oracle recommends that special characters not be used in term names.

Term Type Select to specify whether a term is a variable or constant. Variable terms must have at least one implementation.

Term Code Uniquely identifies a term when you access a term programmatically. This is user-defined.

Number of Rows The number of rows to be returned, one or many (scalar or vector). If this field is *Many,* the term cannot participate in policy conditions.

Note: Applications directly integrating with the data library are responsible for converting the resolved output value of a term (which will be of data type *any*) to the appropriate data type.

Status Valid values are *Active, Inactive,* and *In-Design.* Only active terms are used in policy conditions and other applications.

Data Type Returns the data type of the term. Possible values are string, number, date, datetime, time, record, and rowset.

Run-Time Display Specify user binds for this term, which will be needed when the resolved value of the term depends on user-defined binds.

Prompt Users for Bind Values Specify the bind name; (optional) specify prompt options.

Generic Implementation	Specify a generic implementation. Generic implementations are resolved by deriving the bind values from the runtime context. Terms having generic implementations can be resolved by multiple contexts. You specify a generic implementation by selecting an existing implementation from the prompt or creating a new one using the Create button. Click the View applicable contexts link to view the list of contexts in which this term would be resolved.
Contextual Implementation	Select an implementation that is specific to a context. Contextual implementations are resolved by deriving the bind values from this specific context.
Input Mapping	Maps the implementation binds to context variables or to constant values. If the term has user binds, one or more implementation binds must be mapped to the user binds. For generic implementations, this mapping is critical for this term to be resolved by multiple contexts.
Output Mapping	Specify the extraction parameters for a term implementation such that a subset of the value returned by the implementation is returned as the resolved value of the term.

Note: Use caution when making changes to the term definition after the term has been associated with one or more policies. Changes to term attributes such as data type, number of rows, implementation category, and implementation details; or changing a non-configurable term to a configurable term and vice versa, could have significant impact on the policies that reference this term. These changes could possibly result in invalidating these policies. Before making any of these changes, view the policies using a term by clicking the link View Policies Using This Term.

Test Term Implementation Page

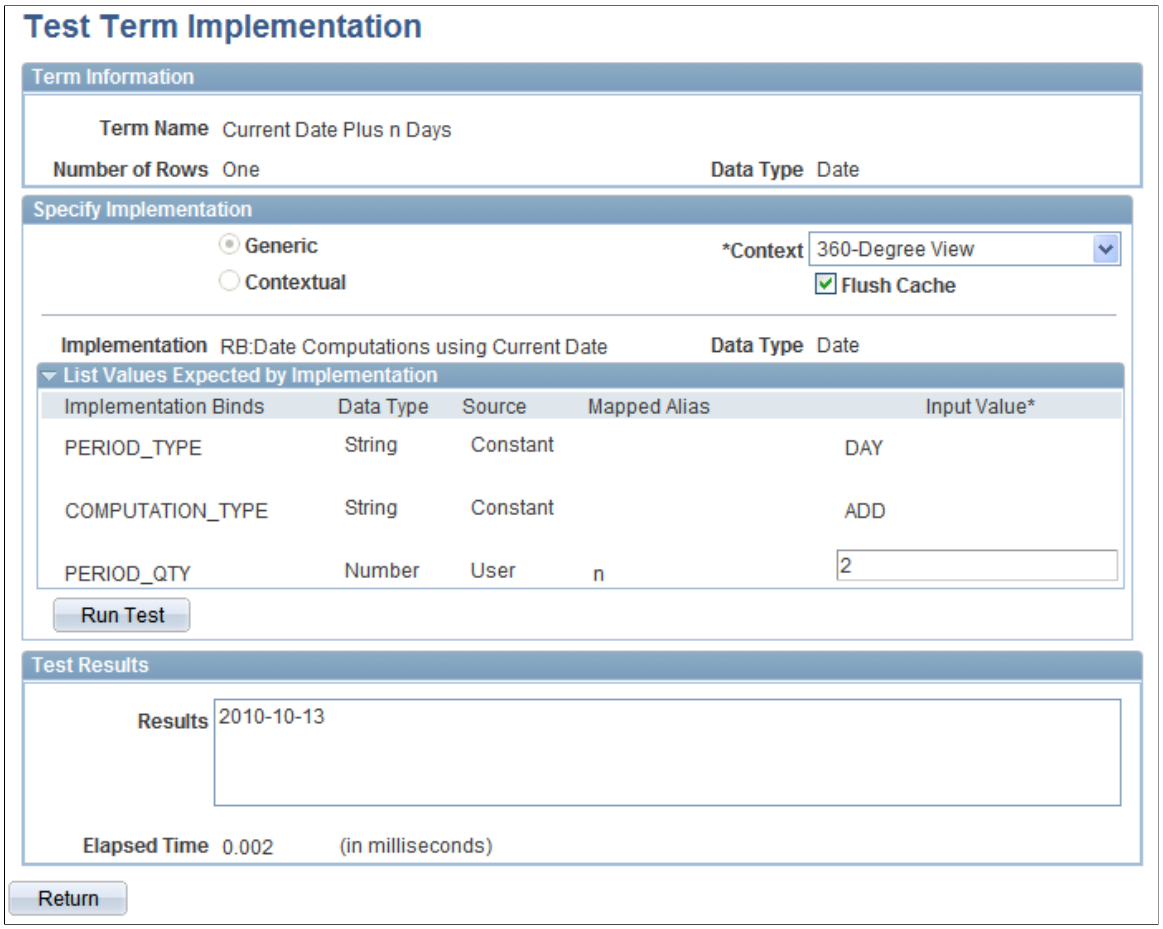
Use the Test Term Implementation page (EOCF_TEST_TERM) to test a term's implementations.

Navigation

Enterprise Components > Active Analytics Framework > Data Library > Manage Terms > Test Term Implementation

Image: Test Term Implementation page

This example illustrates the fields and controls on the Test Term Implementation page. You can find definitions for the fields and controls later on this page.



Specify Implementation

Specify whether you want to test the generic or contextual implementation defined for the term.

- If you select Contextual, you must specify a context name from the drop-down list box to display a list of bind values that are required by this implementation in a grid.
- If you select Generic, the context from which the term is resolved must be specified.

Select Flush Cache if you do not want the system to fetch the value for this implementation from the memory cache.

List values

Enter sample values for the parameters expected by the implementation and click Run Test.

Test Results

Displays the resolved value of a term implementation being tested and the elapsed time to retrieve the value.

Note: Context-variable implementations of a term cannot be tested. Also, terms that have application class implementations accessing data from a component buffer or directly from the context cannot be tested in the Term Tester page. Testing these terms will result in an error message.

Chapter 4

Managing Contexts

Understanding Contexts

Contexts are a key component of how the PeopleSoft Active Analytics Framework works—their purpose is to describe the computing environment from which the decision engine is invoked and select the appropriate term implementation at runtime.

Online and Generic Contexts

Contexts can be online or generic. Online contexts must be associated with a PeopleSoft online page, whereas generic contexts can be used anywhere (including online pages).

The data elements within a context are called *context variables.* Online contexts have built-in context variables—they are the fields of the online page. However, the data elements in an online context are not limited to the online fields; both online and standalone contexts can have additional context variables defined.

Within a given context, all context variables must be assigned unique code names called *aliases.* Aliases are used to associate data with the input binds required by term implementations. For example, a field representing the customer ID may be named CUSTOMER_ID in the application page, but may have an alias of CUST_ID. Using aliases enables terms to be used in the largest possible set of contexts. Page variables exist on a one-to-one basis with their underlying fields; that is, two or more context variables may point to the same page field as long as the aliases of these context variables are distinct. This enables the context user to have aliases named both CUSTOMER_ID and CUST_ID, thereby facilitating term reuse. Context variables are exposed to the user by corresponding term definitions.

At runtime, applications can request the data library engine to automatically populate the online context in memory, provided that the request is made from the component from which the online context can be constructed. In case of generic contexts, applications can either construct the context explicitly by populating values for these context variables, or request the data library to automatically copy values from level 0 context variables of an online context to similar context variables of a generic context.

Context variables can be:

- Page variables that correspond to fields within PeopleSoft application pages.

  Page variables are also referred to as native context variables, because they are native to the pages from which their values come.

  Note: Use caution when altering the component structure after generating page variables for an online context (using the Generate Context component). Changes made to existing fields on the application page might invalidate the corresponding context variables and the terms from which they are resolved.

- Constants.

Generic contexts usually consist of named constants. Constant-type context variables may not have specific values associated with them when the context is registered; some of these variables may get values at runtime.

- Term variables.

  These context variables are data library terms that have been inserted into the context. A context may have any number of terms included within it. However, if a term's implementations require binds, then the context must provide them from its page variables and constants. Term variables can be used to add extended data elements for implementation binds and actions without your having to customize the application.

Configuring Contexts

This section discusses how to generate and manage context objects.

Pages Used to Configure Contexts

Page Name	*Definition Name*	*Usage*
Generate Context Object Page	EOCF_CTX_IMPORT	Generate context objects.
Manage Context Object - Definition Page	EOCF_CONTEXT_DEFN	Manage context objects.
Manage Context Object - Notes Page	EOCF_CTXDEFN_NOTES	Enter text descriptions for context objects.

Generate Context Object Page

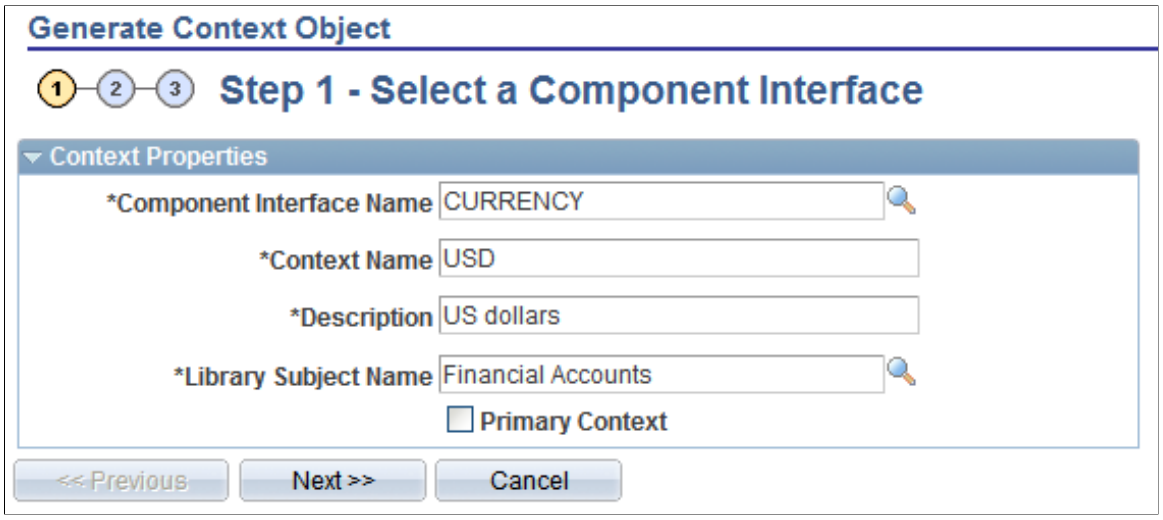
Use the Generate Context Object page (EOCF_CTX_IMPORT) to generate context objects.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Generate Context Object

Image: Generate Context Object page (1 of 4)

This example illustrates the fields and controls on the Generate Context Object page (1 of 4). You can find definitions for the fields and controls later on this page.



Component Interface Name Used to extract the contents of a PeopleSoft component for use in generating the context. A component interface is required to create an online context.

Context Name The required, unique, and descriptive name of a context.

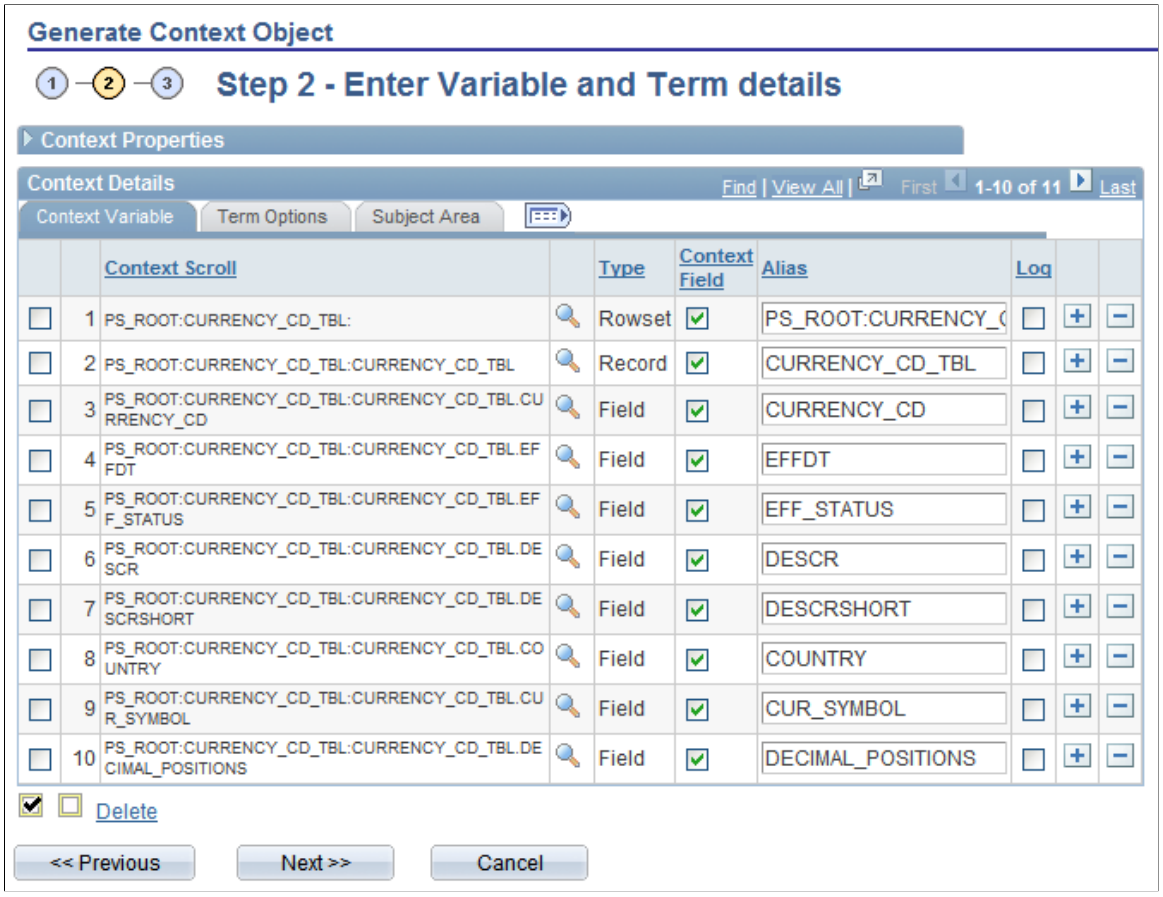
Description The required, appropriate description of the intent of this context.

Library Subject Name The default subject area for terms created by this process. The prompt for this field shows the subject area selection list.

Primary Context Select to denote the context to be generated as the primary context for this online transaction.

Image: Generate Context Object page (2 of 4)

This example illustrates the fields and controls on the Generate Context Object page (2 of 4). You can find definitions for the fields and controls later on this page.



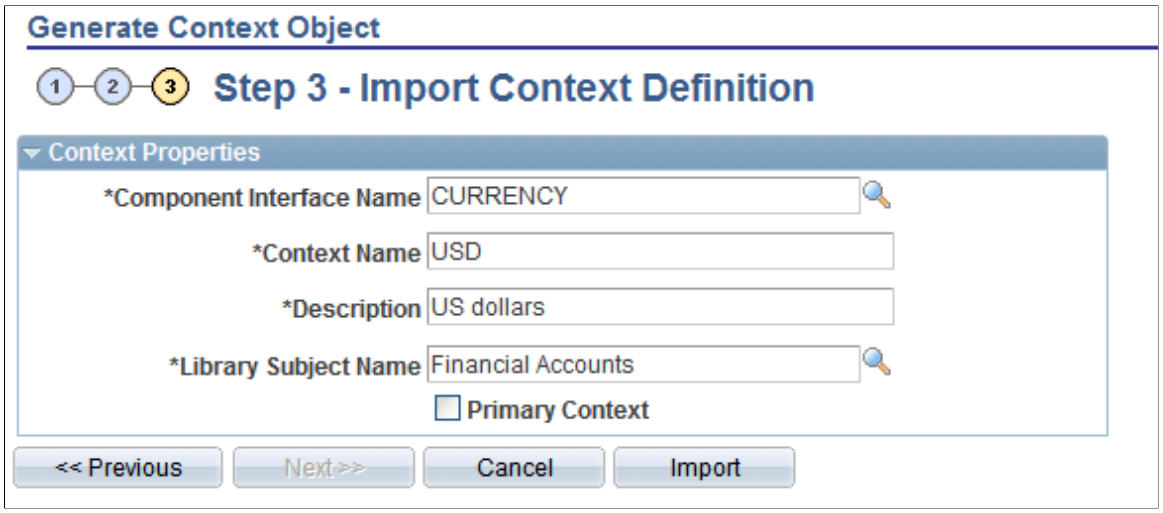
Review and configure the context variables and terms that were automatically generated based on the component interface that you selected. Select the Log check box to denote that the field should be logged when a context is persisted at runtime (this is a key to the transaction, and the process automatically sets this field).

Select the Term Options tab to create a new term for each context variable that is created. Also, you can add a contextual implementation to an existing term for this context variable.

Select the Subject Area tab if you want to override the default subject area chosen on the first page.

Image: Generate Context Object page (3 of 4)

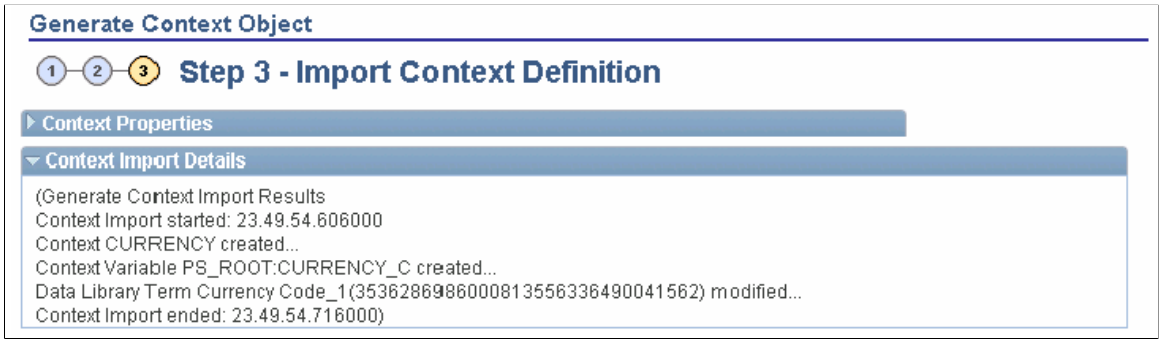
This example illustrates the fields and controls on the Generate Context Object page (3 of 4). You can find definitions for the fields and controls later on this page.



Select Import to create the context and terms. The Generate Context Object page displays the import details.

Image: Generate Context Object page (4 of 4)

This example illustrates the fields and controls on the Generate Context Object page (4 of 4). You can find definitions for the fields and controls later on this page.



Manage Context Object - Definition Page

Use the Manage Context Object - Definition page (EOCF_CONTEXT_DEFN) to manage context objects.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Manage Context Object

Image: Manage Context Object - Definition page (1 of 2)

This example illustrates the fields and controls on the Manage Context Object - Definition page (1 of 2). You can find definitions for the fields and controls later on this page.

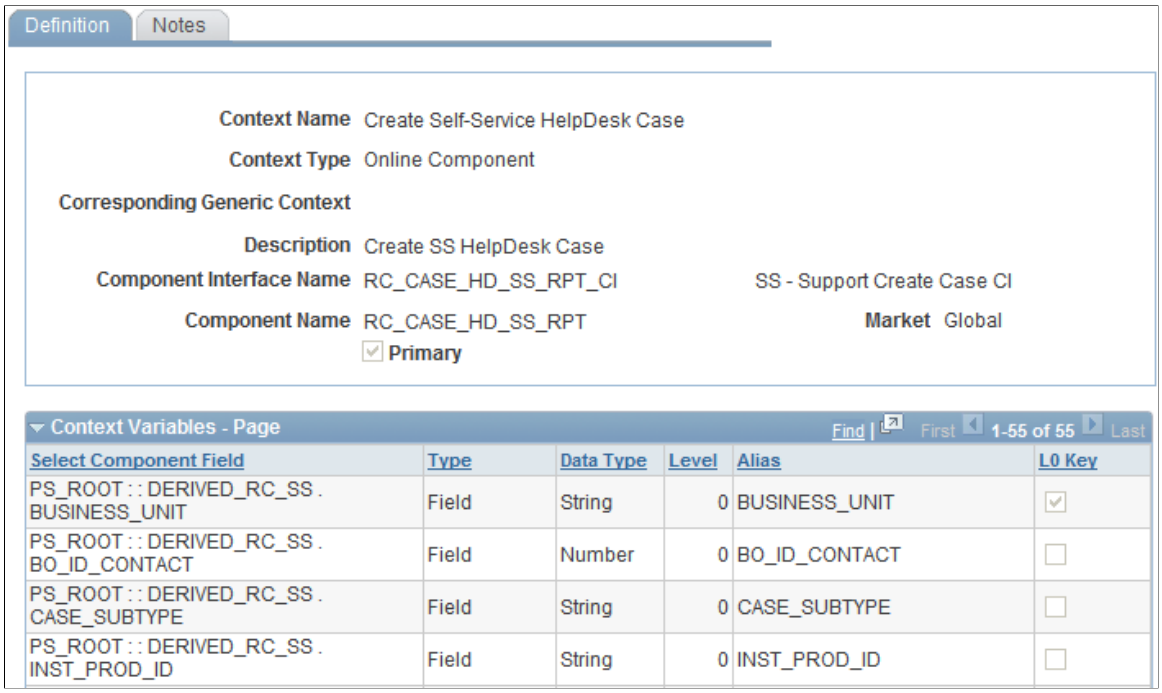
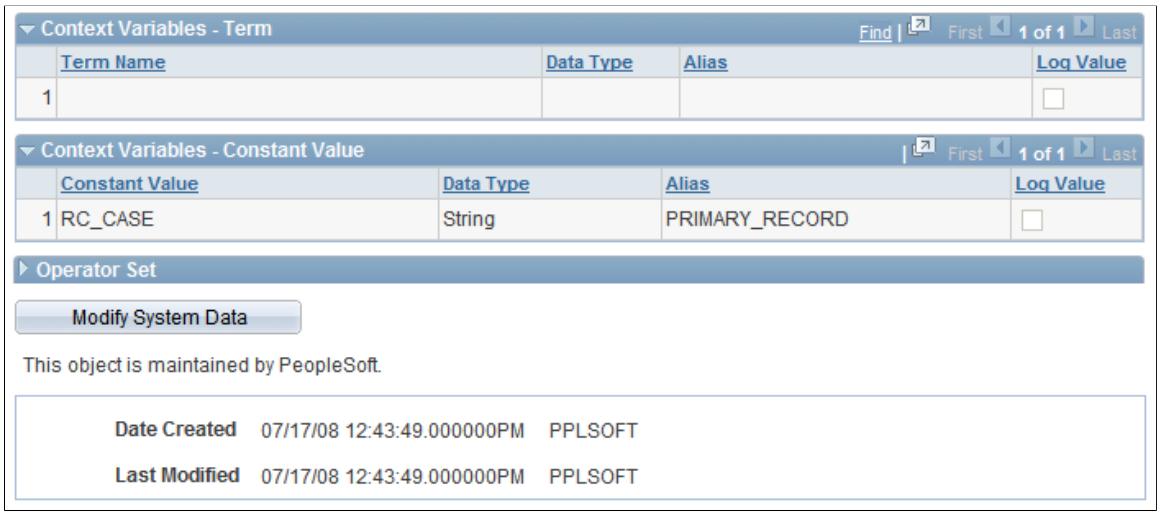


Image: Manage Context Object - Definition page (2 of 2)

This example illustrates the fields and controls on the Manage Context Object - Definition page (2 of 2). You can find definitions for the fields and controls later on this page.



You can, if applicable, enter descriptive information in the text box on Manage Context Object - Notes page.

Corresponding Generic Context This field is used to programmatically create a standalone context from its corresponding online context. Applications directly embedding data library terms need this information to supply context information to the data library runtime engine.

Context Variables - Page

This section is used to define the page context variables.

- Select Component Field. Click to select a component field from the displayed component buffer hierarchy.
- Type. This column specifies the object type of the selected component field. Valid values are *Field, Record,* and *Rowset.*
- Data Type. Specifies the PeopleSoft data type of the selected component field.
- Level. Specifies the scroll level of the selected component field.
- Alias. The unique identifier for a context variable.
- L0 Key. Specifies whether a component field is a key field for this component.

Context Variables - Term

This section defines the context variables of the type term.

- Term name. Select a term name from the prompt.

  The resolved value of this term is used as the value of this context variable.

- Data Type. Specifies the data type of the term selected.

Context Variables - Constant Value

This section defines the context variables of the type constant.

- Constant Value.
  - Specify a constant value to be used as the runtime value for this context variable.
  - Leave blank and a value has to be supplied at runtime.
- Data Type. Specify the data type of this context variable.

Note: A constant value specified for a context variable cannot be overridden at runtime.

Operator Set

Optionally, specify an operator set to be used for this context. This operator set is used to present the list of valid operators for selected terms in policy conditions. A default operator set is automatically entered.

Chapter 5

Setting Up the Action Framework

Understanding the Action Framework

The action framework enables users to specify the actions to be executed when policy conditions evaluate to true within a trigger point.

Architecture of the Action Framework

The architecture of the action framework comprises:

- An action type registration component for creating and maintaining action types by programmers and IT staff. An *action type* is metadata pertaining to a particular class of actions that might be performed at runtime.
- An action type bundle registration component for creating and maintaining action type bundles, or classes of actions that can be combined. An action type can be in only one action type bundle.

  Note: In this release, if other display action types are created by a PeopleSoft product line, they must be combinable with the delivered display alert action type and must be registered in the display action type bundle.
- A design-time environment facilitating the embedding and configuring of consequent actions within policies.
- A runtime environment that enables the decision engine to invoke particular actions as needed.
- A generic display alert action type, which is available to all product lines using the framework. It can be used to display important information about a customer or a suggestion for a course of action. For example, if a high-value customer calls on the phone, a pop-up window appears with an alert and a suggestion that the phone connection be routed to a manager.

Understanding Action Types

An *action type* refers to a category of actions that can be associated with a policy. For use in the framework, an action type must be registered with the following information:

- Location of the code that handles the design-time and runtime aspects of the action type.
- Configuration requirements.

  When adding actions of specific action type to a policy, policy-specific configuration requirements must be set before the policy can be enacted.
- Whether actions of an action type terminate the analytic processing.

For example, if the action is a transfer from an application page to another page, the action terminates the framework processing that was triggered by a trigger point associated with the original page.

Note: A terminal action type is not combinable.

- Whether the action can be combined with other actions at runtime. Such an action is referred to as a *combinable* action.

  For example, the display alert action type can be combinable; therefore, when two display alert actions are specified, they are combined, and a single window appears that includes both alerts.

- Whether the action is part of an action bundle.

  If actions of *different* action types are combinable with each other, the action types must be included in an action type bundle. Consequently, if combinable action types are not combinable with certain other action types, then the action type does not need to be part of an action type bundle.

- Triggering environment—where the action can be deployed.

  For example, the display alert action executes only from an application page, not from an Application Engine program.

- The trigger types and trigger points that include the action type as a valid action type.

  For example, you may want an action to be valid for a ComponentPostBuild trigger type, specifically the *When a Customer is Presented* trigger point, associated with this trigger type.

Understanding How Actions Execute

After the framework evaluates conditions for all a trigger point's policies, the actions associated with the policies having true conditions are forwarded to the action framework. The action framework is responsible for firing the actions.

Only one terminal action can be fired for any trigger point. Because a trigger point may be associated with one or more policies, and each policy may include more than one action, more than one terminal action might be associated with a trigger point. In this case, the first terminal action in the first policy that evaluates to true is executed. If present, a terminal action fires after executing all the nonterminal actions.

Individual action types determine how actions can be combined. Some action types, such as the display alert, combine all their actions from the same trigger point. Therefore, at runtime all the display items appear in the same pop-up window.

Note: For display actions to execute, pop-up blockers must be turned off.

An action that is not combinable will not affect the execution of another action.

Registering Action Types and Action Type Bundles

This section discusses how to register action types and action type bundles.

Pages Used to Register Action Types and Action Type Bundles

Page Name	*Definition Name*	*Usage*
Register Action Type Page	EOCF_ACTN_TYPE_REG	Register an action type.
Action Type Triggers Page	EOCF_ACT_TYP_EVNTS	Specify action type triggers.
Register Action Type Bundle Page	EOCF_ACTION_BUNDLE	Register action type bundles.

Register Action Type Page

Use the Register Action Type page (EOCF_ACTN_TYPE_REG) to register an action type.

Navigation

Enterprise Components > Active Analytics Framework > Action Framework > Register Action Type

Image: Register Action Type page

This example illustrates the fields and controls on the Register Action Type page. You can find definitions for the fields and controls later on this page.

Action Type Name The name of a class of similar actions.

DesignTime Action Behavior Specify the design time details of the action type. When you add actions of this action type in a policy, these design time specifications are used to present the action type configuration page and store the configuration.

RunTime Action Behavior	Specify the runtime details of the action type. The application class details specified here are executed at runtime to trigger actions of this type.
Triggering Environment	Specify the triggering environments to be supported for this action type.

Action Type Triggers Page

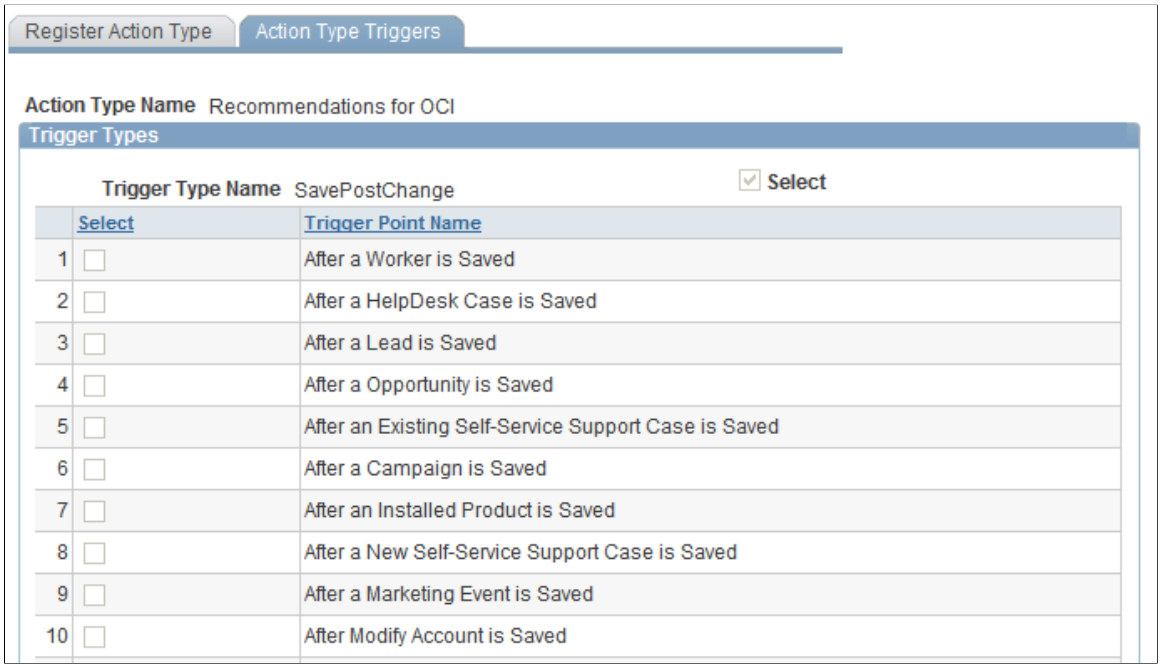
Use the Action Type Triggers page (EOCF_ACT_TYP_EVNTS) to specify action type triggers.

Navigation

Enterprise Components > Active Analytics Framework > Action Framework > Register Action Type > Action Type Triggers

Image: Action Type Triggers page

This example illustrates the fields and controls on the Action Type Triggers page . You can find definitions for the fields and controls later on this page.



Select the appropriate check boxes to associate this action type with listed trigger types and trigger points.

- Selecting a trigger type makes this action type available for the selected trigger type.
- One or more trigger points can be selected only if the corresponding trigger type is selected.

Register Action Type Bundle Page

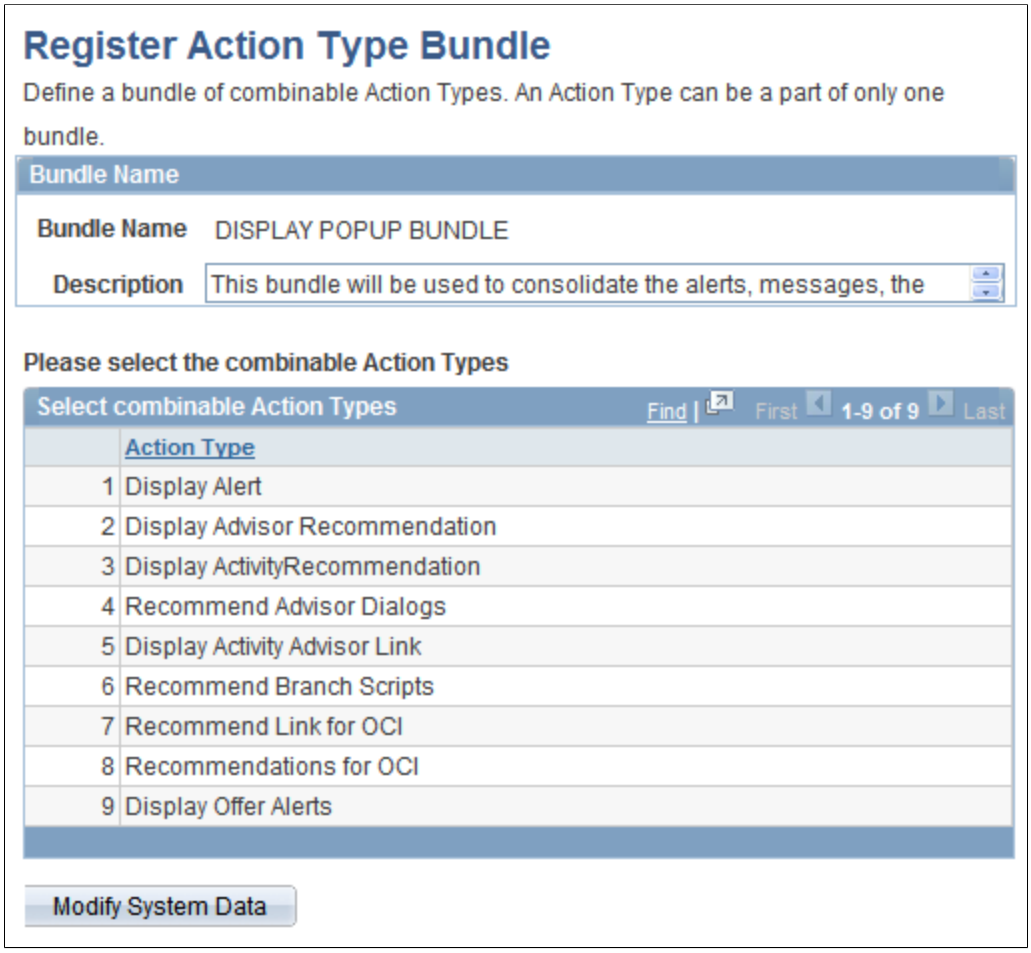
Use the Register Action Type Bundle page (EOCF_ACTION_BUNDLE) to register action type bundles.

Navigation

Enterprise Components > Active Analytics Framework > Action Framework > Register Action Type Bundle

Image: Register Action Type Bundle page

This example illustrates the fields and controls on the Register Action Type Bundle page. You can find definitions for the fields and controls later on this page.



Enter a name and description for this action type bundle. Select the action types that can be combined from the drop-down list box in each row.

Chapter 6

Administering the PeopleSoft Active Analytics Framework

Setting Logging and Installation Options

This section discusses how to set the data library log and installation options.

Pages Used to Set Logging and Installation Options

Page Name	*Definition Name*	*Usage*
Data Library Log Settings Page	EOCF_DL_LOG_SET	Set Data Library logging options.
Install Options Page	EOCF_INSTALL	Set Active Analytics Framework installation options.

Data Library Log Settings Page

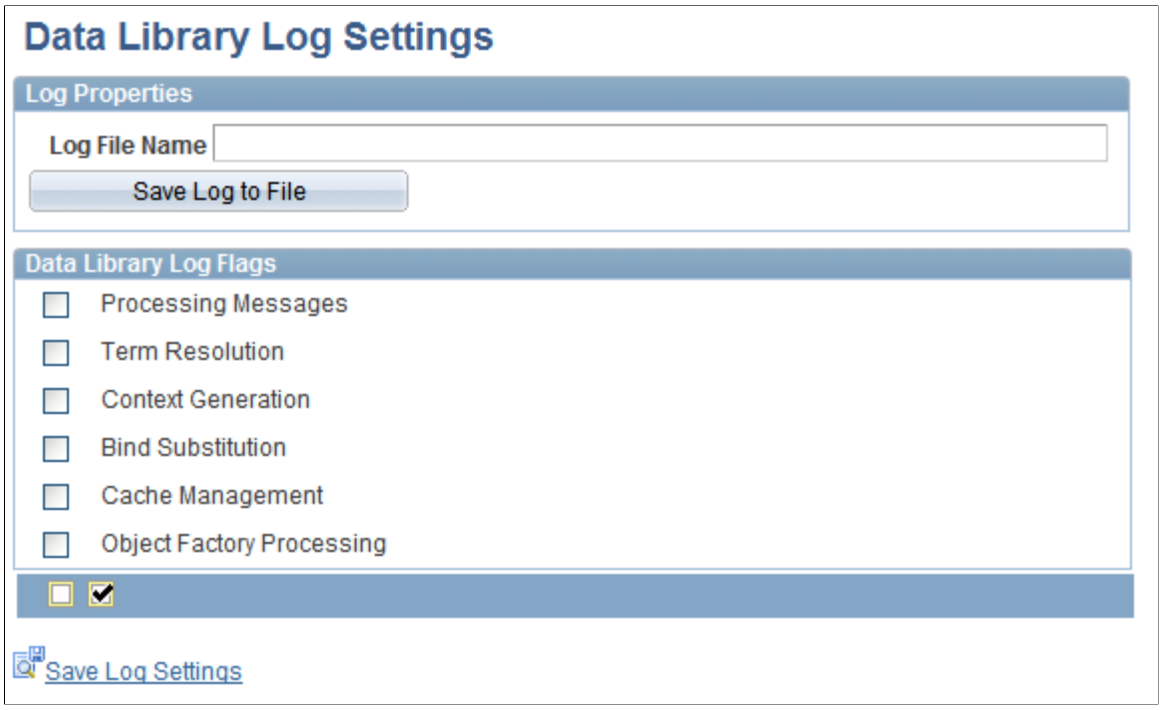
Use the Data Library Log Settings page (EOCF_DL_LOG_SET) to set the data library logging options.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Data Library Logging Settings

Image: Data Library Log Settings page

This example illustrates the fields and controls on the Data Library Log Settings page. You can find definitions for the fields and controls later on this page.



Log File Name	Enter a name for the log file and select Append to an Existing File, if desired.
Term Resolution	Log technical details of the resolution of individual terms.
Context Generation	Log technical details about the generation of contexts.
Bind Substitution	Log technical details about implementation binds substituted with values from the runtime context.
Cache Management	Log technical details about usage of cache.
Object Factory Processing	Log technical details about the loading of framework object definitions.

Install Options Page

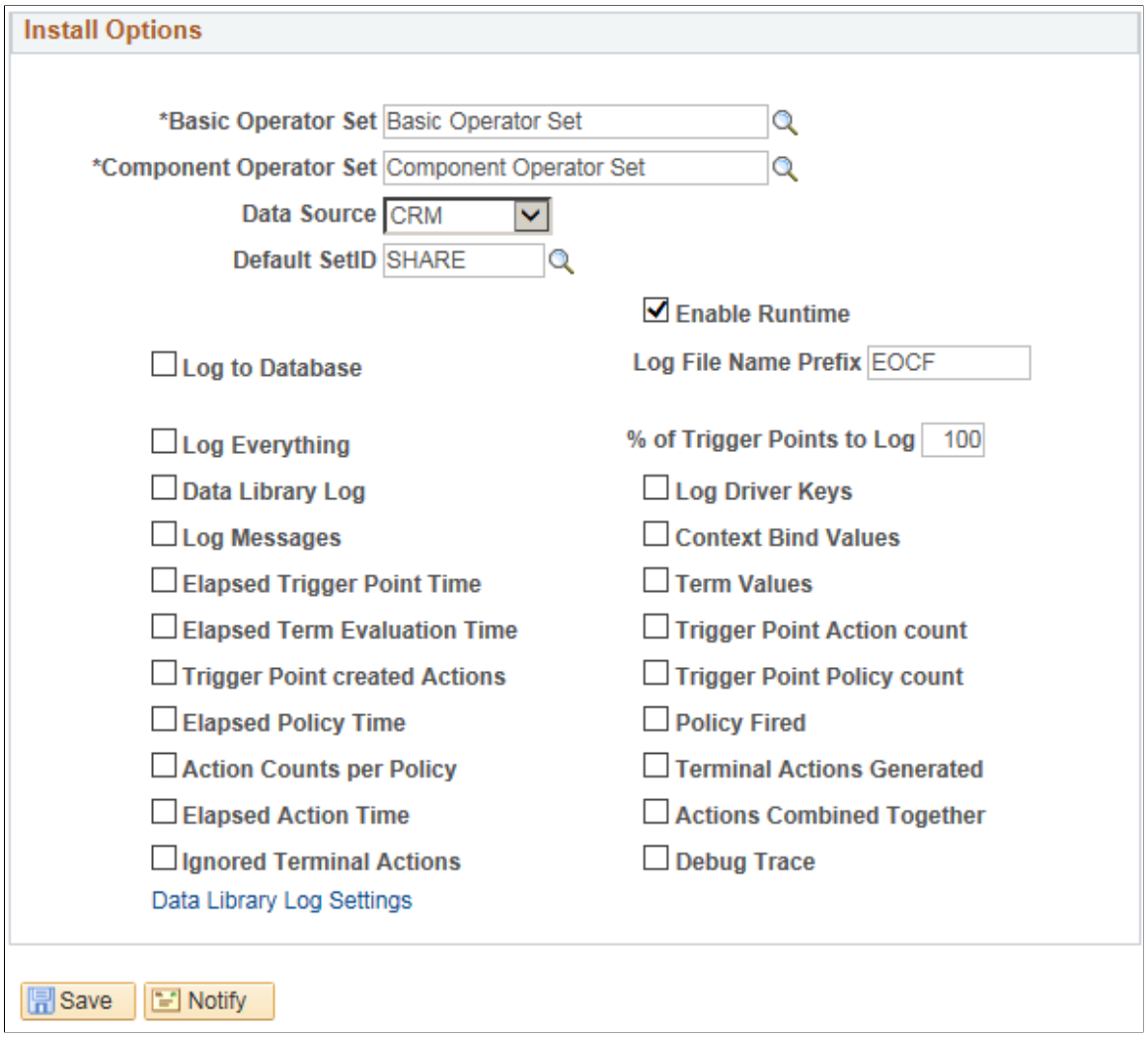
Use the Install Options page (EOCF_INSTALL) to set Active Analytics Framework installation options.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Install Options

Image: Install Options page

This example illustrates the fields and controls on the Install Options page. You can find definitions for the fields and controls later on this page.



Data Source	The product line for this database, for example CRM or HCM.
Default SetID	The setID value to use for a component that does not use set control.
% of Trigger Points to Log	When you are logging for performance monitoring, the impact of logging can be alleviated only by logging a percent of trigger points executed. Specify a percent value between 1 and 100.
Enable Runtime	This option enables all trigger points from running. **Note:** Enable Runtime" is necessary for ‘Preview the form’.
Log to Database	Determines whether the log will be written to the database or to a text file.
Log File Name Prefix	If a file is logged instead of the database, this prefix is prepended to each user's log file.

Note: The decision engine log file is created in the default files directory for this application server instance. Typically, the application server uses the directory %PS_CFG_HOME%\appserv\<servername>\files. The application server creates this directory as needed; therefore, if no log files have been written, it may not yet exist.

Log Everything	If selected, causes all normal logging options to be activated.
Data Library Logging	Includes the data library log entries in the decision engine log.
Log Driver Keys	Log the primary key values driving each trigger point's context (transaction).
Log Messages	Log messages that policies or actions may generate.
Context Bind Values	Log values retrieved directly from the operant context.
Elapsed Trigger Point Time	Log the time required to complete a trigger point.
Elapsed Term Evaluation Time	Log the time required to evaluate a term.
Term Values	Log the values of evaluated data library terms.
Trigger Point Action Count	Log the number of actions created by the execution of a trigger point.
Trigger Point Policy Count	Log the number of policies evaluated for a trigger point.
Trigger Point created Actions	Log the actions created during the evaluation of policy conditions for a trigger point.
Elapsed Policy Time	Log the time required to evaluate the condition portion of a policy.
Policy Fired	Log whether a policy condition was true or false.
Action Counts per Policy	Log the number of actions created by a policy.
Terminal Actions Generated	Log terminal actions created by a policy.
Elapsed Action Time	Log the time that an action to takes to finish.
Actions Combined Together	Log actions that were combined during execution.
Ignored Terminal Actions	Log any terminal actions that could not be invoked.
Debug Trace	Log debugging information; this option is not normally used.

Registering Action Objectives

This section discusses how to register action objectives.

Page Used to Register Action Objectives

Page Name	Definition Name	Usage
Register Action Objective Page	EOCF_ACTION_OBJ	Use to functionally categorize actions defined in policies.

Register Action Objective Page

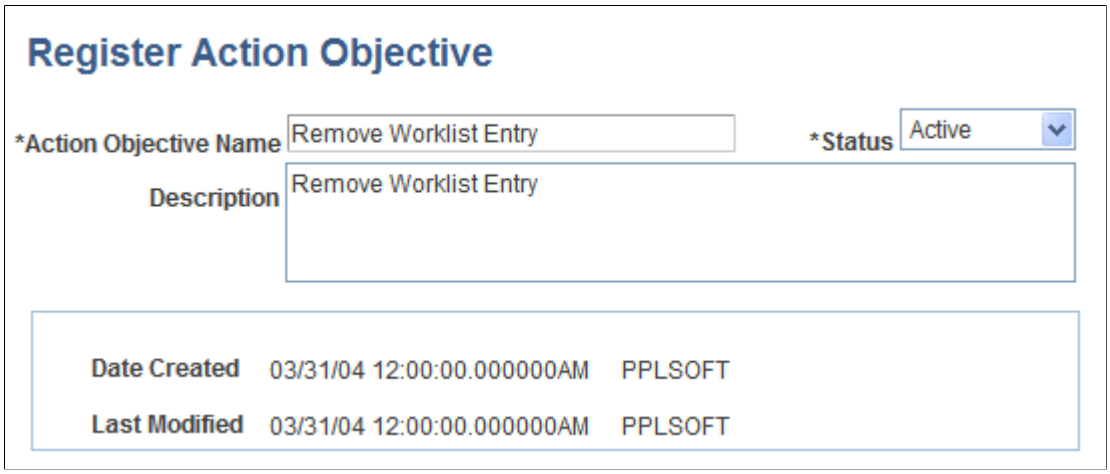
Use the Register Action Objective page (EOCF_ACTION_OBJ) to use to functionally categorize actions defined in policies.

Navigation

Enterprise Components > Active Analytics Framework > Action Framework > Register Action Objective

Image: Register Action Objective page

This example illustrates the fields and controls on the Register Action Objective page. You can find definitions for the fields and controls later on this page.



Enter the action objective name and a description.

Registering Trigger Types and Trigger Points

This section discusses how to register trigger types and trigger points.

Pages Used to Register Trigger Types and Trigger Points

Page Name	Definition Name	Usage
Register Trigger Type Page	EOCF_EVTYP_DEFN	Define trigger types.
Register Trigger Point Page	EOCF_EVENT_DEFN	Define trigger points.

Register Trigger Type Page

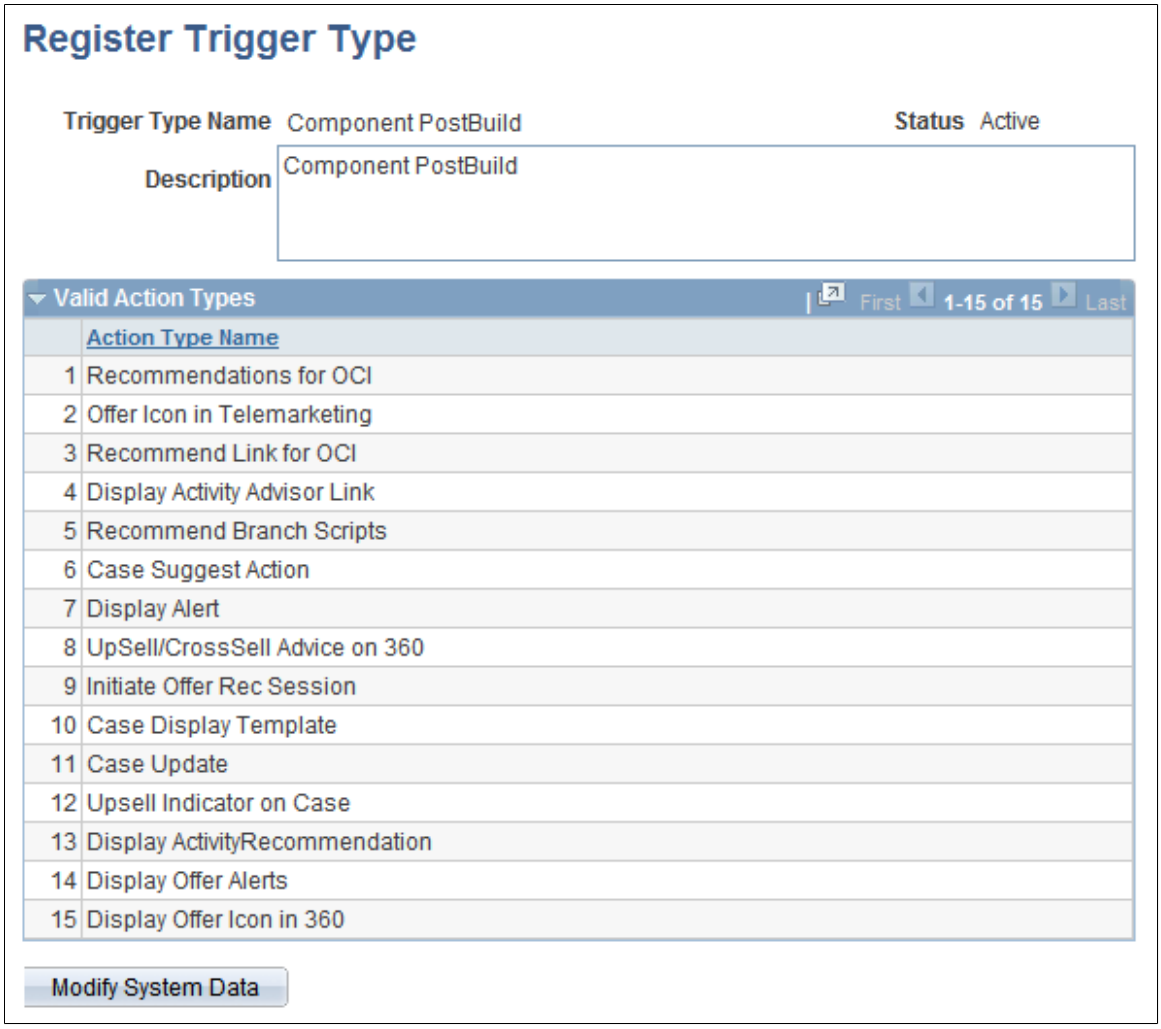
Use the Register Trigger Type page (EOCF_EVTYP_DEFN) to define trigger types.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Register Trigger Type

Image: Register Trigger Type page

This example illustrates the fields and controls on the Register Trigger Type page. You can find definitions for the fields and controls later on this page.



Enter details about the trigger type that you are defining and select the valid actions.

Register Trigger Point Page

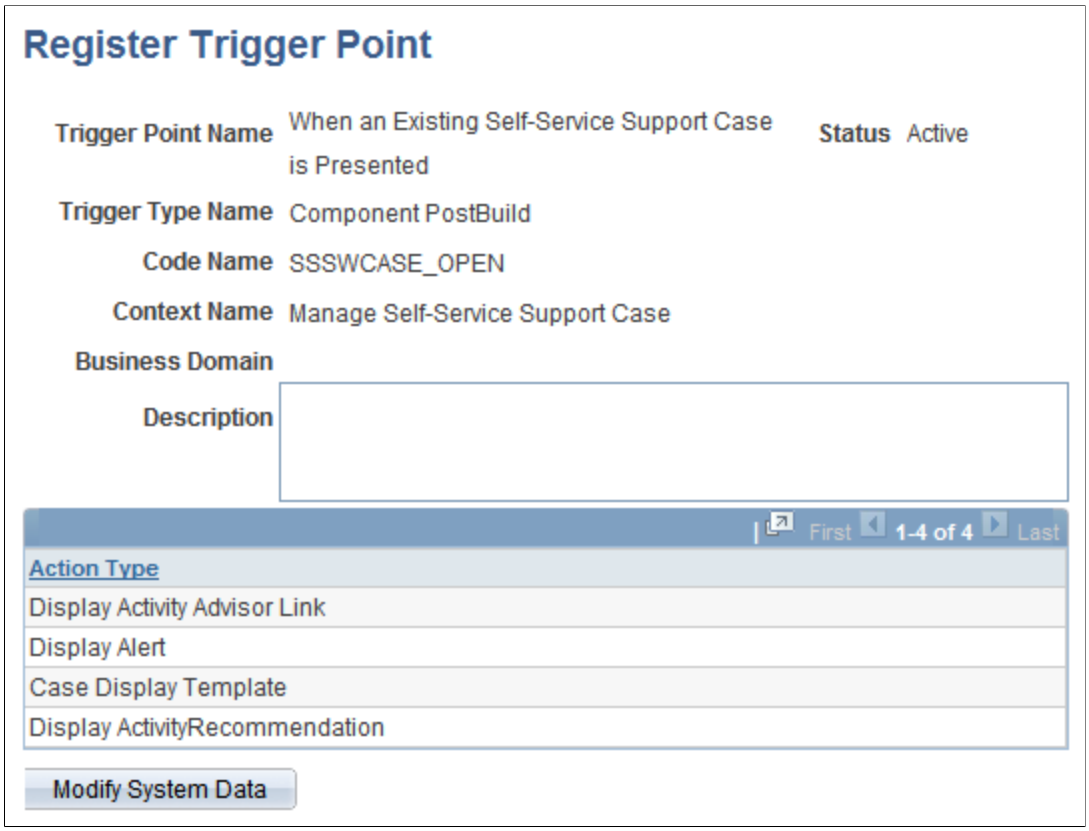
Use the Register Trigger Point page (EOCF_EVENT_DEFN) to define trigger points.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Register Trigger Point

Image: Register Trigger Point page

This example illustrates the fields and controls on the Register Trigger Point page. You can find definitions for the fields and controls later on this page.



Trigger Point Name The unique and descriptive name of the trigger point.

Trigger Type Used by IT developers to determine which PeopleSoft event in the component needs to be enabled with this trigger point. This option also restricts the list of valid action types that can be used when you are creating policies for this trigger point.

Code Name Use to reference the trigger point when programmatically enabling this trigger point for a PeopleSoft component. Changes made to the code name, after you enable this trigger point for a PeopleSoft component, may disrupt the execution of policies for this trigger point at runtime.

Context Name Defines the context for the trigger point.

Defining Subject Areas

This section discusses how to define subject areas.

Page Used to Define Subject Areas

Page Name	Definition Name	Usage
Define Subject Areas Page	EOCF_SUBJ_HIER	Define subject areas.

Define Subject Areas Page

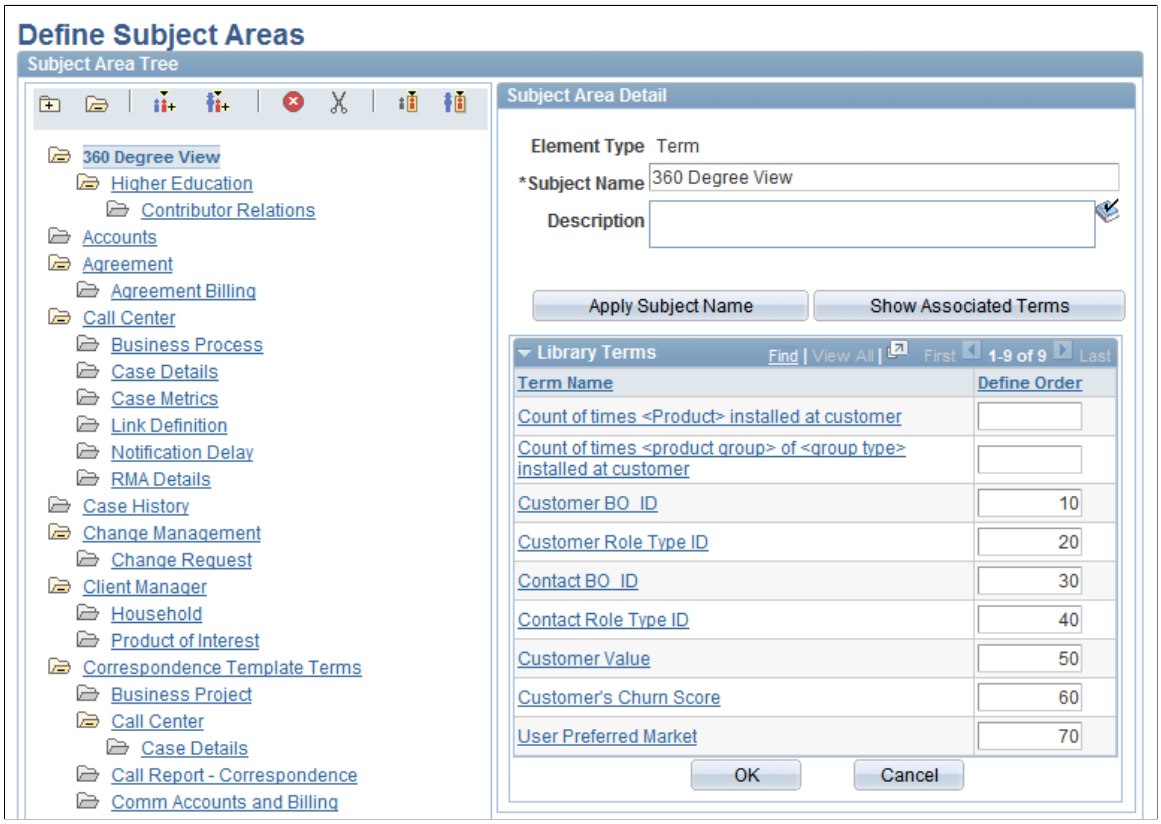
Use the Define Subject Areas page (EOCF_SUBJ_HIER) to define subject areas.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Define Subject Area

Image: Define Subject Areas page

This example illustrates the fields and controls on the Define Subject Areas page. You can find definitions for the fields and controls later on this page.



Subject Area Detail

Enter a subject name and description and click Apply Subject Name. The subject area name does not need to be unique.

Click Show Associated Terms to display the list of terms associated with this subject area node. Order the terms appearing for a subject area node by specifying display order numbers. The order depicts how the terms are displayed for a subject area node in the term list presented in the condition builder.

Subject Area Tree

The subject area tree displayed on the left-hand side has the following icons:

- *Expand All.* Expands all subject area nodes.
- *Collapse All.* Collapses all subject area nodes.
- *Add Sibling.* Adds a new sibling node under the subject area node currently selected. Enter a subject area name and description on the right-hand pane and click Apply Subject Name to apply changes.
- *Add Child.* Adds a new child node for the subject area node currently selected. Enter a subject area name and description on the right-hand side and click Apply Subject Name to apply changes.
- *Delete Node.* Deletes the selected node.
- *Cut.* Cuts the currently selected node.
- *Paste as Sibling.* Pastes the node previously cut as a sibling of the currently selected node.
- *Paste as Child.* Pastes the node previously cut as a child of the currently selected node.

Registering Operators and Operator Sets

This section discusses how to register operators and operator sets.

Operators within PeopleSoft Active Analytics Framework are defined in text expressions that are used to evaluate a condition at runtime. The operand values in the text expressions must be substituted for an operator definition for integration with policies. The substitution of operand values is ensured by the correct placement of metatext corresponding to the desired operand or source term.

To refer to the left-hand side of a condition, use *%0* as the value. To refer to right-hand-side operands 1-4 , use the text values *%1, %2, %3,* and *%4,* respectively. For example, the following operator expression determines whether the numeric term's value is less than a right-hand-side value:

```
%0 < %1
```

Note: If you edit operators, policies that reference them must be recompiled. Do this by editing and saving the policy.

Pages Used to Register Operators and Operator Sets

Page Name	*Definition Name*	*Usage*
Register Operators Page	EOCF_OPERATOR_DEFN	Define operators.

Page Name	Definition Name	Usage
Register Operator Sets Page	EOCF_OPERSET_DEFN	Define operator sets.

Register Operators Page

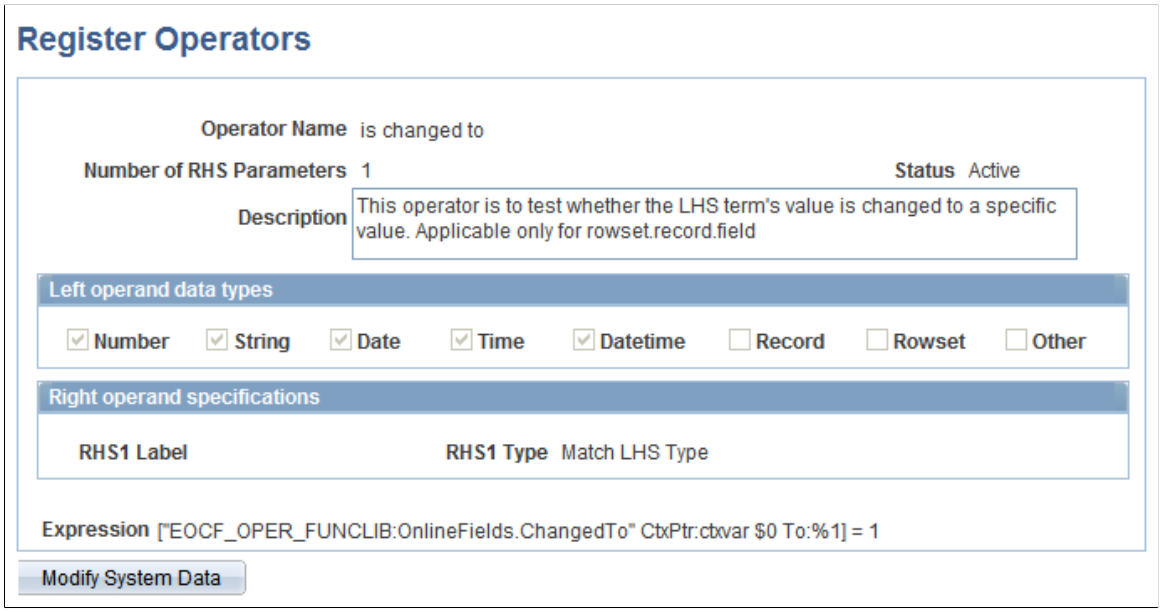
Use the Register Operators page (EOCF_OPERATOR_DEFN) to define operators.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Register Operators

Image: Register Operators page

This example illustrates the fields and controls on the Register Operators page. You can find definitions for the fields and controls later on this page.



Register custom operators by defining the left-hand-side and right-hand-side operand specifications and entering expression text.

Operator Name

Enter the name of the operator you are defining. Operator names do not need to be unique as long as the expression texts are different.

Number of RHS (right-hand side) Parameters

Specify the number of right-hand-side parameters that are required to evaluate the Boolean value for this operator. The condition builder uses this to display one or more right-hand-side fields to enter values.

Note: In this release, the condition builder interface supports a maximum of two right-hand-side parameters for an operator.

Left operand data types	Specify the supported data types for the left-hand-side operand of this operator. This data is used to constrain the list of operators in the condition builder.
Right operand specifications	Specify the type for each of the right-hand-side parameters. • *Match LHS Type.* Supports the left-hand operand data types. • *Fixed Type.* Specify a fixed data type for this parameter. • *Multi-Select.* Enables selection of multiple values for this parameter at condition building time.
Expression	Specify the technical expression defining how the operator works.

Register Operator Sets Page

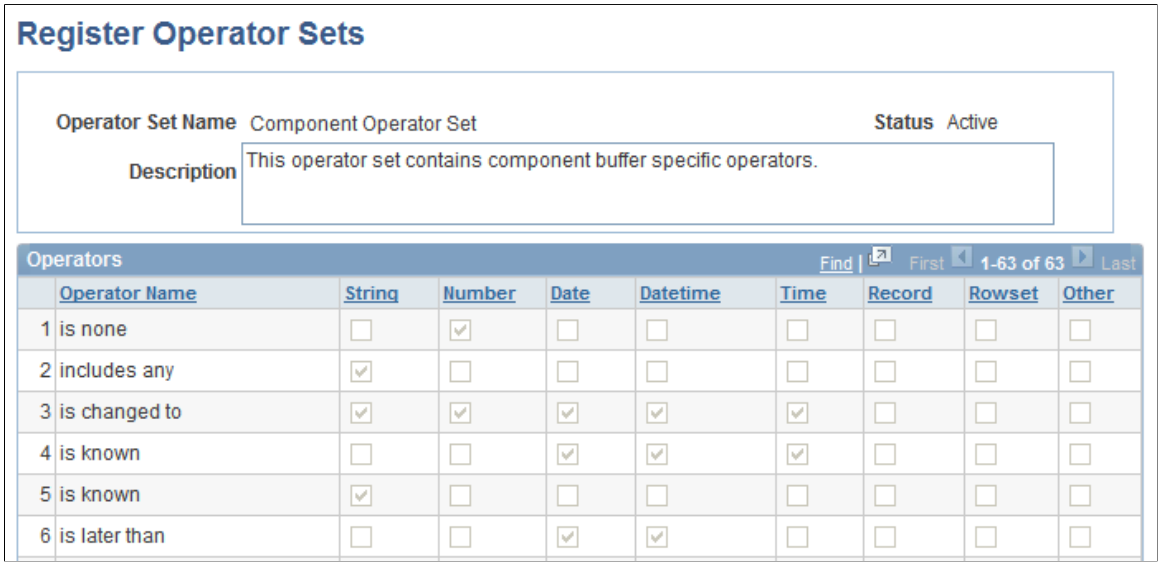
Use the Register Operator Sets page (EOCF_OPERSET_DEFN) to define operator sets.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Register Operator Sets

Image: Register Operator Sets page

This example illustrates the fields and controls on the Register Operator Sets page. You can find definitions for the fields and controls later on this page.



Operator sets define the list of valid operators for a term selected on the left-hand side of a condition.

Enter a unique operator set name and description, and add valid operators for this set.

Registering Resolution Methods

This section discusses how to register a resolution method.

Page Used to Register Resolution Methods

Page Name	*Definition Name*	*Usage*
Register Method Page	EOCF_DEF_RESLMTHD	Define the resolution method.

Register Method Page

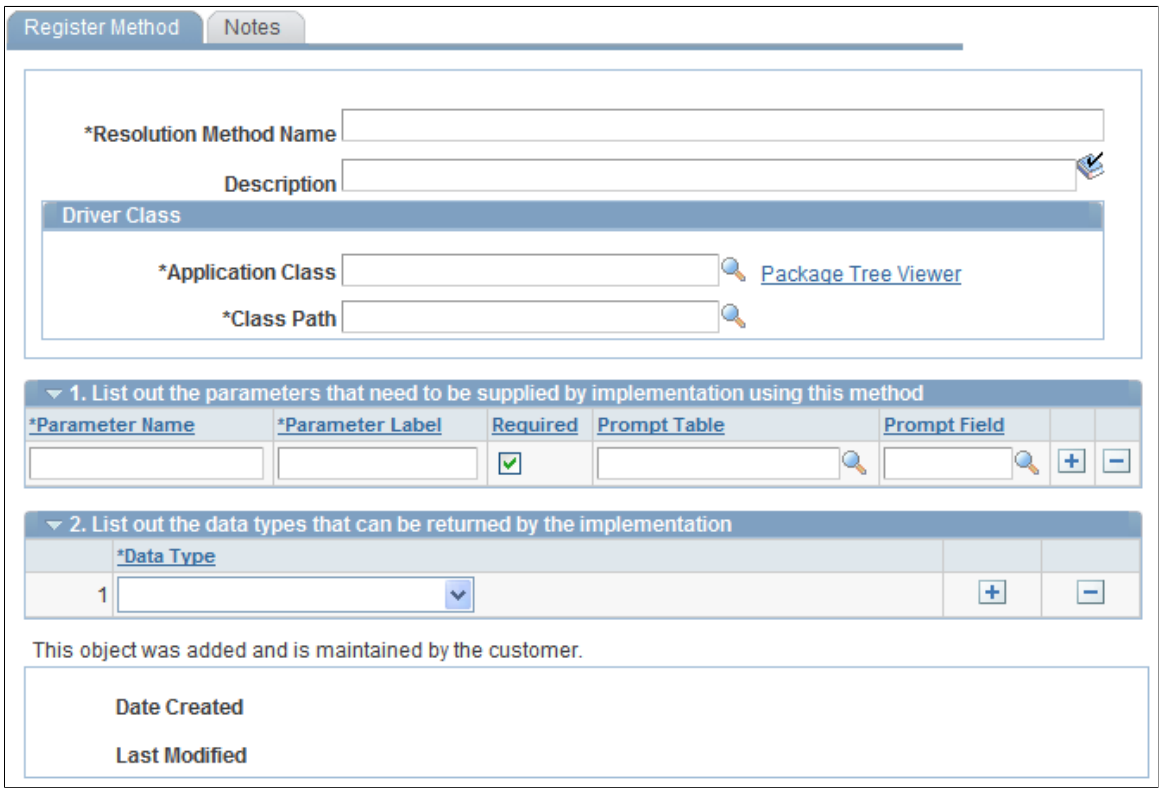
Use the Register Method page (EOCF_DEF_RESLMTHD) to define the resolution method.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Register Resolution Method

Image: Register Method page

This example illustrates the fields and controls on the Register Method page. You can find definitions for the fields and controls later on this page.



Resolution methods are used to build implementations. Oracle supports the following resolution methods:

- Application class
- PS Query

- SQL Object
- Record.Field

Resolution Method Name	The unique name of the resolution method.
Driver Class	The application class that encapsulates the data retrieval mechanism for this resolution method.
List out the parameters that need to be supplied by implementation using this method	Lists the parameters that an implementation using this method needs to supply.
List out the data types that can be returned by the implementation	Lists the valid data types that can be returned by an implementation using this resolution method.

Registering Business Domains and Categories

This section discusses how to register business domains and categories.

Pages Used to Register Business Domains and Categories

Page Name	*Definition Name*	*Usage*
Register Business Domain Page	EOCF_BUS_DOMAIN	Define a business domain, which is used to functionally classify trigger points.
Register Category Page	EOCF_IACATEGORY	Use to functionally classify policies.

Register Business Domain Page

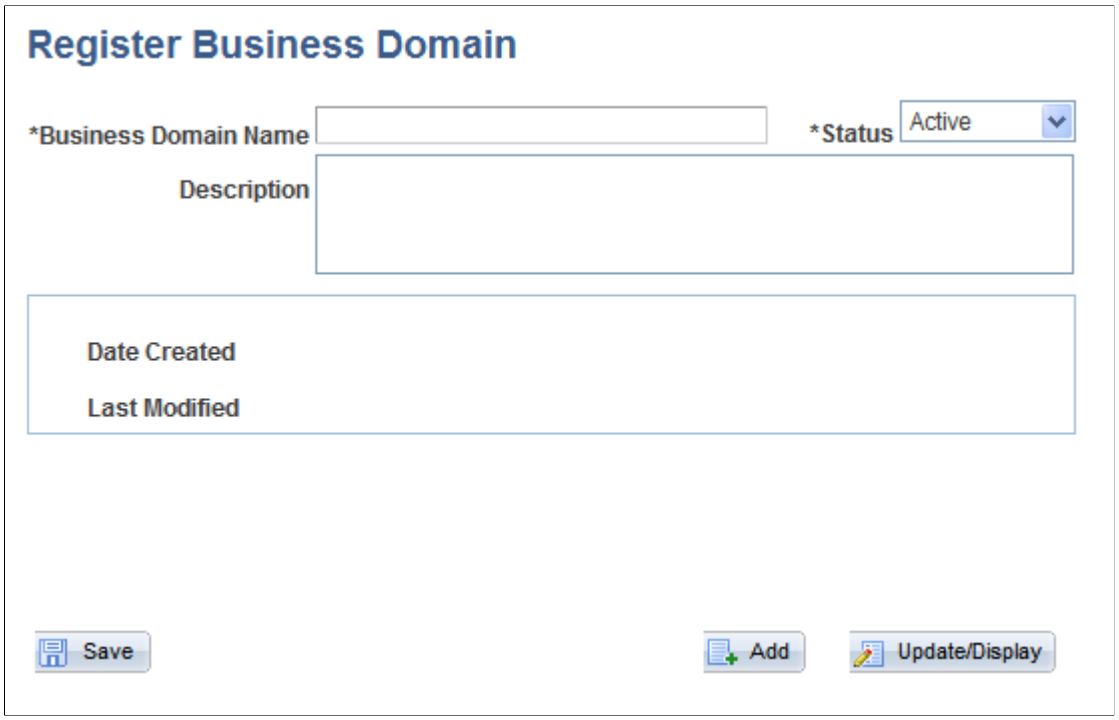
Use the Register Business Domain page (EOCF_BUS_DOMAIN) to define a business domain, which is used to functionally classify trigger points.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Register Business Domain

Image: Register Business Domain page

This example illustrates the fields and controls on the Register Business Domain page. You can find definitions for the fields and controls later on this page.



Enter the business domain name and description and select the appropriate status.

Register Category Page

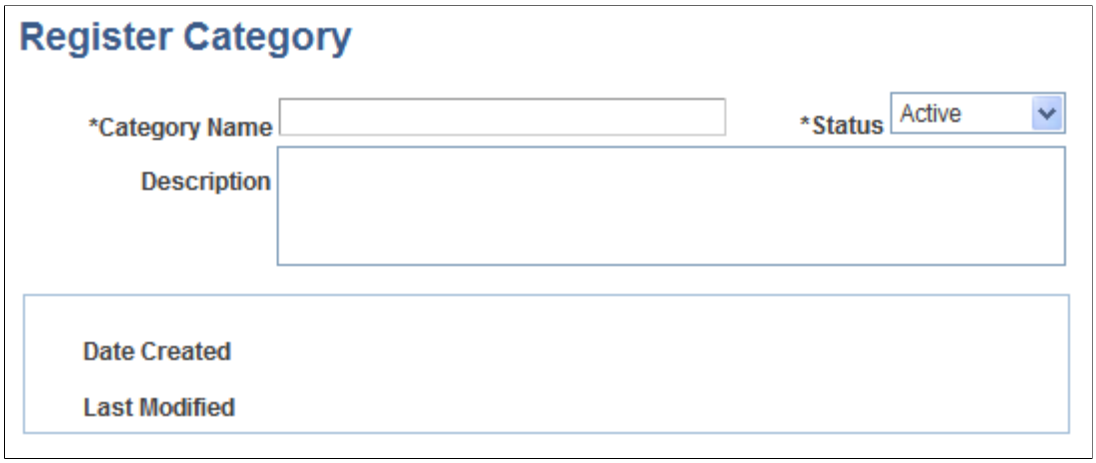
Use the Register Category page (EOCF_IACATEGORY) to use to functionally classify policies.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Register Category

Image: Register Category page

This example illustrates the fields and controls on the Register Category page. You can find definitions for the fields and controls later on this page.



Enter the category name and description and select the appropriate status.

Maintaining Cache

This section provides an overview of cache maintenance.

Pages Used to Maintain Cache

Page Name	*Definition Name*	*Usage*
Maintain Cache Page	EOCF_CLR_CACHE	Run the cache maintenance AE process.
Results Page	EOCF_CLR_RESULTS	Review the cache maintenance process result.

Understanding Cache Maintenance

To improve the overall performance of transactions, AAF implements rowset cache, which is used for populating the information on AAF objects, such as terms, implementations, contexts, subject areas and so on. Data caching occurs when AAF objects are added and updated. However, with caching in place, sometimes the system may not pick up all updates that are made to existing objects.

To avoid missing updates, AAF provides an application engine (AE) process that, when run, clears rowset cache.

Maintain Cache Page

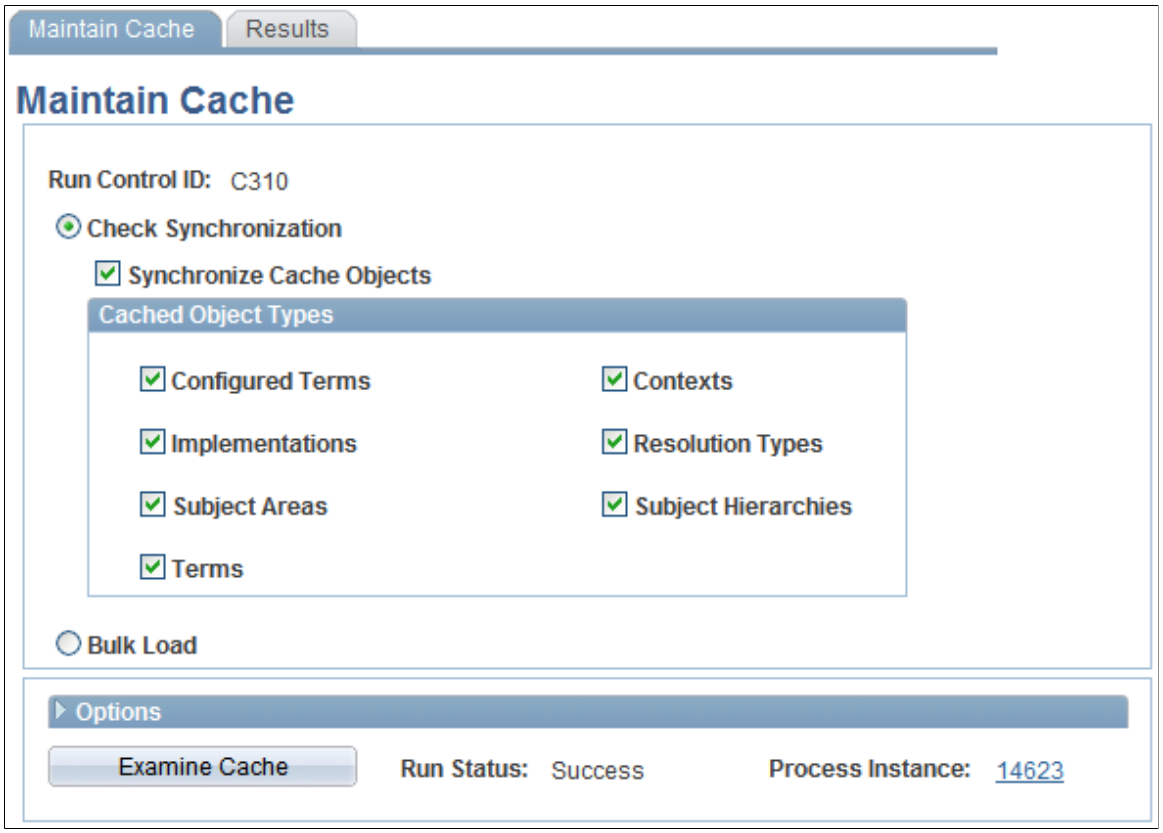
Use the Maintain Cache page (EOCF_CLR_CACHE) to run the cache maintenance AE process.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Maintain Cache

Image: Maintain Cache page

This example illustrates the fields and controls on the Maintain Cache page. You can find definitions for the fields and controls later on this page.



Cache maintenance is a two-step process. First, perform a synchronization check on the types of cached objects you selected. When the process instance completes successfully, load the cached objects. When the loading process is also completed successfully, reboot the application server and web server, clear all caches, and check to make sure that modified AAF objects have in fact been updated in the system.

Note: You need to sign into the PeopleSoft system in the same language that was used to run the cache maintenance process to view any AAF object updates that occurred as a result of the process.

Check Synchronization	Select to run a synchronization check on the types of cached objects that are selected in the Cached Object Types group box. This is the first part of the cache maintenance process.
Bulk Load	Select to load the selected types of cached objects to the system. This is the second part of the cache maintenance process.
Examine Cache	Click to start the cache maintenance (EOCF_DCACHE) process (for either *Check Synchronization* or *Bulk Load*) in Process Scheduler. Both the link and the current status of the actual process instance are available on this page.

Options

If you wish to run the process periodically, select the applicable server and frequency for its run schedule.

Results Page

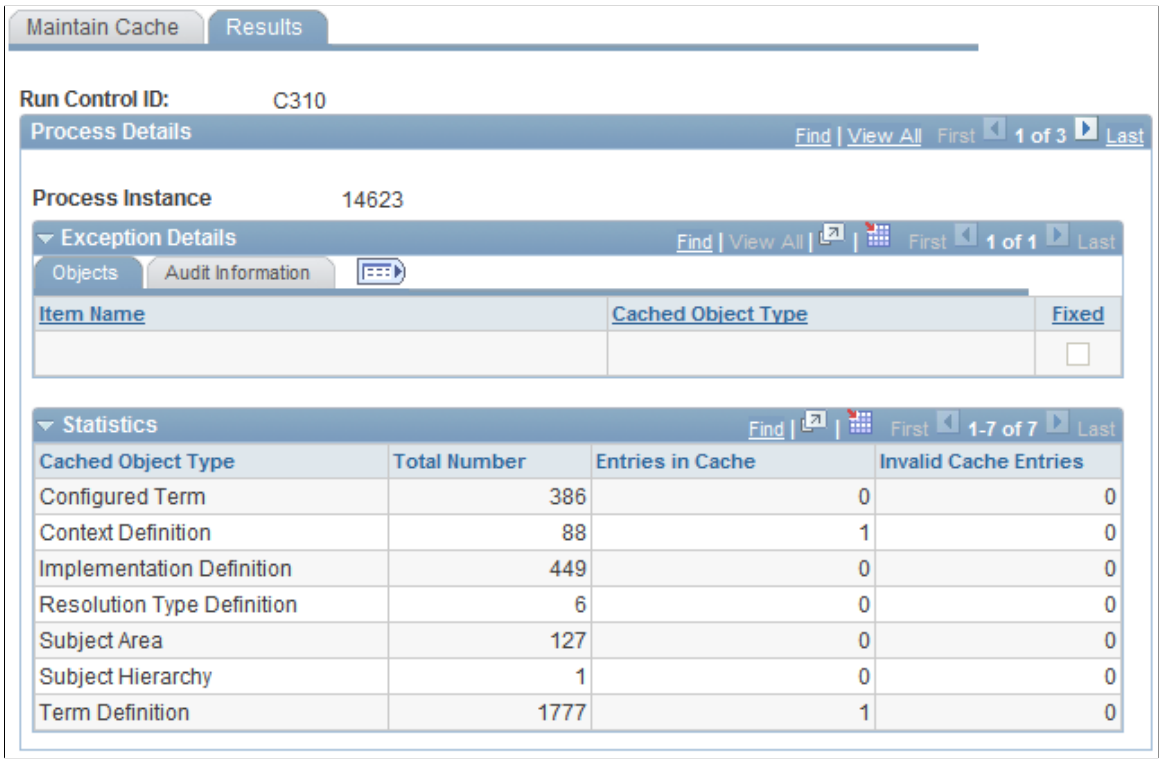
Use the Results page (EOCF_CLR_RESULTS) to review the cache maintenance process result.

Navigation

Enterprise Components > Active Analytics Framework > Setup > Maintain Cache

Image: Results page

This example illustrates the fields and controls on the Results page. You can find definitions for the fields and controls later on this page.



Use this page to review the EOCF_DCACHE AE process result. The counts or status of fixed cache objects is stored in the EOCF_CLR_STATS table.

Chapter 7

Considerations for Enabling the Framework

Considerations for Creating Contexts

To create an online context, you must be able to specify a component interface. You may encounter situations in which sets of components share the same base records and display the same information on-screen, but for different purposes. In this case, decide whether you need to create two contexts or whether you can use one.

Whether to Create One or Two Contexts

Determine whether the common terms for the two contexts should be registered as the same term objects or as different term objects. When common terms have the same meaning, the most economical practice is to reuse the common term definitions for both contexts.

Create two contexts if:

- The components have actionable fields that they do not share in common.
- One or the other of the components needs significant customization.

Considerations for Exposing Component Buffer Fields as Terms

Carefully consider which component buffer fields are to be made available as terms.

- Remember the big picture—select terms that can be reused at some point.

  However, you do not want to expose every component field as a term. Some of the transactional components that enable users to perform complex tasks may have component fields that are used as work fields to implement the component, but which do not represent functional data. Exposing these fields as terms could overpopulate the data library.

- Oracle recommends that you expose terms of those data elements on the component that are displayed to the end user and the elements that make up the component's search keys.
- Work-fields may contain valuable data and may be exposed as terms, but if these fields have not yet been populated by the time that the values from the corresponding terms are requested, unpredictable results may occur. If you know for certain that the work-fields will contain data at the appropriate times, such as in the example of a work-field for which data is computed as part of component prebuild, then exposing the data element as a term is not harmful. Although a Manage Context component is available for adding terms to a context, configuring terms is easier in the Generate Context component.

Considerations for Naming Terms

Note the following guidelines when you name terms:

- The framework prefixes each alias with the name of the context to reduce the creation of duplicate names.
- In record and rowset objects, the default term name is the name of the page buffer scroll, which will not be meaningful for the end user.

  Oracle recommends this name be changed to something that is meaningful for the user.
- Generally, rowset object names should be plural and record object term names should be singular.

  For example, if a component's scroll-level 1 contained purchase order line items, the rowset object is named Detail Items and the record object is named Line Item.
- If any terms are filed under alternate subject areas and you want your alternate heading to override, use the Overridden Subject Area prompt.

  Oracle recommends that you carefully consider whether all the names generated are meaningful before importing the context. Whenever context terms have the same functional meaning, have the context refer to preexisting terms rather than creating new ones.

Considerations for Creating Custom Operator Expressions

Each operator definition contains an *operator expression,* a text template that defines the meaning of the operator and how operands are integrated into the expression. All operator expressions must be Boolean expressions.

Comparisons supported in operator expressions are =, <. >, <=. >=, <> (not equal). All comparisons are type-driven; for example, a number cannot be compared to a string.

If type conversion is necessary, the value to be converted may be preceded by the token's string, number, date, datetime, or time, as needed.

Values are placed in expression text with two types of substitution tokens, location number and ? (question mark).

Location Number As an Operand

Location number determines the operand in the expression that is being referred to. For the left-hand-side term, the location number is zero; for the right-hand-side elements, the range is from one to four, depending on which right-hand-side parameter is being referred to. Usually, the location number is preceded by a percent sign. The percent sign and location number combination form a complete operand token.

A complete operand token refers to the value of the configured operand. When you create a condition, an expression-text entry is substituted for the operand token. This expression-text entry corresponds to the definition of the operand value's source.

Sometimes a term ID value needs to be provided in the expression instead of a location number. In this rare case, instead of a percent sign, a dollar sign is used to precede the location number.

Parentheses around subexpressions are allowed, so long as each left parenthesis has a corresponding right parenthesis, and so long as the subexpressions denoted within parentheses are valid. (While “(%0) > %1”

and “(%0 > %1) are valid, “(%0 >) %1” is not.) Constants are available for the current date, date-time, and current time by means of the items “%today”, “%timestamp” and “%now”.

Use of Term Code

In addition to referring to the results of terms that have been configured in the Operator Definition page, references to terms can be embedded directly in the expression text by means of the term code. A reference to a term is made by placing the term code name (which may not contain spaces) within square brackets. The bind parameters needed for the term must be part of the context; therefore, the operator should be part of a restricted operator set that is available only where the operator will be known to be valid.

You can also manually specify parameters for a term by following the term code name with a parameter specification, which consists of a name, a colon symbol, and the value to be bound to the parameter.

In the following example, a term is used as an input to another term in a condition to detect whether a customer's car is out of warranty:

[WarrantyExpirationDate VIN:[PrimaryVehicle DriversLicense:%0]] < %today

Generally, this type of construction is awkward and difficult to maintain, but it is available if needed. A more maintenance-friendly facility that doesn't require the configuration overhead of a term definition is a member-function call on an application class.

Two rules must be satisfied to use this facility:

- The class must not require any constructor arguments.
- The method to be invoked must accept an *array of any* as its only parameter.

To use this facility, set up an operator expression as if you are referring to a term; instead of using a term code name, use a quoted string value giving the fully qualified name of the class to be instantiated followed by a period and the name of the member function to be invoked. For example:

["MyAppPackage:MySubPackage:MyClass.MyMethod" MyParm1:%0 MyParm2:%2] = %1

Should an application class method need to return a Boolean result, the standard practice is to return a number, and to compare the number to 0 for false values or to 1 for true values.

Note: Be aware that when you create operators for a specific purpose, they are neither valid nor relevant except in a specific situation. Therefore, the operator should belong to an operator set that corresponds to the context or term for which the custom operator makes sense.

Considerations for Creating a Trigger Point

Use caution when introducing new trigger points, especially in the Field Change PeopleTools event. These trigger points, depending on where they are set, could result in significant overhead to the system.

In addition, introducing a new trigger point increases the likelihood of adding policies. These policies, depending on the number of terms and the mode of retrieval, could increase system overhead. This can result in an unfavorable user experience and throughput.

Oracle recommends that Field Change event trigger points be introduced only when it is critical. Consider deferring the execution of policies to a Save event.

Considerations for Enabling a Trigger Point in a PeopleCode Event

The events in which a trigger point is to be executed must include PeopleCode. The following is a minimal example:

```
Declare Function GetRuleService PeopleCode FUNCLIB_EOCF.EOCF_DE_PUBLIC FieldFormula⇒
; GetRuleService().reset.SET_CONTROL = &mySetControlValue;
GetRule Service.fireEvent(My_Trigger_Point_Code_Name);
```

Note: The use of the reset property on the first line is critical to the framework's correct functioning. If reset is not used, the state of the RuleService (the runtime API for triggering the decision engine) is unknown from previously executed PeopleCode.

Considerations for Enabling the Display Action in a PeopleSoft Application Page

To enable the provided display alert action type:

- Add the PeopleTools subpage EOCF_DISPLAY_SUBPG at Level 0 to each page within the component.
- Ensure that no other fields overlap the read-only display alert subpage; otherwise, the display alert action will not work correctly.
- Ensure that the roles having access to the transactional component that requests the framework to evaluate policies have access to the EOCF9002 permission list.
- Ensure that all pop-up blockers have been disabled.

Note: Any display action types created must be registered in the display action type bundle in addition to the delivered display alert action type.

Considerations for Creating a New Action Type

When creating a new action type, consider:

- The location of the code that handles the design-time and runtime aspects of the action type.
- The configuration requirements of the new action type before being enabled.
- Whether the action type is a terminal action.

For example, the action is to transfer to another page terminates the framework processing. If several terminal actions are associated with a trigger point, only the first of the terminal actions is chosen, and it fires after all other nonterminal actions have been fired.

Configuring the Action Type

If your action type requires a configuration page:

- Create record definitions, keyed by EOCF_ACTION_ID, to capture design time configuration data.
- Create an action configuration component and page.
- Insert EOCF_ACTN_CFG_SBPG in the configuration page.

  This displays the action type and action name on the configuration page.
- Insert the policy information subpage, EOCF_RULEINFO_SBP, on the page.

  This displays policy information, including the policy name and condition text.
- If you need to access terms, put a clone of EOCF_SRCH_DSP_AL as a secondary page on your action configuration page, and add the following code in the page's activate code.

  See the display alert action configuration page, EOCF_DSPL_ALRT_CFG, as an example of a configuration page.

```
import EOCF_CLF_RB:Definition:Rule:Rule;
import EOCF_CLF_RB:UI:*;

Component SearchBuilder &cobjSearchBuilder;
Component SearchConfig &cobjSearchConfig;
Global Rule &gobjRule;
Local RuleBuilder &objRuleBuilder = create RuleBuilder();
 &cobjSearchConfig = create EOCF_CLF_RB:UI:SearchConfig();

/*** Build Search page display **/
/*** Add where clause to filter terms by scalar **/
&cobjSearchConfig.AddCustomWhereClause(FetchSQL(SQL.EOCF_WHERECL_TMSCALAR), 0,
   "");
&cobjSearchBuilder = create SearchBuilder(EOCF_ACTION_WRK.EOCF_CONTEXT_ID,
   &cobj SearchConfig);
&cobjSearchBuilder.BuildDisplay();
/*** Populate Rule info **/
&objRuleBuilder.PopulateRuleInfo(&gobjRule);
```

- Insert a push button on the configuration page to transfer to this secondary page (for the purpose of choosing a term).

  Some of the terms on your configuration page may be configurable. Therefore, you must paste another subpage, EOCF_TERMCFG_SBP, on the configuration page for this purpose.
- Insert the following page activate code on your action configuration page to initialize the term configuration subpage:

```
import EOCF_CLF_RB:Definition:Rule:Rule
import EOCF_CLF_RB:UI:*;
import EOCF_CLF_RB:UI:ConfigBuilder;
import EOCF_CLF_RB:Definition:RuleTermConfig;
import EOCF_CLF_DL:Factory:DataLibraryFactory;
import EOCF_CLF_DL:Definition:Term:TermDefn;
```



```
import EOCF_CLF_DL:Utility:DLConstants;
Local Rowset &rs_level0, &rsActionTypeDefn;
Local number &bundleId, &actionId;
Local number &actionTypeId;
Local string &isTerminal, &isCombinable, &isConfigurable;
Local string &aeTriggered, &appMsgTriggered, &ciTriggered, &piaTriggered;
Local string &commit, &path, &id, &dtAppClassId, &appClsPath, &dtAppClsPath,
   &rtApp ClassId, &rtAppClsPath, &actTypeName, &actionName;
Global EOCF_CLF_RB:Definition:Rule:Rule &gobjRule;
Local EOCF_CLF_RB:UI:RuleBuilder &objRuleBuilder = create EOCF_CLF_RB:UI:
   Rule Builder();
Local EOCF_CLF_RB:Definition:RuleTermConfig &objRuleTermConfig = create
   EOCF_CLF_RB: Definition:RuleTermConfig();
Local EOCF_CLF_DL:Factory:DataLibraryFactory &objDlFactory = create
   EOCF_CLF_DL: Factory:DataLibraryFactory();
Local EOCF_CLF_DL:Definition:Term:TermDefn &objTermDefn = create
   EOCF_CLF_DL: Definition:Term:TermDefn();
Local EOCF_CLF_DL:Utility:DLConstants &objDLConstants = create EOCF_CLF_DL:
   Utility: DLConstants();
/*********************************************************************/
/* CREATE A CONFIG BUILDER TO CREATE UI ELEMENTS THAT ENABLE TERM    */
/* CONFIGURATION-TO BE DONE ONLY BY ACTIONS THAT NEED TO ACCESS DATA */
/* LIBRARY TERMS, AND THUS HAVE THE TERM-PICKER SCROLL AND THE TERM  */
/* CONFIGURATION SUBPAGE ON THE PAGE */

Local ConfigBuilder &objConfigBuilder;
Local Rowset &rs1 = GetLevel0()(1).GetRowset(Scroll.EOCF_DS_ALT_TRM);
&objConfigBuilder = create ConfigBuilder();
&objConfigBuilder.InitDisplay();
For &i = 1 To &rs1.ActiveRowCount
   &termId = &rs1(&i).EOCF_DS_ALT_TRM.EOCF_LIB_TERM_ID.Value;
   If All(&termId) Then
      If All(&rs1(&i).EOCF_DS_ALT_TRM.EOCF_CONFIG_ID.Value) Then
         /*****************************/
         &objRuleTermConfig = create EOCF_CLF_RB:Definition:RuleTermConfig();
         /*****************************/
         &objRuleTermConfig.EOCF_CONFIG_ID = &rs1(&i).EOCF_DS_ALT_TRM.
            EOCF_CONFIG_ ID.Value;
         &objRuleTermConfig.getConfiguredTerm( True);
         &rs1(&i).EOCF_DS_AL2_WRK.EOCF_GOTO_BTN.Label = &objRuleTermConfig.
            Get ConfigTermLabel(EOCF_ACTION_WRK.SETID.Value);
         &rs1(&i).EOCF_DS_AL2_WRK.EOCF_GOTO_BTN.Enabled = True;
      Else
         &objTermDefn = &objDlFactory.getTermDefn(&termId, False,
            &obj DLConstants.ALLOCATIONTYPE_READONLY);
         &rs1(&i).EOCF_DS_AL2_WRK.EOCF_GOTO_BTN.Label = &objTermDefn.
            EOCF_TERM_ LABEL;
         &rs1(&i).EOCF_DS_AL2_WRK.EOCF_GOTO_BTN.Enabled = False;
      End-If;
   Else
/*************/
&rs1(&i).EOCF_DS_AL2_WRK.EOCF_GOTO_BTN.Label = " ";
 &rs1(&i).EOCF_DS_AL2_WRK.EOCF_GOTO_BTN.Enabled = False;
   End-If;
End-For;

Local Grid &TERMGRID;
Local GridColumn &TERMCOLUMN, &LOOKUPCOLUMN;

&TERMGRID = GetGrid(Page.EOCF_DSPL_ALRT_CFG, "EOCF_DS_ALT_TRM");
&TERMCOLUMN = &TERMGRID.GetColumn("EOCF_GOTO_BTN");
&TERMCOLUMN.Label = " ";
&LOOKUPCOLUMN = &TERMGRID.GetColumn("EOCF_CONFIGURE");
&LOOKUPCOLUMN.Label = MsgGetText(18112, 2541, "Message not found - Get Term");
/*********************************************************************/
/*POPULATE RULE INFO - NEEDS TO BE DONE BY ALL CLF ACTIONS, i.e.,    */
/*ACTIONS WHOSE CONFIG PAGES HAVE BEEN NAVIGATED TO VIA RULE BUILDER */

&objRuleBuilder.PopulateRuleInfo(&gobjRule);
/*********************************************************************/
/* GET THIS ACTION'S ACTION TYPE NAME - TO BE DONE BY ALL ACTIONS */
```

```
&rs_level0 = GetLevel0();
&actionId = &rs_level0(1).EOCF_ACTION_WRK.EOCF_ACTION_ID.Value;
&actionTypeId = &rs_level0(1).EOCF_ACTION_WRK.EOCF_ACTION_TYP_ID.Value;
&actionName = &rs_level0(1).EOCF_ACTION_WRK.EOCF_ACTION_NAME.Value;

/* Get details of Action Type */
rem SQLExec(SQL.EOCF_ACTTYP_DTLS2_SEL, &actionTypeId, &actTypeName,
   &dtAppClassId, &dtAppClsPath, &rtAppClassId, &rtAppClsPath, &isTerminal,
   &isConfigurable, &is Combinable, &aeTriggered, &appMsgTriggered,
   &ciTriggered, &piaTriggered, &Commit Flag, &dtCommit, &descr);

/***** Get details of Action Type ***/
/**** Modified by SB to enable rel. language processing for action type name *⇒
**/

&rsActionTypeDefn = CreateRowset(Record.EOCF_ACTN_TYPE);
&numrows = &rsActionTypeDefn.Fill("WHERE EOCF_ACTION_TYP_ID = :1",
   &actionTypeId);

If &numrows > 0 Then
   &actTypeName = &rsActionTypeDefn(1).GetRecord(1).EOCF_ACT_TYP_NAME.Value;
   &dtAppClassId = &rsActionTypeDefn(1).GetRecord(1).EOCF_DT_APPCLASSID.Value;
   &dtAppClsPath = &rsActionTypeDefn(1).GetRecord(1).EOCF_DT_APPCLSPATH.Value;
   &rtAppClassId = &rsActionTypeDefn(1).GetRecord(1).EOCF_RT_APPCLASSID.Value;
   &rtAppClsPath = &rsActionTypeDefn(1).GetRecord(1).EOCF_RT_APPCLSPATH.Value;
   &isTerminal = &rsActionTypeDefn(1).GetRecord(1).EOCF_IS_TERMINAL.Value;
   &isConfigurable = &rsActionTypeDefn(1).GetRecord(1).EOCF_IS_CFG_FLAG.Value;
   &isCombinable = &rsActionTypeDefn(1).GetRecord(1).EOCF_IS_CMBIN_FLAG.Value;
   &aeTriggered = &rsActionTypeDefn(1).GetRecord(1).EOCF_AE_TRIGGERED.Value;
   &appMsgTriggered = &rsActionTypeDefn(1).GetRecord(1).EOCF_APPMSG_TRIGGR.Val⇒
ue;
   &ciTriggered = &rsActionTypeDefn(1).GetRecord(1).EOCF_CI_TRIGGERED.Value;
   &piaTriggered = &rsActionTypeDefn(1).GetRecord(1).EOCF_PIA_TRIGGERED.Value;
   &CommitFlag = &rsActionTypeDefn(1).GetRecord(1).EOCF_COMMIT_FLAG.Value;
   &dtCommit = &rsActionTypeDefn(1).GetRecord(1).EOCF_DT_COMMIT.Value;
   &descr = &rsActionTypeDefn(1).GetRecord(1).DESCR.Value;
End-If;

/* Populate the ActionTypeName field for display */
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_ACT_TYP_NAME.Value = &actTypeName;

/*****************************************************************/
/* PLEASE DO THE FOLOWING IF YOU NEED THE DETAILS OF THE ACTION TYPE */
/* THAT WERE ENTERED AT THE TIME OF ACTION TYPE Registration*/
/*  This Display Alert action Does not need these details*/
/*
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_DT_APPCLASSID.Value = &dtAppClassId;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_DT_APPCLSPATH.Value = &dtAppClsPath;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_RT_APPCLASSID.Value = &rtAppClassId;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_RT_APPCLSPATH.Value = &rtAppClsPath;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_IS_CFG_FLAG.Value = &isConfigurable;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_IS_CMBIN_FLAG.Value = &isCombinable;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_IS_TERMINAL.Value = &isTerminal;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_APPMSG_TRIGGR.Value = &appMsgTriggered;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_CI_TRIGGERED.Value = &ciTriggered;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_AE_TRIGGERED.Value = &aeTriggered;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_PIA_TRIGGERED.Value = &piaTriggered;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_COMMIT_FLAG.Value = &CommitFlag;
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_DT_COMMIT.Value = &dtCommit;
&rs_level0(1).EOCF_DSTIME_WRK.DESCR.Value = &descr;
*/
/*****************************************************************/
/* PLEASE DO THE FOLLOWING IF  THE ACTION IS A COMBINABLE ONE AND YOU */
/* NEED THE INFO REGARDING THE ACTION BUNDLE OF WHICH IT IS A PART    */
/* This Display Alert Action , though combinable, does not need this  */
/*
SQLExec(SQL.EOCF_ACT_BUNDLE_SEL, &actionTypeId, &ActnBundleId);
&rs_level0(1).EOCF_DSTIME_WRK.EOCF_ACT_BUNDLE_ID.Value = &ActnBundleId;*/
/*****************************************************************/
```

Creating Design and Runtime Application Class Code

To create the design-time and runtime application class code:

1. Create an application package named <*your product code*>*_AAF_ACTIONS*.
2. Create a class in the package to be your design-time action application class.

   It must extend the EOCF_CLF_AF: ActionCfgBase and must implement the designAction() method.

   This class is responsible for transferring from the policy builder page to the action configuration page previously created.

   All relevant information regarding the rule and the context will be available to the configuration page.

   (See the EOCF_CLF_ACTIONS: DisplayAlertCfg as an example.)
3. Create another class in the package to be your runtime application class.

   This class must extend EOCF_CLF_AF:ActionBase and must implement the runAction() method.

   This class is responsible for launching the action.

   In order to perform the action, all of the relevant information will be available.

   (See the EOCF_CLF_ACTIONS: DisplayAlert as an example.)
4. Register the action type in the Register Action Type page.

   Specify the name of your new action type, the names of your design-time and runtime application classes, and other action type characteristics.

   Specify the trigger types and trigger points that may have policies using this action type.

   For example, you may want an action to be triggered during ComponentPostBuild trigger types, specifically the trigger point “When a Defect is Presented, in Quality Management.” After you complete these configuration steps, the action type is available for use.
5. If an action type is combinable with other action types, register it in an action type bundle in addition to the other combinable action types.

   All display action types must be included in a single action type bundle.
6. If your new action is configurable, click Configure on the page.

   This transfers you to the action configuration page for more specifications.
7. Insert the subpage EOCF_DISPLAY_SUBPG in Level 0 on all pages that you want display-enabled.

Note: This display subpage must not overlap any other field and no pop-up blockers should be running.

Considerations for Creating an Application Class Implementation

Oracle recommends that:

- Implementations of related terms be grouped in a single class in which different methods are responsible for deriving the term's value.

  This practice facilitates maintaining a manageable number of application classes.

- In situations in which the probability is high that multiple related terms are accessed during a single business event:

  - You have a single implementation return a rowset, record, or vector containing the data for as many of the terms as possible.

  - When defining the term, you specify which data element in the rowset, record, or vector is to be used for the term.

Technical Details of Application Class Implementations

The data library calls the appropriate application class based on the class and package information registered with the implementation. The application class is instantiated automatically and the specified application class method is invoked. The application class constructor must be coded to accept an instance of the AccessMethod class.

For example:

```
class LeadOppMetrics
 method LeadOppMetrics(&_oAccessMethod As EOCF_CLF_DL:Runtime:AccessMethods:
   Access Method);
 method LeadCountbyBO();
    private instance EOCF_CLF_DL:Runtime:AccessMethods:AccessMethod
       &ioAccess Method;
end-class;
```

The AccessMethod object enables the application class to access values for all the implementation binds (input parameters) that are needed by the implementation and all the information about the term being resolved.

Example of an Application Class Accessing Implementation Binds

The following partial code shows how an application class can access any of its implementation binds.

```
method GetCaseID Local any &CaseID;
  &CaseID = &ioAccessMethod.getBindValueByName("CASE_ID");
  Local EOCF_CLF_DL:Runtime:Results:ResultsScalar &oResultsScalar;
  &oResultsScalar = create EOCF_CLF_DL:Runtime:Results:ResultsScalar(&CaseID);
      &ioAccessMethod.Results = &oResultsScalar;
end-method;
```

The application class does not need to know how the data is retrieved and passed by the data library engine. The data could have been passed by the calling application, defined as a constant in context

definition, or defined as a term and resolved by the data library engine. The application class can retrieve the data either by position or by name.

```
/* BO_ID is the name of the implementation bind;the bind name specified
 here matches the one to be specified at the time of registering the
 implemenation in &nBOID = &ioAccessMethod.getBindValueByName("BO_ID");
 Here the request is made to retieve the value for the first bind. */
&nBOID = &ioAccessMethod.getBindValueByPosition(1);
```

Another way that the calling application can pass additional information to implementations or policy actions is to have the calling application use the addUserVariable method to add the information to the context object. To access this information, the application class uses the getUserVariable method.

For example:

```
&AdditionalData = &oAccessMethod.AMContext.getUserVariable("ExtraInfo);
```

Note: The shape of the data received by an application class using any of the previous methods will be a PeopleCode Any object. Therefore, the application class is responsible for converting the object to the appropriate data type before the value is consumed. If the application class has a method to be invoked by the data library engine, then that method should not require any parameters.

Sample Methods of an Application Class Resolving Terms

The following code is an example of an application class that has several methods for resolving terms.

```
class OperatorInfo
 method Get_Person_Name();
 method Get_Person_Salutation();
 method Get_Person_Title();
 method Get_Person_Gender();
 method Get_Person_BirthDate();
 method OperatorInfo(&_oAccessMethodParam As EOCF_CLF_DL:Runtime:
     AccessMethods: AccessMethod);
  private
   instance EOCF_CLF_DL:Runtime:AccessMethods:AccessMethod &ioAccessMethod;
end-class;
```

If the data library engine invokes an application class method, that method is responsible for deriving the results and for inserting the results in the AccessMethod object. The data library engine transmits the results to the calling application. If an application class method is not specified in the implementation definition, the constructor is responsible for performing these tasks.

The data library provides several mechanisms through which the application class can return the output value to the data library:

- The application class instantiates one of the following application classes with the output value that needs to be passed to the engine.
  - `ResultsRecord Record &oResultsRecord = create EOCF_CLF_DL:Runtime: Results:ResultsRecord(RecordVariable);`
  - `ResultsRowset Rowset &oResultsRowset = create EOCF_CLF_DL:Runtime: Results:ResultsRowset(RowsetVariable);`
  - `ResultsScalar Scalar &oResultsScalar = create EOCF_CLF_DL:Runtime: Results:ResultsScalar(ScalarVariable);`

- `ResultsVector Vector &oResultsVector = create EOCF_CLF_DL:Runtime: Results:ResultsVector(VectorVariable);`

- The type of class to be instantiated depends upon the type of data that needs to be returned.
- The instantiated object is assigned to the member of AccessMethod object.

Note: Application class objects are not cached. When the data library engine invokes the application class because the data is not available in the cache, the application class is instantiated by invoking the constructor, then the method specified in the implementation. Therefore, the instance variables created in the constructor cannot be shared across multiple methods of the same class to resolve different terms.

See Defining Term Properties.

Example of How a Value Can Be Passed to the Data Library

The following example is a partial code listing of how a value can be passed to the data library:

```
method GetCaseID
Local any &CaseID;
   &CaseID = &ioAccessMethod.getBindValueByName("CASE_ID");
/* use the appropriate sub class of Results class for assigning value */
Local EOCF_CLF_DL:Runtime:Results:ResultsScalar &oResultsScalar;
&oResultsScalar = create EOCF_CLF_DL:Runtime:Results:ResultsScalar(&CaseID);
&ioAccessMethod.Results = &oResultsScalar;end-method;
```

Sample Code of an Application Class Implementation

The following is an example of an application class implementation.

```
import EOCF_CLF_DL:Runtime:AccessMethods:*;
import EOCF_CLF_DL:Runtime:Resolution:*;
import EOCF_CLF_DL:Contexts:*;
import EOCF_CLF_DL:Runtime:Results:*;

class CaseRecordInformation
   method CaseRecordInformation(&_oAccessMethod As EOCF_CLF_DL:Runtime:
      Access Methods:AccessMethod);
   method GetResultRecord();

private
   instance EOCF_CLF_DL:Runtime:AccessMethods:AccessMethod &ioAccessMethod;
end-class;

method CaseRecordInformation
   /+ &_oAccessMethod as EOCF_CLF_DL:Runtime:AccessMethods:AccessMethod +/

   &ioAccessMethod = &_oAccessMethod;
   end-method;

method GetResultRecord
   Local Record &result;
   Local number &nResolved_value, &nCaseID;
   Local string &sBusUnit;

/* Retrieving the implementation bind using the API */
   &nCaseID = &ioAccessMethod.getBindValueByName("CASE_ID");

   SQLExec("SELECT BUSINESS_UNIT FROM PS_RC_CASE WHERE CASE_ID = :1",
      &nCaseID, &s BusUnit);
   &result = CreateRecord(Record.RC_CASE);
```



```
   &result.CASE_ID.Value = &nCaseID;
   &result.BUSINESS_UNIT.Value = &sBusUnit;
      &result.SelectByKey();
/* Passing the values (in this case, it is a record) back to the data
   library engine - this record could be used to resolve multiple terms */
   &ioAccessMethod.Results = (create EOCF_CLF_DL:Runtime:Results:
      ResultsRecord (&result));

end-method;
```

Considerations for Creating a PS Query Implementation

Oracle recommends that you perform these tasks when multiple related terms may be accessed during a single business event:

1. Create a single PS Query implementation to return a rowset containing the data for several terms.
2. Specify which data element or field position in the rowset or record is to be used for the term.
3. Ensure that appropriate privileges have been set for accessing the objects present in the query.

Note: Use caution when using this type of implementation because the query may execute very slowly.

Considerations for Creating a Policy

The PeopleSoft Active Analytics Framework is neither a programming tool nor a substitute for PeopleCode. Do not use the framework as a way to program policies that do not need to be configurable. If policies are static, the use of PeopleCode is the best method. The framework should not be used as a tool to configure the presentation layer: for dynamically hiding or unhiding fields, making the fields editable, and so on. Furthermore, the data library should not be used when no need exists for the data abstraction layer.

Use the PeopleSoft Active Analytics Framework when a need exists for business users to configure business processes with business rules, without having to customize the application.

Before building policies, consider that each term used in conditions or in actions within policies could negatively affect performance of the application component, especially if the term's retrieval mechanism is time consuming.

