CHAPTER 1

Bundle 29
New chapter.

Working with the Rules Engine

This chapter provides an overview of the Rules Engine Framework and discusses:

- Setting Up the Rules Engine
- Constructing Rules
- Using Statements for Evaluation and Calculation in a Rule
- Building and Testing Rules

Understanding the Rules Engine

This section discusses:

- Rules Engine components.
- A high level description of the Rules Engine.
- Rules Engine Manager and Entity Registry.

The Rules Engine provides:

- a non-programmer user interface to create complex business rules.
- a secure way to retrieve data from the system in a logical manner, perform calculations and evaluations, and update data.
- a way to use the Entity Registry, a familiar logical hierarchy, to retrieve data from the system; for example, the curriculum structure of the Academic Item Registry (AIR) or the results structure of the Academic Progress Tracker (APT).
- System Variables and Functions for creating Rules.
- a compiler (Rule Builder) that compiles and readies Rules for execution.
- a means to manage changes to Rules over time and a large number of Rules using versioning.

Rules Engine Components

There are three components used to manage the Rules Engine feature:

- Rules Engine Manager

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Rules Engine Manager is the interface for creating institution business rules. The Rules Engine Manager can be deployed for two Rule building Skill Levels, Expert and Developer, each with its own set of Rules-building capabilities. The Rules Engine Manager uses the Entity Registry by allowing users to build Rules using a familiar logical hierarchy; for example, the curriculum structure.

Note. Oracle plans to introduce additional Skill Levels in future releases.

- Rules Engine

  The Rules Engine builds (compiles) and executes the user-created business rules.

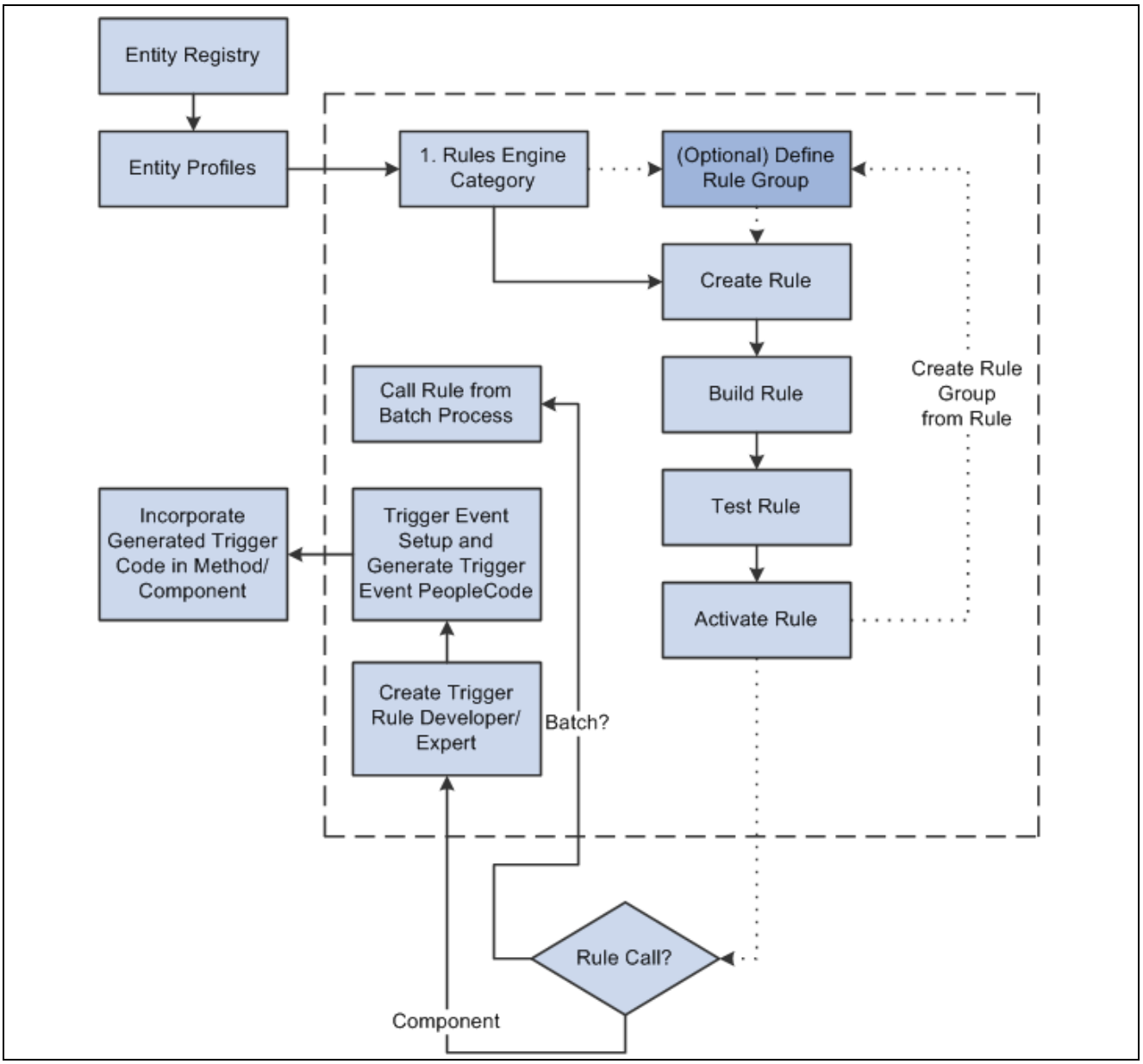
- Rules Engine Categories.

  Rules Engine Categories are used to restrict access to pre-defined and secured areas of application functionality including access to Rules in other Rules Engine Categories and what types of Rules can be created: Triggers, Functions or Rules.

High Level Description of the Rules Engine

This diagram shows Rules Function functionality:

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Rules Engine Business Process (Generic)

Rules Engine Manager and Entity Registry

This section discusses the relationship between the Rules Engine and the Entity Registry.

Entity Registry

When building Rules you need access to the data in your system. For Query Manager and Equation Engine, data access is based on granting the user access to specific records in the system.

However, data access for the Rules Engine is controlled through the Entity Registry. An Entity is an object that provides access (view, create, update) to data in a record. The properties on the Entity represent the fields in the actual records.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

By design, an Entity is the primary point of access to the underlying records. This avoids potential inconsistencies of having the same logic in multiple places and helps in making relevant code reusable and maintenance easier. This becomes relevant when the same data is accessed or updated in multiple ways: a user-interface, web services, or the Rules Engine.

Entity relationships are represented in a tree-like hierarchy, making the underlying data structure logical for functional users to understand. When building a Rule that needs to access specific system data, you first choose which Base Entity you want to with.

For more information about Entity Registry:

See [PeopleSoft Campus Community 9.0 Fundamentals PeopleBook]Setting Up Entity Registry.

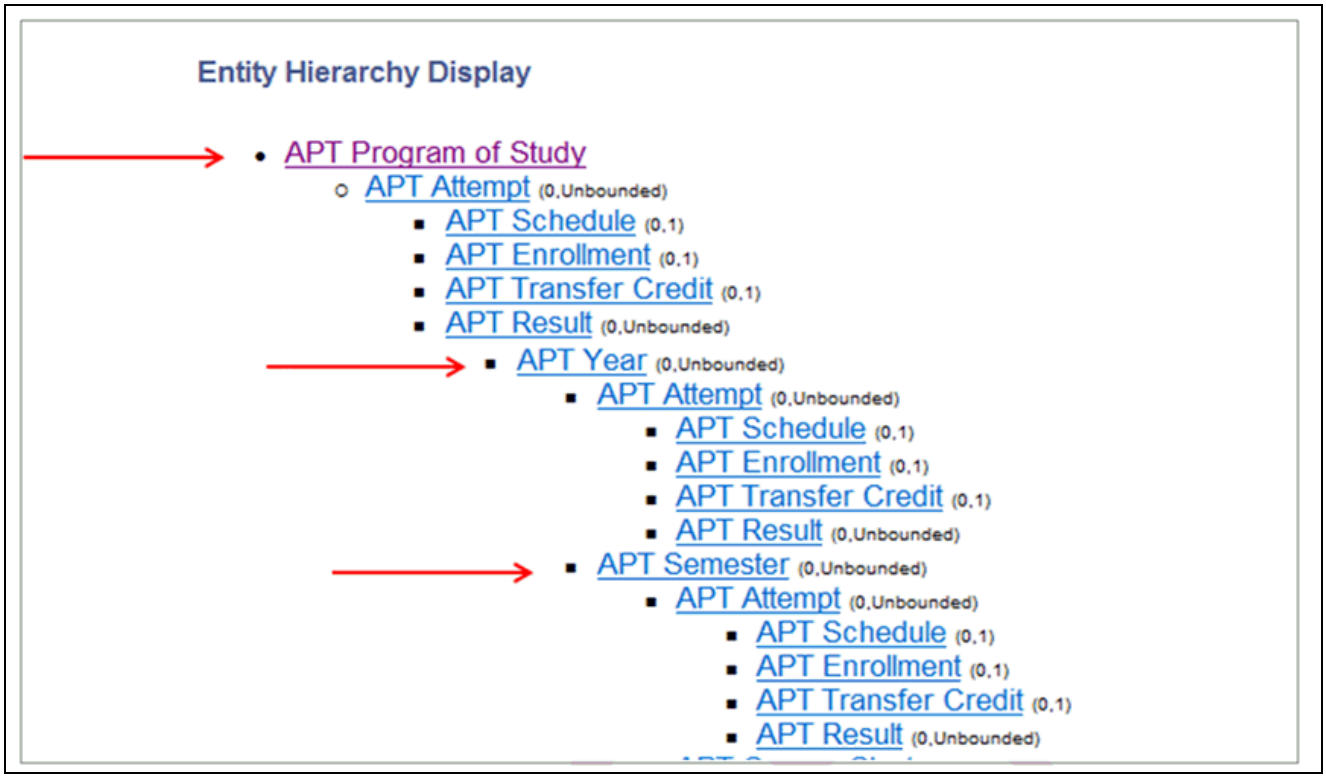
Note. Although the Rules Engine can be used with any Entity Registry delivered with the system, it is the Entity Registries created for data records from the Program Enrollment Academic Item Registry (AIR) and Academic Progress Tracker (APT) features that are optimized for Rules Engine use.

These Entities are used throughout this chapter as examples of how Entity-based data can be used in the Rules Engine.

For more information about AIR and APT:

See *PeopleSoft Student Records 9.0 PeopleBook, Setting Up Program Enrollment* and *PeopleSoft Student Records 9.0 PeopleBook, Managing Program Planning and Enrollment*.

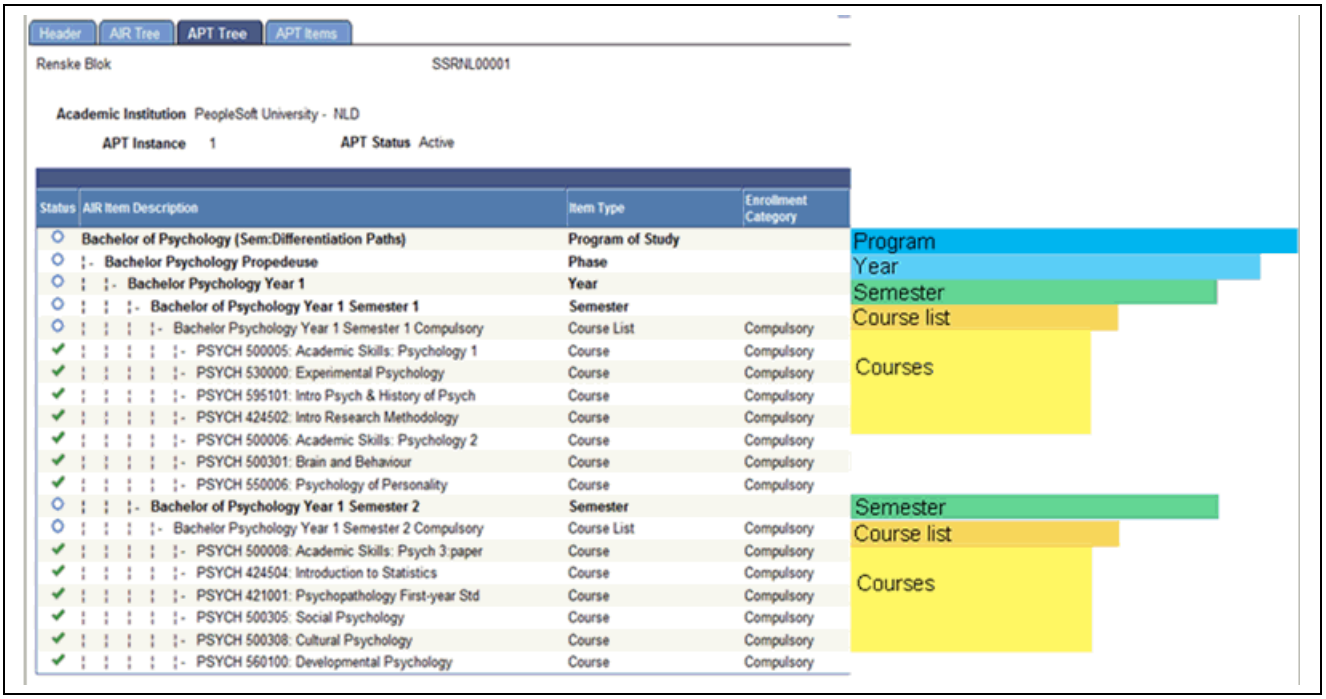
Here is a example that shows part of the APT Entity Registry Hierarchy:



Academic Progress Tracker Program of Study Entity Registry Hierarchy example

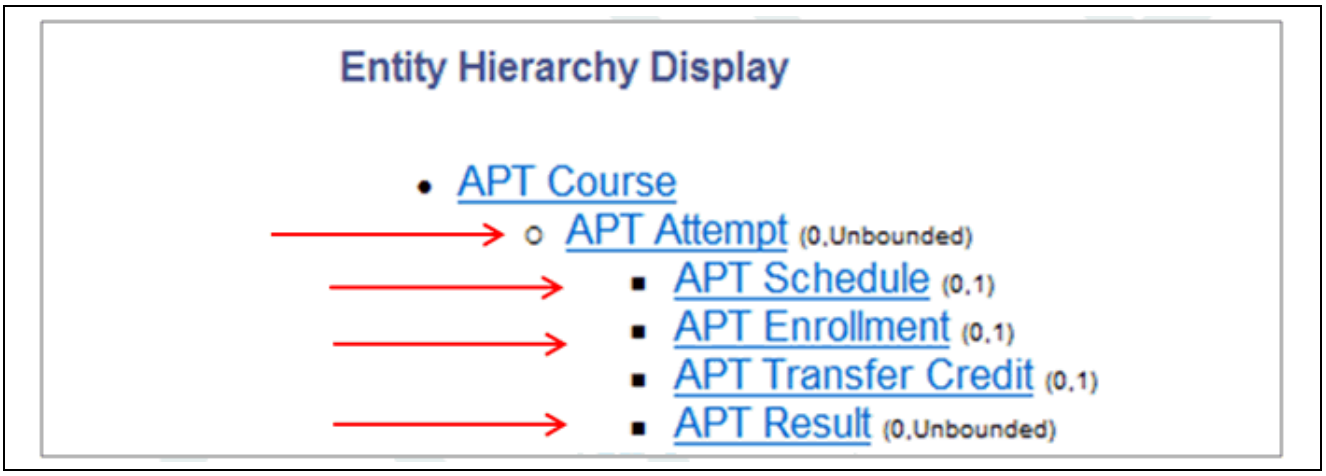
 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the example above, the APT Year and APT Semester Entities have been generated from non-system Academic Item Types Year and Semester. AIR and APT structures are unique in that they can be specified using institution specific objects. The Entity Registry reflects this. The hierarchical relationships between Program of Study, APT year and APT semester are reflected in the APT hierarchy tree and are very similar to how the Curriculum structure reflects in a student's APT:



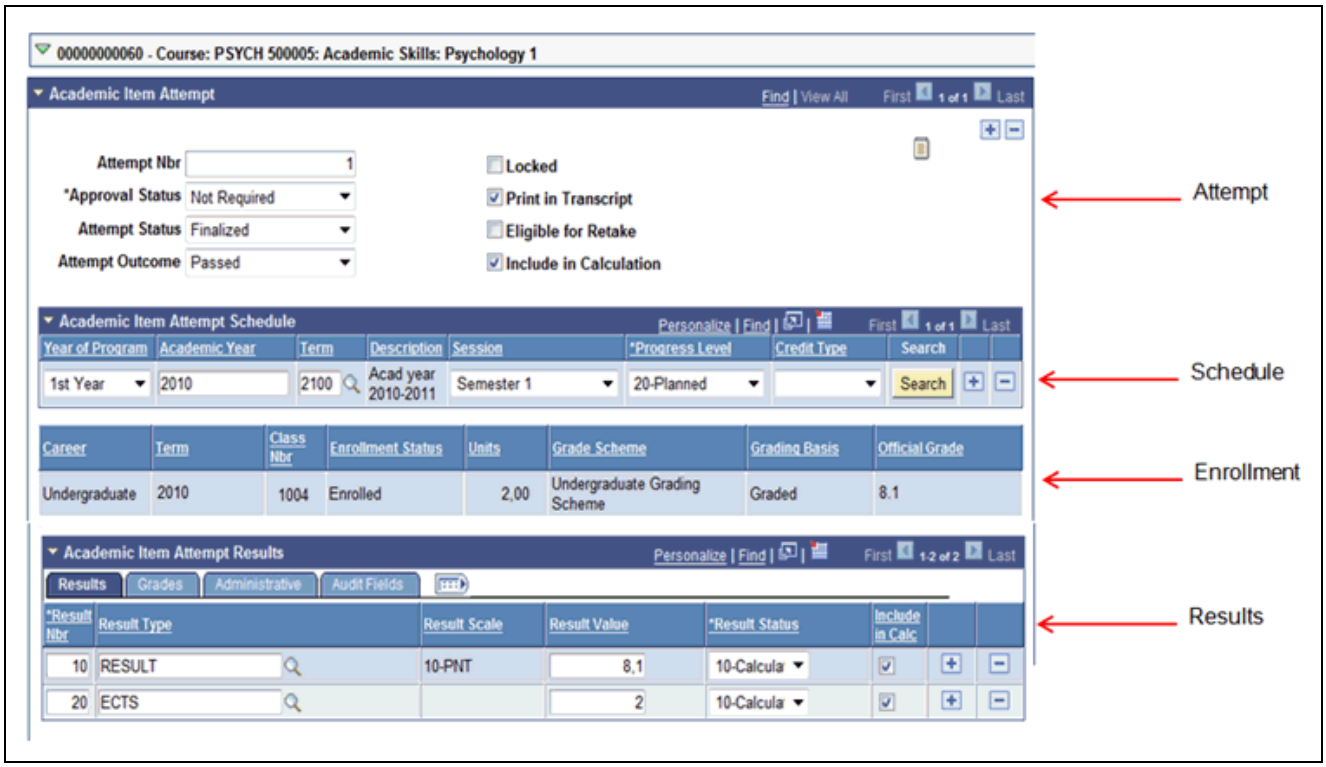
Example of the Academic Progress Tracker Tree

The results from the student's APT are reflected in a similar fashion in the Entity Registry Structure:



Academic Progress Tracker Course Entity Registry Hierarchy example

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

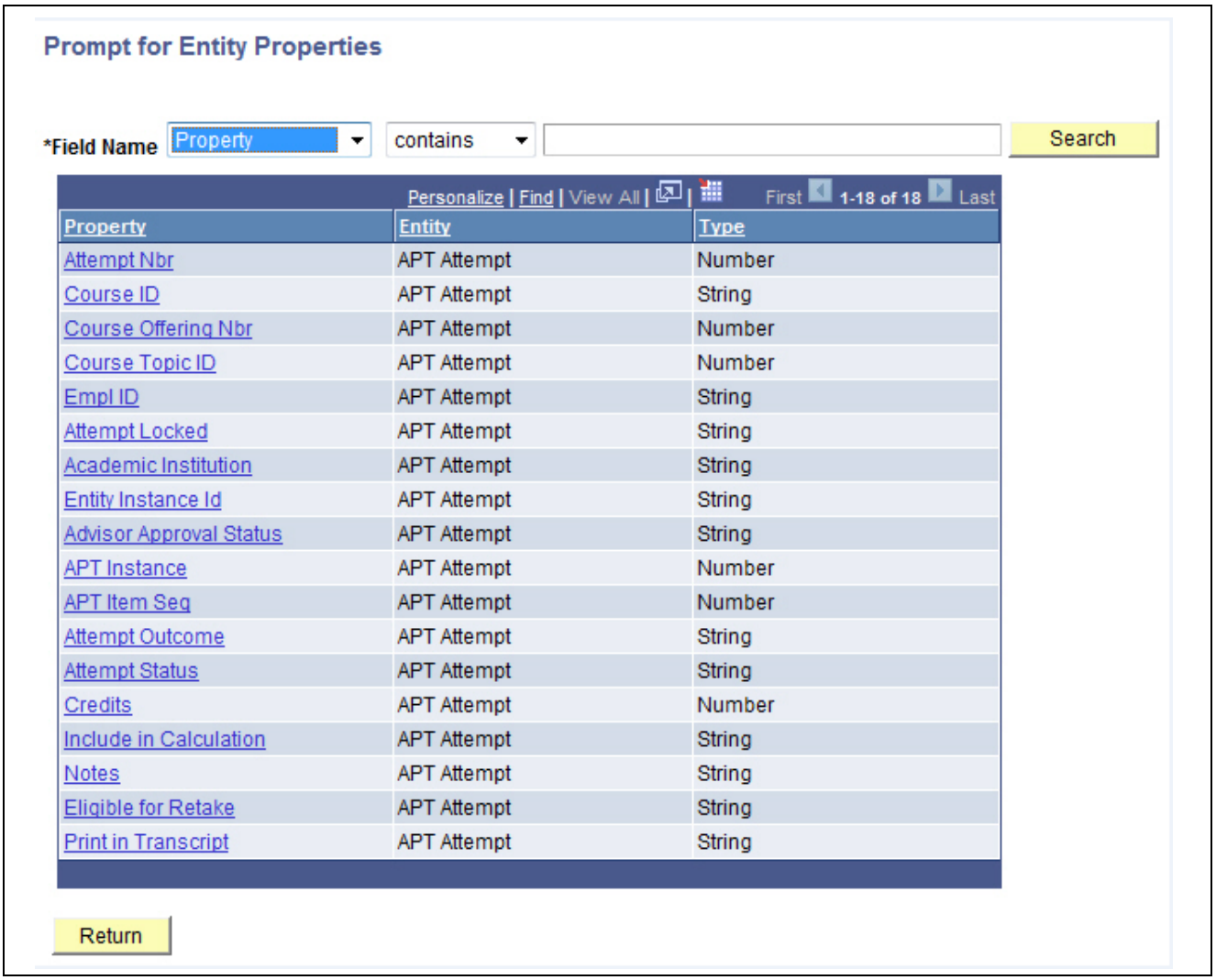


Example of Academic Item Attempt

Entity Properties

Table fields are represented as properties in the Entity Registry. The Rules Engine uses properties to retrieve or update information.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Property	Entity	Type
Attempt Nbr	APT Attempt	Number
Course ID	APT Attempt	String
Course Offering Nbr	APT Attempt	Number
Course Topic ID	APT Attempt	Number
Empl ID	APT Attempt	String
Attempt Locked	APT Attempt	String
Academic Institution	APT Attempt	String
Entity Instance Id	APT Attempt	String
Advisor Approval Status	APT Attempt	String
APT Instance	APT Attempt	Number
APT Item Seq	APT Attempt	Number
Attempt Outcome	APT Attempt	String
Attempt Status	APT Attempt	String
Credits	APT Attempt	Number
Include in Calculation	APT Attempt	String
Notes	APT Attempt	String
Eligible for Retake	APT Attempt	String
Print in Transcript	APT Attempt	String

Example of Entity Properties for Academic Item Attempt

Entity Profiles

Entity Profiles are used to grant or restrict access to specific Entities and/or Entity properties. For example, you want to allow a specific user to create Rules which retrieve information from APT like the Grade Result Value for a Course, but do not want to allow the ability for this same Rule to retrieve any administrative notes that have been added by the supervisor. Through setup of specific Entity Profiles, you can restrict access to the property that represents the Result Notes.

Security for access to specific Entities via the Rules Engine Manager is enforced by associating specific Entity Profiles with Rules Engine Entity Categories.

Additional Rules Engine Features

This section discusses Rules Engine major features.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Rules Engine Categories

Rules Engine security is enforced through the set up of Rules Engine Categories. A Rules Engine Category is user-defined and tied to one or more Roles and/or specific Users. Rules Engine Category security determines which Entities can be accessed by these Roles and Users and what types of Rules can be created. There are three types of Rules: Triggers, Functions, and Rules. For example, specific Rules Engine Categories may allow the creation of Triggers but not of Rules or Functions. Also, Rules Engine Categories can be set up to use other Rules Engine Categories.

Rule Groups

A Rule Group is a template which can be used to create new Rules that share the same functional purpose. A Rule Group provides a predefined set of input and output parameters for a Rule or Function and an option to predefine a Base Entity. Whenever a Rule is created using the predefined Rule Group, all the Input and Output variables are populated using the Rule Group Template options. By using the same parameters, all Rules created in the same Rule Group can be used the same way. This is beneficial when, for example, you need multiple Rules to be called from a user interface, and you need all of those Rules all to provide similar feedback; like a message that can be displayed on screen. You may also want to dictate that the Input for all of these Rules needs to be the same, namely confined to the information available on the user interface. A Rule Group can subsequently be used to dynamically call all Rules associated with that Rule Group.

Creation of Rule Groups is optional.

Rule Creation

You can use the Rules Engine Manager component to Create, Build, Test, and Version a Rule, and determine if a Rule is used by other Rules.

- *Creating*

  You can identify and select a functional application area from which to retrieve data by selecting an Entity and, in the case of a Rule, defining the Criteria which need to be used to select specific data. Use Statements to act upon the selected entity by creating evaluative statements, performing calculations, calling other Rules and Functions, and updating and inserting data in the system.

- *Building*

  Once a Rule is created, you must build (compile) it before testing and using it. The Build action compiles the created Rule and converts it into executable code, which means the Rule is ready to perform evaluation and calculation tasks.

- *Testing*

  After Rules have been created and built, you can test them with the Rules Engine Tester. The Tester allows users to define and save one or more Test Profiles with test specific data so that multiple scenarios can be tested for the same Rule.

- *Versioning*

  The Rules Engine Manager allows the user to create new versions of a Rule and administer one or more version codes and/or code and comments whenever a new version of a Rule is created.

- *Cross Referencing*

  The Cross Reference page lists all Functions and Rules which reference the Rule in context.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Calling Rules and Creating Triggers

After Rules have been created, built, tested, and activated, they are ready to use. To use a Rule as part of a functional business process, it needs to be associated with and called from that process. For example, you can set up a Trigger and use the code created by the Trigger to call Rules from a user interface; keeping in mind that the Trigger-generated code may need some adjustment to work for the specific purpose for which you need it.

Note. The determination of where you want to use Rules and how to make them available must be discussed with the technical team that supports Campus Solutions at your school.

Rules can be called and executed from the following application functionality:

- Rules Engine Manager Tester
- Application Component (for example, a user interface)

For more information, see the *Defining Rule Triggers* section of this chapter.

Setting Up the Rules Engine

This section discusses how to:

- Set Up Rules Engine Install Options.
- Define Rule Category Security.
- Define Color Codes for Rules Engine Manager Elements.
- Define Rules Engine Statements.
- Define Rules Version Reason Codes.
- Set up Rules Engine Variables.
- Define Lists of Values.

Pages Used to Setup the Rules Engine

Page Name	Definition Name	Navigation	Usage
Rules Engine Install Options	SCC_INSTALL_RE	Set up SACR, System Administration, Rules Engine, Setup, Rules Engine Install Options	Set up Rules Engine Install Options
Rule Category Definition	SCC_RULE_CAT_SETUP	Set up SACR, System Administration, Rules Engine, Setup, Define Categories, Definition	Add and define Rule Categories to control the ability to create Rules by Rule Usage. Control Rule Category access to called Rules and Functions and assign a valid Entity Profile.
Rule Category Rule Groups	SCC_RULE_CAT_RLGRP	Set up SACR, System Administration, Rules Engine, Setup, Define Categories, Rule Groups	Determine if Rule Groups are required for a Rule Category and assign valid Rule Groups.

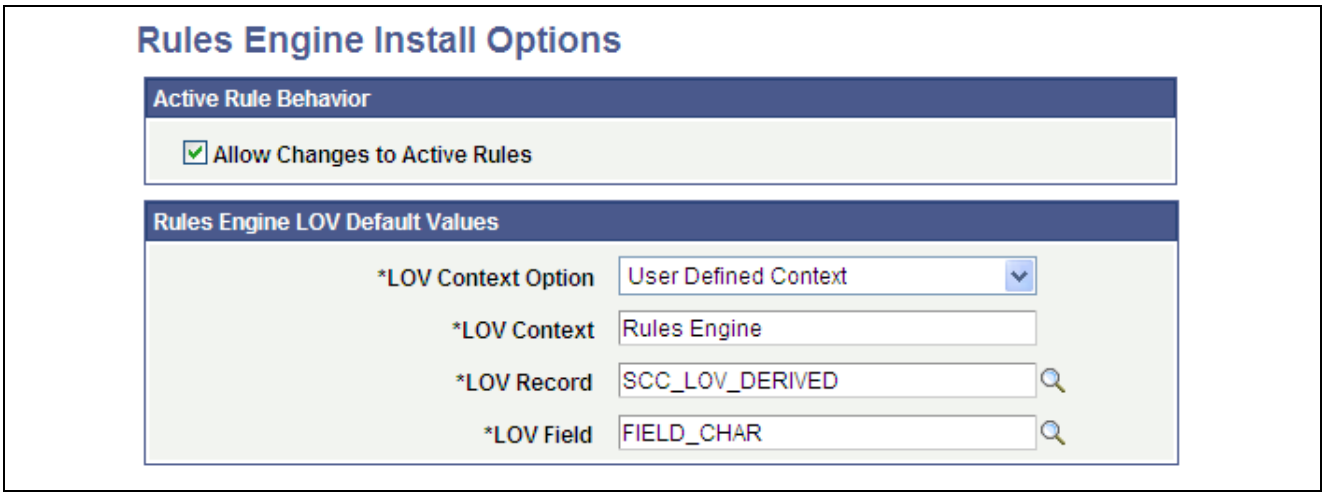
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Page Name	Definition Name	Navigation	Usage
Rule Category Security	SCC_RULE_CAT_SCRTY	Set up SACR, System Administration, Rules Engine, Setup, Define Categories, Security	Assign valid roles and additional users that have access to a Rule Category.
Define Colors	SCC_COLORS	Set up SACR, System Administration, Rules Engine, Setup, Define Colors	Define colors to associate with Rules Engine user interface elements.
Define Rules Engine Text Colors	SCC_RULE_TXT_COLOR	Set up SACR, System Administration, Rules Engine, Setup, Define Text Color	Associate colors with Rules Engine user interface elements.
Define Rules Engine Statements	SCC_RULE_STMT_TBL	Set up SACR, System Administration, Rules Engine, Setup, Define Statements	Define Statements to use in Rule evaluations and calculations.
Define Version Reason Codes	SCC_RULE_VRSN_TBL	Set up SACR, System Administration, Rules Engine, Setup, Define Version Reason Codes	Define Rules Version Reason Codes to use when creating new versions of existing Rules.
Define System Variables	SCC_RULE_SYSVAR	Set up SACR, System Administration, Rules Engine, Setup, Define System Variables	Define System Variables to use when creating Rules.
Define Data Sets	SCC_RULE_EDS	Set up SACR, System Administration, Rules Engine, Setup, Define Data Sets	Define Data Sets to use as a temporary storage when creating Rules.
Data Set Property Details	SCC_EDS_PROP_DTLS	Set up SACR, System Administration, Rules Engine, Setup, Define Data Sets, Details link	Define Data Set property details.
Define List of Values	SCC_RULE_LOV_DEFN	Set up SACR, System Administration, Rules Engine, Setup, Define List of Values	Define lists of values to use for property prompting when creating Rules.

Setting Up Rules Engine Install Options

Access the Rules Engine Install Options page (Set up SACR, System Administration, Rules Engine, Setup, Rules Engine Install Options).

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Rules Engine Install Options page

Allow Changes to Active Rules

Select this check box to allow changes to active Rules or Functions. You can use this flag to override the feature that makes Rules features inaccessible when Rules or Functions are active. This option can be useful in non-production environments in cases where it should be possible to change active Rules.

The check box is not selected by default.

When Allow Changes to Active Rules is not selected only the following can be changed or accessed by the user when accessing the Rules Engine Manager Component:

- Rule Long Description (change)
- View of Variables (access)
- View of Evaluations and Calculation Details. (access)

When Allow Changes to Active Rules is not selected, the following Action Drop down options are not available:

- Create new Version of Rule
- Delete Rule
- Inactivate Rule
- Remove Rule Group from Rule
- Delete Rule Group
- Inactivate Rule Group

When Allow Changes to Active Rules is not selected, Rule Groups that are selected to be Available In Dynamic Rules cannot be altered.

Rules Engine LOV Default Values

Rules Engine List of Values (LOV) allows you to define Rules Engine prompting for variables created and used in the Rules Engine Manager. The system-delivered default values of the fields in this group box control LOV functionality in the Rules Engine Manager.

Warning! Do not change the values delivered with the system.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

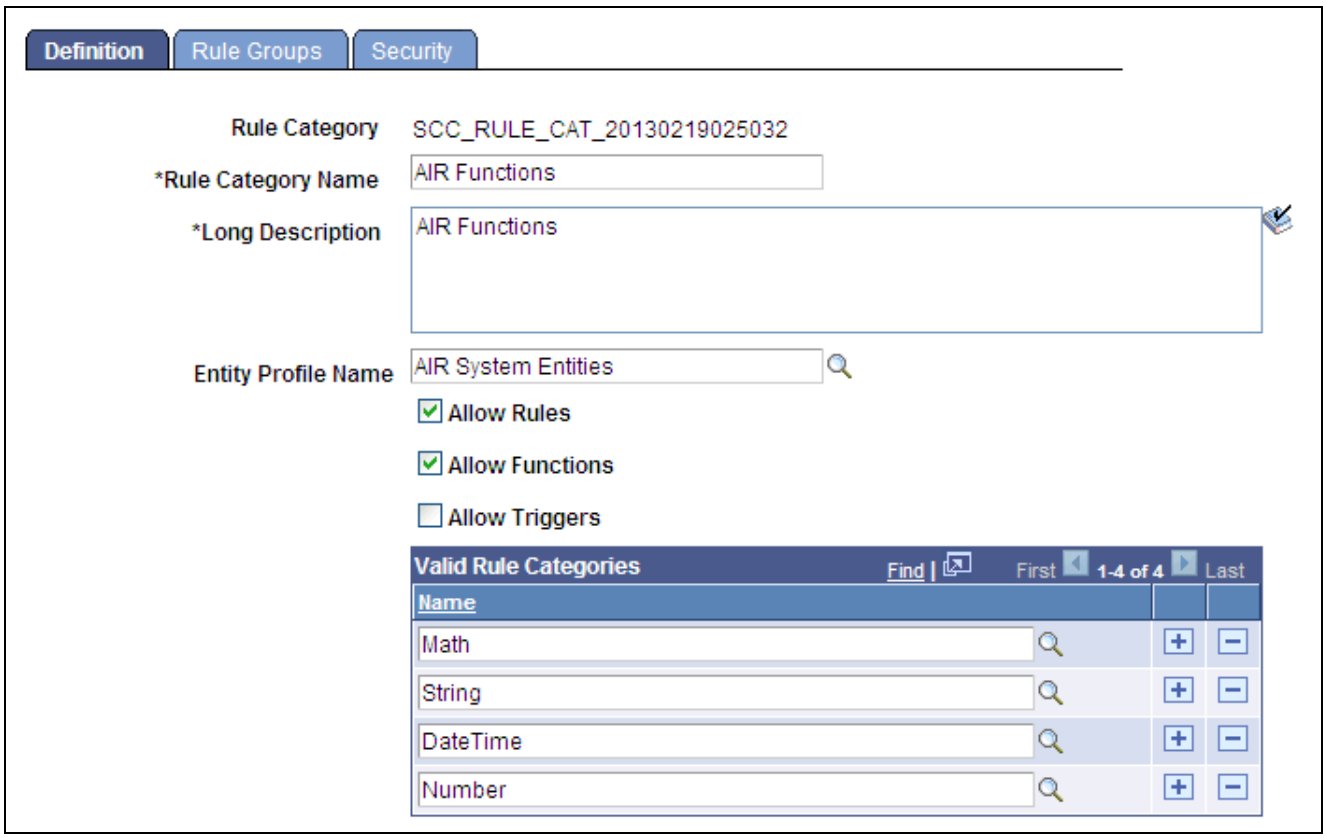
Defining Rule Category Security

Rules Engine Categories provide a means of administering various Rules Engine settings by Role and/or User. Every Rule that is created must belong to one predefined Rule Category. The Rule Category is used to control the following:

- The type of Rule that can be created as defined by Rule Usage: Rule, Function, and/or Trigger. Rule Usage determines how a Rule can be used in a business process.
- Access to Rules in other Rule Categories. Multiple Rule Categories can be added to the Rule Category definition. When these Rule Categories are added, Rules or Functions belonging to the associated Rules Category can be called from the main Rule.
- Access to Data through Entities. The Entity Profile attached to the Rule Category defines which Entities can be used as a Base Entity in the Rule created. The Base Entity is the starting point from which the logic in the Rule is built. Depending upon the Entity Profile setup, you may have access to one or more Base Entities *and* all or a defined set of Entity Properties.

  See [PeopleSoft Campus Community 9.0 Fundamentals PeopleBook]Setting Up Entity Registry.
- Valid Rule Groups. Valid Rule Groups can be associated with a Rules Category. Restrictions may be applied so that you can only build Rules from a Rule Group. This can be beneficial in circumstances where you want to guide a group of users with Rule creation by making sure that they build their Rule according to the defaults provided by the Rule Group or Rule Groups associated with this Rule Category.

Access the Rule Category Definition page (Set up SACR, System Administration, Rules Engine, Setup, Define Categories, Definition).



Rule Category Definition page

Using this category as an example, you cannot create Functions or Rules in category Math, but you can create Functions and Rules in category AIR Functions and CALL Functions in category Math.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Rule Category Displays a unique ID generated by the system. When adding a new Rule Category, the default value is *NOID*. After saving the Rule Category, a unique ID is generated by the system and assigned to the Rule Category. The ID is created by combining prefix "SCC_RULE_CAT_" with the system date and time stamp in format YYYYMMDDHHMMSS.

Rule Category Name Enter a Rule Category Name. The Rule Category Name is used when searching for a Rule Category and for display.

Long Description Enter descriptive text explaining the function of the Rule Category.

Entity Profile Name Enter the Entity Profile Name that you want to associate to this Rule Category. Rules, Functions and Triggers can be built using the Base Entities which have been setup in this Entity Profile. If the Entity Profiles uses Views access to certain properties within an Entity may haven been restricted.

Allow Rules Select this check box to allow the creation of Rules with this Rule Category.

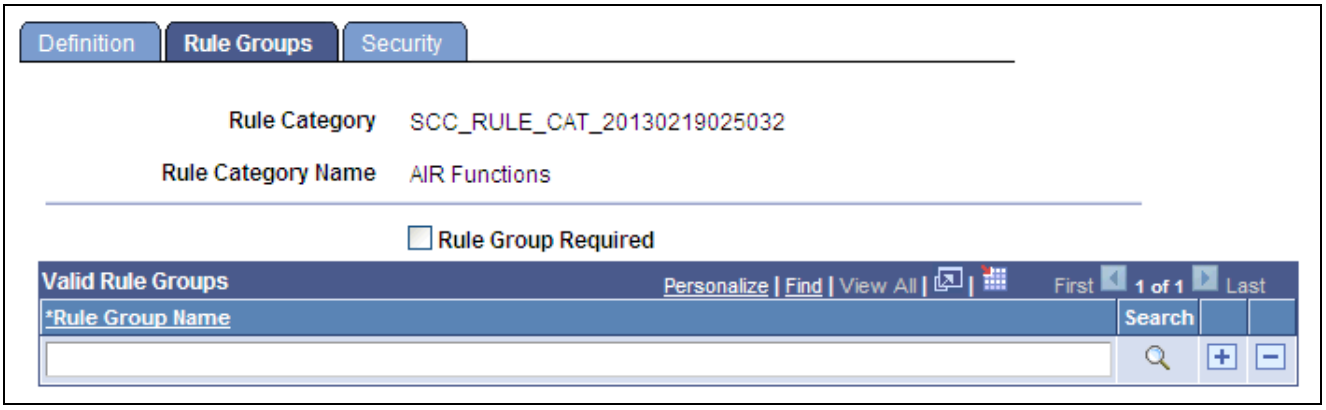
Allow Functions Select this check box to allow the creation of Functions with this Rule Category.

Allow Triggers Select this check box to allow the creation of Triggers with this Rule Category.

Valid Rule Categories Select Rule Categories available to this Rule Category. The Rules, Functions, and Triggers of the selected categories can be called by Rules created in the main Category.

Note. This is not an inheritance model. Adding a valid Rule Category here gives you access to the Rules, Functions, and Triggers *native to* that Rule Category but *not* to Rules, Functions, and Triggers of any Rule Categories associated with the selected Rule Category.

Access the Rule Groups page (Set up SACR, System Administration, Rules Engine, Setup, Define Categories, Rule Groups



Rule Category Rule Groups page

Use this page to add valid Rule Groups to this Rule Category or to restrict users to only being able to build Rules by using one of the associated Rule Groups.

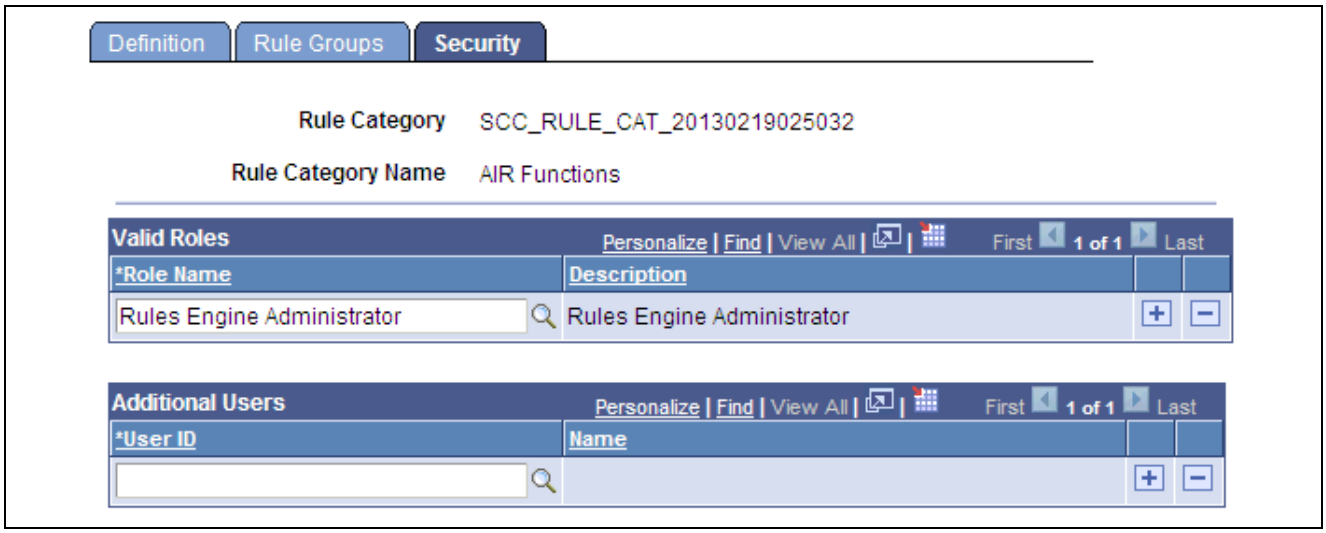
Rule Group Required Select to enforce Rule Group usage. When building Rules users must select one of the Valid Rule Groups before being able to build their Rules.

Valid Rule Groups Select Rule Groups.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note. Rule Groups are added to this setup page automatically when you add Rules to a Rule Group by using the Create Rule Group option from Rules Engine Manager or by using Create Rule Group from a Rule when on the Rules Engine Search option page.

Access the Rules Category Security page (Set up SACR, System Administration, Rules Engine, Setup, Define Categories, Security).



Rule Category Security page

Valid Roles Enter Valid Roles for which the Rules Category definition is valid. These are Roles that are set up using the standard PeopleTools Security function.

Additional Users Enter additional users for which the Category Definition is valid. The registration of Additional Users is supplemental to the Valid Roles.

Defining Color Codes for Rules Engine Manager Elements

A set of predefined color codes for the Rules Engine Manager user interface are delivered. The different colors represent different types of elements of the Rule; for example, Rule Names, Statements, Variables, etc. You can customize the colors and the Rule elements with which they are associated. The following color codes are delivered:

Color	Color Code ID
Dark Purple	380B61
Black	000000
Blue	0000ff
Red	ff0000
Green	008000
Purple	800080
Brown	a52a2a
Gray	808080
Dark Blue	0000a0
Yellow	ffff00

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Color	Color Code ID
Orange	ffa500
Maroon	800000

Access the Define Colors page (Set up SACR, System Administration, Rules Engine, Setup, Define Colors, Define Colors).

Define Colors page

Although Oracle recommends using the pre-delivered color range, you can add new colors to the system for the Rules Engine Manager user interface elements. New colors can be added simply by adding a new entry, specifying a Color Name and a Color Code which conforms to the HTML color code standard.

Color Code ID Displays a unique ID generated by the system. When adding a new value the default value is *NOID*. The unique ID is generated when the Color Code is saved. The Color Code ID is created by combining prefix SCC_COLORS_ with the system date and time stamp in format YYYYMMDDHHMMSS.

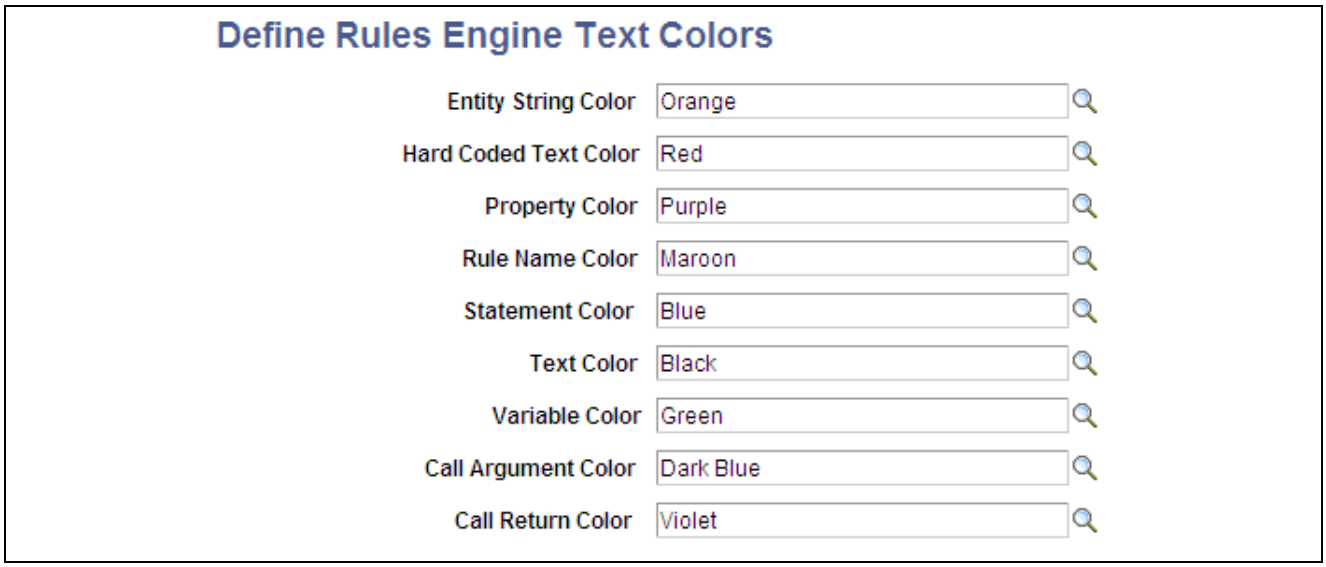
Color Name Enter the name of the color; for example, Sky Blue.

Color Code Enter an HTML standard color code. New color codes can be defined using HTML standards.

- HTML color codes format – Each HTML code contains the symbol "#" and 6 letters or numbers. These numbers are in hexadecimal numeral system; for example, "FF" in hexadecimal represents number 255 in decimal.
- Meaning of the HTML color codes format – After the “#” symbol, the first two positions in HTML color code represent the intensity of red color. “00” is the least, and “FF” is the most intense. The third and fourth positions represent the intensity of green color, and the fifth and sixth positions represent the intensity of blue color. By varying the intensity of red, green and blue, you can create a large number of colors.

Access the Define Rules Engine Text Colors page (Set up SACR, System Administration, Rules Engine, Setup, Define Text Color, Define Rules Engine Text Colors).

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Define Rules Engine Text Colors page

Use this page to assign colors to Rules Engine Manager elements. The following elements can be color coded:

- Entity String
- Hard Coded Text
- Property
- Rule Name
- Statement
- Text
- Variable
- Call Argument
- Call Return

Defining Rules Engine Statements

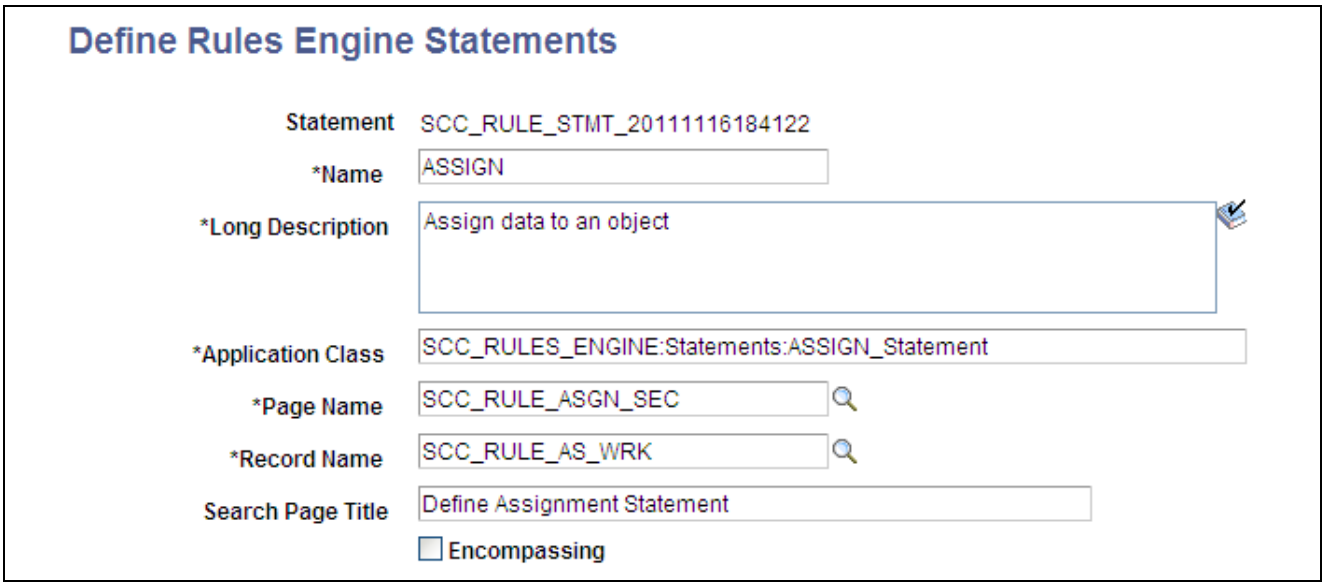
Rules Engine Statements help you create business rules in the Rules Engine Manager to perform specific processing Functions like assigning values to variables or creating Evaluative Statements. An extensive set of Statements is delivered with the system to facilitate the creation of a comprehensive set of Rules. The processing functionality for each statement is defined in Application Class PeopleCode and associated with a Rules Engine Manager secondary page through a predefined work record. This secondary page is used to display fields and logic associated with the Statement.

For more information, see the *Using Statements for Evaluation and Calculation in a Rule* section of this chapter.

Note. Oracle may deliver additional Statements in the future. You are advised *not* to create your own Statements at this time.

Access the Define Rules Engine Statements (Set up SACR, System Administration, Rules Engine, Setup, Define Statements, Define Rules Engine Statements).

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Define Rules Engine Statements page

Statement

Displays a unique system-generated Rules Engine Statement identifier. When adding a new value the default value is *NOID*. The unique ID is generated when the Rules Engine Statement is saved. The Rules Engine Statement ID is created by combining prefix SCC_RULE_STMT_ with the system date and time stamp in format YYYYMMDDHHMMSS.

Name

Enter a Statement Name. The Statement Name is used when searching for a Statement and for display.

Long Description

Enter descriptive text explaining the function of the Statement.

Application Class

Enter the Application Class for this Rules Engine Statement. Logic for system–delivered Statement functionality uses the SCC_RULES_ENGINE:Statements Application Class.

Warning! To ensure that delivered functionality works as intended, do *not* change delivered Application Classes.

Page Name

Enter the Page Name for this Rules Engine Statement. Each Statement references a secondary page specifically created for this Statement. The secondary page is used to display Statement functionality to the end user. The Statement secondary pages are pre–delivered and are designed to render Statement logic in a specific way.

Warning! To ensure that delivered functionality works as intended, do *not* change the Page Name for the Statement.

Record Name

Enter the Record Name for this Rules Engine Statement. Each Statement has a unique work record reference. The associated work record is used to technically facilitate the information shown on the secondary pages. The Statement work records are pre–delivered and have been designed to render Statement logic in a specific way.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Warning! To ensure that delivered functionality works as intended, do *not* change the Record Name for the Statement.

Search Page Title Enter the Page Title for the Rules Engine Statement. The Page Title is displayed at the top of the Statement grid once the Statement has been selected for use in the Rule.

Encompassing Select to indicate whether the Statement created is an encompassing statement. An Encompassing Statement is one which can enclose one or more other Statements in the Rule. The following delivered statements are encompassing:

- IF (ELSE)
- For-each
- Create-Entity

For more information, see the *Using Statements for Evaluation and Calculation in a Rule* section of this chapter.

Define Rules Version Reason Codes

New versions of Rules can be created for each Rule, Trigger or Function. Rule Versions are managed on the Rules Engine Manager Rule Version Page. New versions of Rules can be created for reasons like needing to incorporate new Rule logic or correcting Rule mistakes. To facilitate Rule maintenance, use Rule Version reason codes to indicate why a new version was created. Define Rule Version Codes on the Rules Version Reason Code setup page.

Access the Define Version Reason Codes page (Set up SACR, System Administration, Rules Engine, Setup, Define Version Reason Codes).

Define Version Reason Codes page

Version Reason Code Displays a unique ID generated by the system. When adding a new value the default value is *NOID*. The unique ID is generated when Version Reason Code is saved. The ID is created by combining prefix SCC_RULE_VRSN_ with the system date and time stamp in format YYYYMMDDHHMMSS.

Name Enter a Version Reason Name. The Version Reason Name is used when searching for a Version Reason and for display.

Description Enter descriptive text explaining the function of the Version Reason Code..

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Setting Up Rules Engine Variables

This section discusses setting up Rules Engine Variables.

Understanding Rules Engine Variables

Variables provide you with a flexible way to create a temporary placeholder or storage location which can be used in a Rule or passed from the current Rule to another. A Variable has a name and a certain type; for example, “Text” or “Number”. The Variable starts out as empty storage and can be assigned a value, cleared, and re-used.

For example, a Variable named “Total Units” is a variable of type “Number”. The Variable is used in a Function that calculates Total Units from Course Units that are found in a specific Course List. At the end of the Function, the Variable “Total Units” is displayed as the result or outcome of the Rule.

The following Variables can be created when working with the Rules Engine Manager:

- Rules Engine Manager Variables
- System Variables
- Data Sets

Rules Engine Manager Variables are created in a specific Rule and act as a storage place to temporarily store a value in that particular Rule. The previously mentioned “Total Units” is an example. Such a variable can be passed from one Rule to another, but it cannot be referenced by another Rule without having been passed.

System Variables have been predefined in the Define System Variable component and have a specific predefined value. System Variables are available for all Rules in the system and are intended to provide values which are generic and are Variables that are appropriate in multiple situations. Their values do not need to change from one Rule to the next. An example is “Current Date” which always provides the user with value of the current system date. System Variables are typically created by Developers but, once created, can be used by Functional Expert Users in their Rules.

Data Sets allow you to define multiple Variables that can be referenced as a group. When building more complex Rules you may need more than one Variable as a temporary placeholder. There may be situations where you need multiple temporary placeholders that can be stored together as a logical set of Variables. For example, when calculating Total Units for multiple students in batch, you may need to not only store the “Total Units” but also the Student (in the example below we store Student ID as well as Student Name) and the Course List ID. In this case you may want to create three temporary Variables that can be referenced together as a group. This is a Data Set. The Data Set described above would allow you to store the following example data:

Student ID	Student Name	Course List ID	Total Units
0000012	Brad Wilkinson	Math100	12
0000011	Brenda Benson	Math100	22
0000010	Billy Mathews	Math100	18

Defining Rules Engine Manager Variables

Rules Engine Manager Variables can be defined whenever you need to use a named temporary storage space of a specific type in which to store a value in the Rules Engine. You can pass that Variable to other Rules or use the Variable to store data retrieved from other Rules. Variables are always created within a Rule itself.

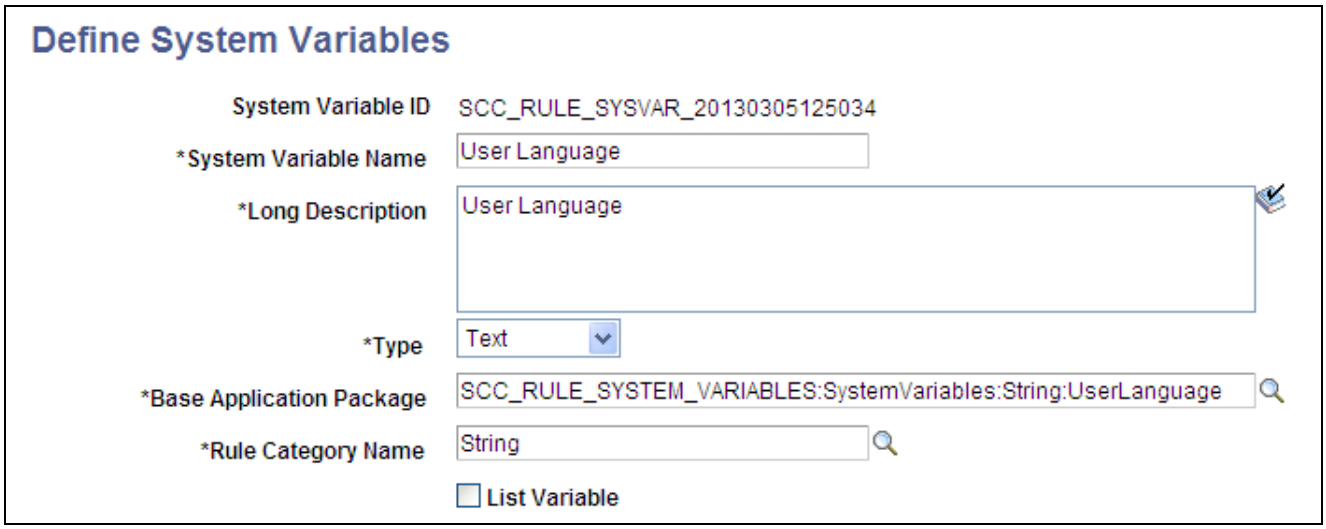
For more information, see the *Adding Variables to a Rule* section of this chapter.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Defining Rules Engine System Variables

A System Variable is predefined in the system typically by a developer or programmer. Once created, the System Variable can be used in Rules built by functional experts. System Variables can be created to provide re-usable defaults for commonly used values such as system date, Operator ID, etc.

Access the Define System Variables (Set up SACR, System Administration, Rules Engine, Setup, Define System Variables).



Define System Variables page

System Variable ID Displays a unique ID generated by the system. When adding a new value the default value is *NOID*. The unique ID is generated when the system variable is saved. The Statement ID is created by combining prefix SCC_RULE_SYSVAR_ with the system date and time stamp in format YYYYMMDDHHMMSS.

System Variable Name Enter a System Variable Name. The System Variable Name is used when searching for a System Variable and for display.

Long Description Enter descriptive text explaining the function of the System Variable.

Type Select the Type of Variable. Values are:

- Date
- DatetTime
- Text
- Time
- Number
- True/False (This Variable Type accommodates Boolean values.)

Base Application Package Enter the Base Application Package for this System Variable. The Base Application Package contains the logic that returns the System Variable. System Variables can only be assigned using application package PeopleCode. Delivered system variables have been created in the reserved application package: SCC_RULE_SYSTEM_VARIABLES:SystemVariables.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note. Oracle expects to make new system variables available as needed.

Rule Category Name Enter the Rule Category Name for this System Variable. This restricts direct access to specific System Variables.

List Variable Select this check box if the System Variable needs to accommodate the return of multiple values.

The following System Variables are delivered with the system for use in Rules and Functions by an Expert user:

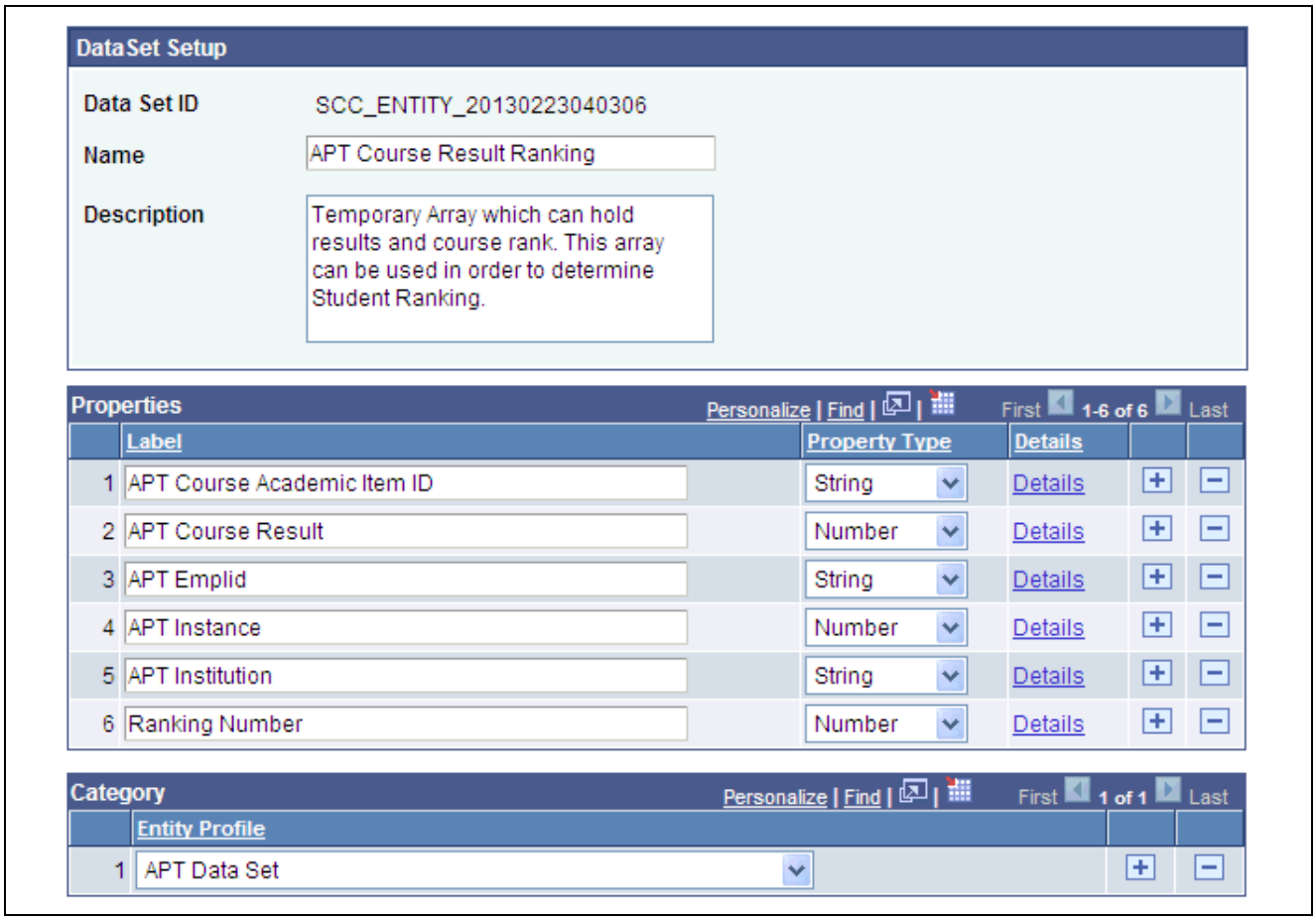
System Variable Name	Rule Category Name	Function
Current Date	DateTime	Returns the Current Date in format YYYY-MM-DD
Current date Time	DateTime	Returns the Current Date in format YYYY-MM-DD HH:MM:SS
OperatorID	String	Returns the ID of the Operator that is running the Rule.
Current Time	DateTime	Returns the Current Time in format HH:MM:SS

Defining Rules Engine Data Sets

Data Sets can be used in Rules when you need to create placeholders or temporary storage for multiple Variables and reference those Variables together as a logical group. The Data Set can be used across any Rule in the system and is not specific for one Rule alone.

Access the Define Data Sets Page (Set up SACR, System Administration, Rules Engine, Setup, Define Data Set).

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Data Set Definition page

Data Set ID Displays a unique ID generated by the system. When adding a new value the default value is *NOID*. The unique ID is generated when the Data Set is saved. The Statement ID is created by combining prefix SCC_ENTITY_ with the system date and time stamp in format YYYYMMDDHHMMSS.

Name Enter a Data Set Name. The Data Set Name is used when searching for a Data Set and for display.

Description Enter descriptive text explaining the function of the Data Set..

Label Enter a label for each of your Data Set properties. This Label is used in the Rules Engine Manager to display the Data Set Property.

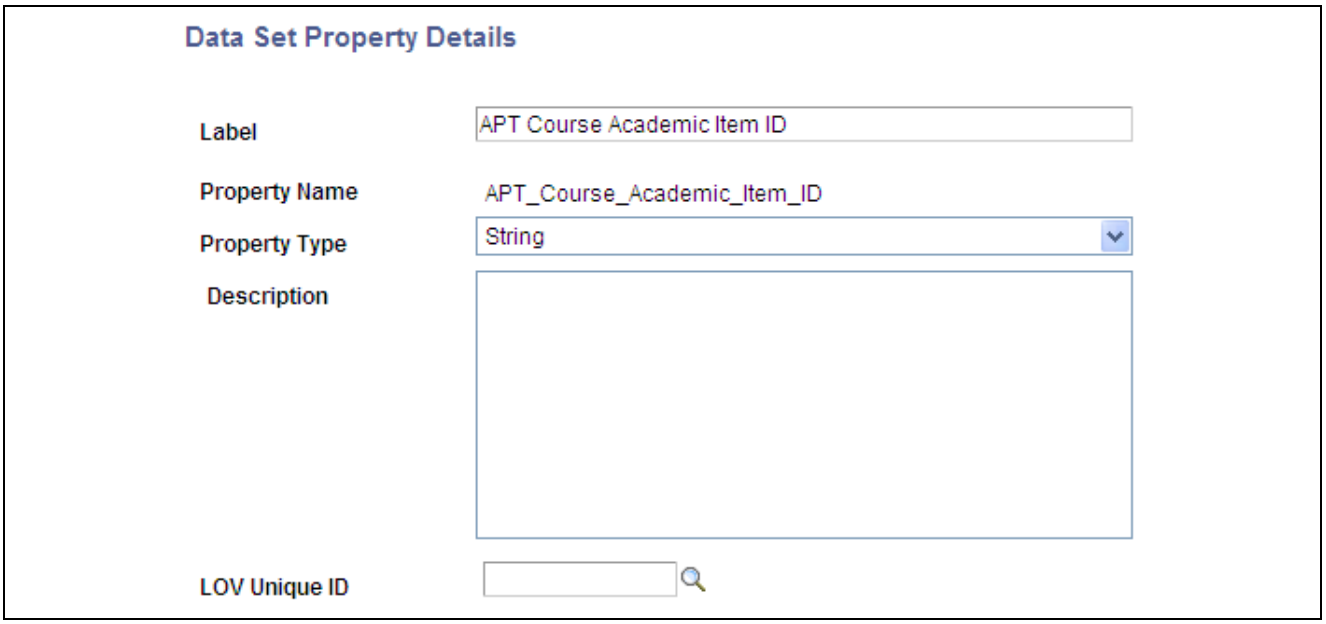
Property Type Select the Property Type:

- Date
- DateTime
- Number
- String
- Time

Details Click on the Details link to access more information about each Data Set Property.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Access the Data Sets Details page (Set up SACR, System Administration, Rules Engine, Setup, Define Data Set, Details link).



Data Set Property Details page

The Data Set Property details page allows you to add a long Description for a Property and List of Values (LOV) which can be used to enforce Prompt Edits.

For more information, see the *Defining Lists of Values for Rules Engine Variables* section of this chapter.

Entity Profile Select Entity Profiles to associate multiple Data Sets to a group of Data Sets. The Category is used to search for Data Sets when associating Rules Engine Variables with created Data Sets. There is no security tied into this functionality. A created Data Set can be used by any Rule in the system.

Defining Lists of Values for Rules Engine Variables

This section discusses defining lists of values for use with Rules Engine Variables. There are cross references to information about adding lists of values to Entity Properties and Data Set Properties.

Understanding Lists of Values for Rules Engine Variables

List of Values (LOV) functionality allows you to add the functionality of prompting with edits against a defined List of Values where there were no edits defined before. In Rules Engine Manager, a LOV can be added to properties or variables of the following Types:

- Rules Engine Variable
- Entity Property
- Data Set Property

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

For example, take a Variable created in Rules Engine named “Institution” . When defined as a Variable of type String and referenced in a Rules Engine Rule, any String value can be added to the Variable and no editing is enforced. However, you may want to enforce that users can only add values to this Variable that are valid according to the Institution Table defined in the system (Set up SACR, Foundation Tables, Academic Structure, Institution Table). In order to enforce that edit, you can define a LOV on the Institution Table. Once defined, the LOV can be tied to any Rules Engine Variable which has the same Type (for example, String) as the defined LOV.

To understand more about how to add an LOV to an Entity Property,

See [PeopleSoft Campus Community 9.0 Fundamentals PeopleBook]Setting Up Entity Registry.

Refer to the Setting Up Entity Property Details section

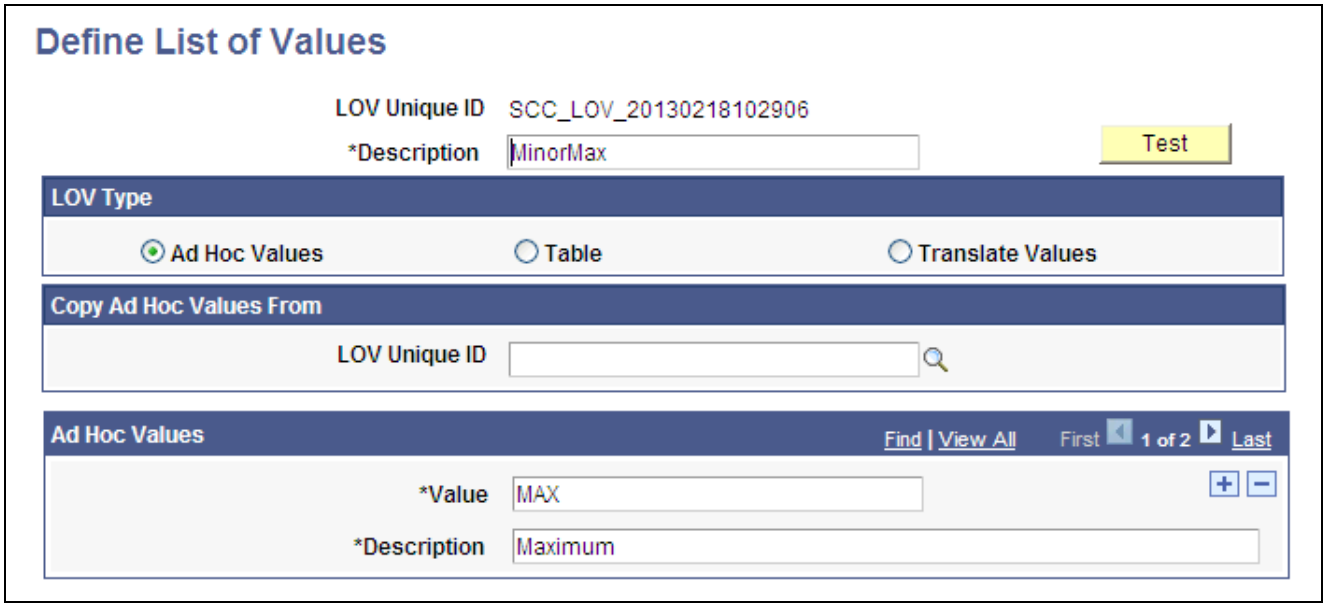
To understand more about how to add an LOV to a Data Set Property, see the *Defining Rules Engine Data Sets* section of this chapter.

To understand more about how LOV functionality can be used throughout the system,

See [PeopleSoft Campus Community 9.0 Fundamentals PeopleBook]Setting Up List of Values.

Defining Lists of Values – Ad Hoc Values

Access the Define List Of Values page (Set up SACR, System Administration, Rules Engine Setup, Define List of Values, select Ad Hoc Values radio button).



Define List of Values – Ad Hoc Values page

Select LOV Type *Ad Hoc Values* if you want to prompt on a user-defined Value or List of Values that is not related to a translate table or an existing prompt table.

LOV Unique ID Displays a unique ID generated by the system. When adding a new value the default value is *NOID*. The unique ID is generated when the LOV is saved. The UID is created by combining prefix SCC_LOV_ with the system date and time stamp in format YYYYMMDDHHMMSS.

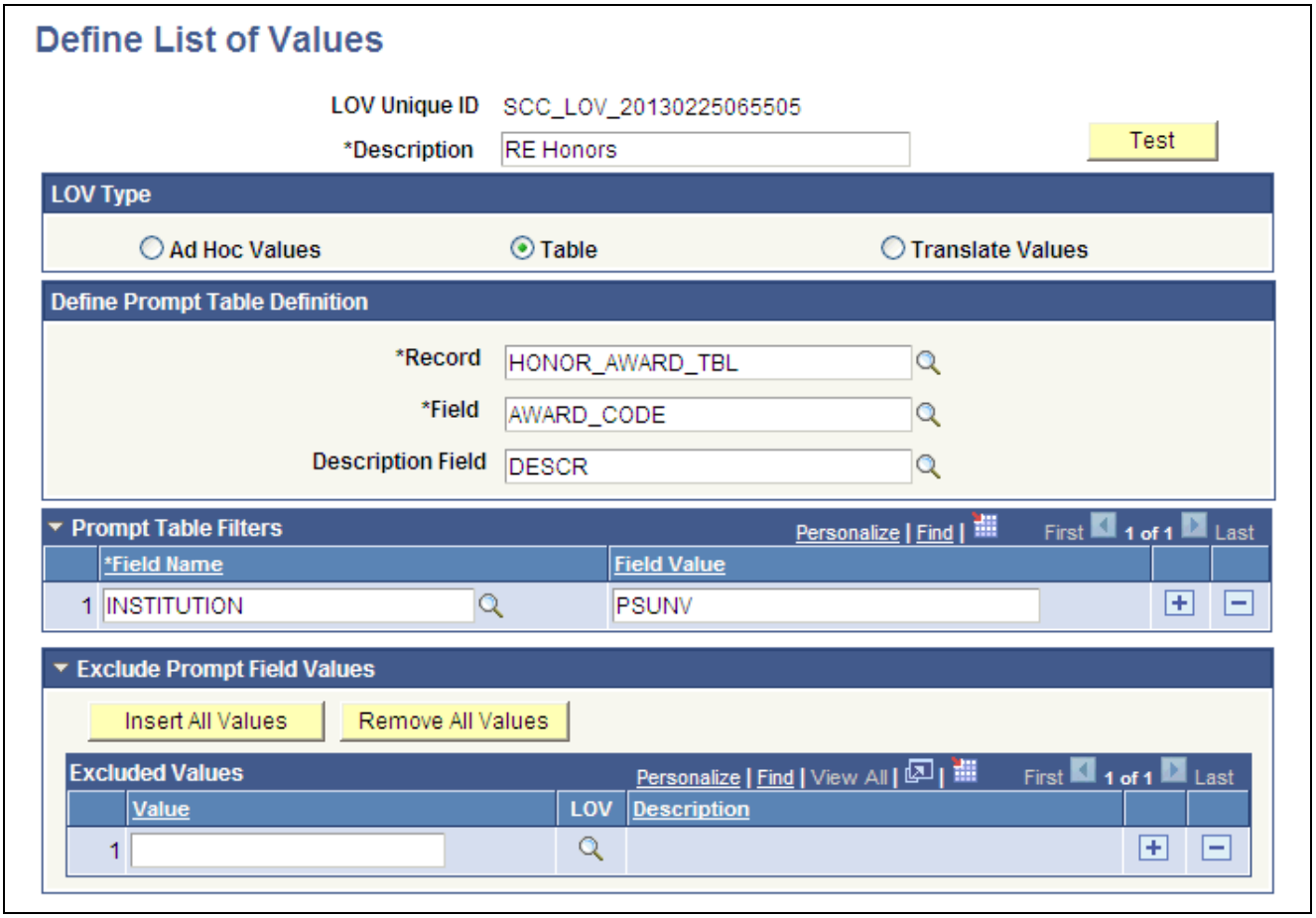
Description Enter a LOV Description. The LOV Description is used when searching for a LOV and for display.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

LOV Type – Ad Hoc Values selected	Create an LOV of *Ad Hoc Values* if you want to prompt on a user-defined value or List of Values that is not related to a translate table or an existing prompt table.
LOV Unique ID – Copy Ad Hoc Values From grid	Select an LOV to copy by using the Search icon to find an existing LOV Unique ID. Then add custom values in the Ad Hoc Values grid.
Value – Copy Ad Hoc Values grid	Enter an Ad Hoc Value from which you would like to copy existing Ad Hoc values.
Description – Copy Ad Hoc Values grid	Enter a Description for the Ad Hoc Value.
Test	Click the Test button to display a preview of the values that will be returned by the LOV when it is deployed as a prompt edit.

Defining Lists of Values – Table

Access the Define List Of Values page (Set up SACR, System Administration, Rules Engine Setup, Define List of Values, select Table radio button).



Define List of Values – Table

Select an LOV Type of *Table* if you want to prompt on an existing Campus Solutions Prompt Table.

LOV Unique ID	Displays a unique ID generated by the system. When adding a new value the default value is *NOID*. The unique ID is generated when the LOV is saved.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

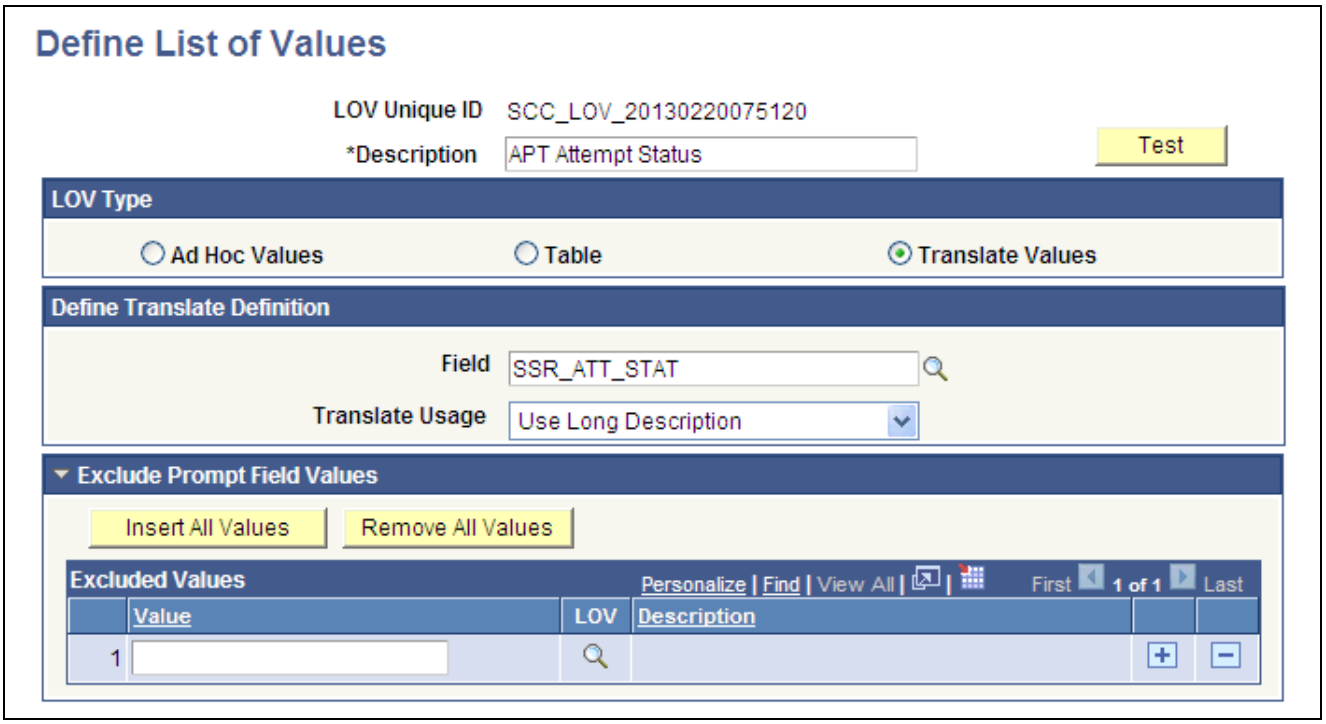
	The UID is created by combining prefix SCC_LOV_ with the system date and time stamp in format YYYYMMDDHHMMSS.
Description	Enter a LOV Description. The LOV Description is used when searching for a LOV and for display.
LOV Type – Table selected	Create an LOV of type *Table* if you want to prompt on an existing Campus Solutions Prompt Table.
Record	Enter the Record where the prompt is found.
Field	Select a Field for the prompt.
Description Field	Select the Field from which the description should be taken.
Prompt Table Filters	Select Field Names to further specify the selection. For example in case of the Honors and Awards Table, the setup is defined by Institution. To specify that only values from institution PSUNV be retrieved, add the Field Name *Institution* and Field Value *PSUNV*.
Exclude Prompt Field Values	• Insert All Values – Click button to add all LOV values to the excluded values list. • Remove All Values – Click button to remove all LOV values from the excluded values list.
Test	Click the Test button to display a preview of the values that will be returned by the LOV when it is deployed as a prompt edit. The Test functionality uses the Exclude Prompt Field Values and Prompt Table Filter settings.

Defining Lists of Values – Translate Values

Access the Define List Of Values page (Set up SACR, System Administration, Rules Engine Setup, Define List of Values, select Translate Values radio button).

Create an LOV of type *Translate Values* if you want to prompt on an existing Campus Solutions Translate Value.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Define List of Values – Translate Values

Select an LOV Type of *Translate Values* if you want to prompt on an existing Campus Solutions Translate Values.

LOV Unique ID

Displays a unique ID generated by the system. When adding a new value the default value is *NOID*. The unique ID is generated when the LOV is saved. The unique ID is created by combining prefix SCC_LOV_ with the system date and time stamp in format YYYYMMDDHHMMSS.

Description

Enter a LOV Description. The LOV Description is used when searching for a LOV and for display.

Field

Enter the Record where the prompt is found.

Translate Usage

Select the Translate Usage to be returned:

- Use Long Description
- Use Short Description

Exclude Prompt Field Values

- Insert All Values – Click button to add all LOV values to the excluded values list.
- Remove All Values – Click button to remove all LOV values from the excluded values list.

Test

Click the Test button to display a preview of the values that will be returned by the LOV when it is deployed as a prompt edit. The Test functionality uses the Exclude Prompt Field Values and Prompt Table Filter settings.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Constructing Rules

This section discusses how to:

- Use Rules Engine Search
- Use Rule Engine Groups Search
- Define Rule Options
- Add Variables to a Rule
- Add Criteria to a Rule
- Define Rule Groups

You create all Rules, including Functions and Triggers, using the Rules Engine Manager. Functionality available to create Rules is determined by:

- Skill Level (Expert or Developer)
- Rule Usage (Rule, Function, or Trigger)
- Rule Category
- Rule Group

Depending on user security as set up in Rule Category Security, the Rules Engine Manager can be accessed using different Skill Levels. Skill Levels that can be selected are Expert or Developer, with Developer having ability to create Rules where the logic is defined in Application Class PeopleCode. Experts create Rule logic in the Rules Engine Manager Define Rule Page.

Note. Currently, the Rules Engine Manager can only be used by users with a Skill Level of *Expert* or *Developer*. Oracle expects to add options for less complex Skill Levels in the future.

An Example Rule

To help describe the process of creating a new Rule, an example scenario of creating a new academic progression Rule is used throughout this chapter. This Rule evaluates data in the Academic Progress Tracker (APT). The example assumes that the Rule is run as part of a batch process at the end of a specific Academic Year; targeting all students in a specific Academic Program (Bachelor of Psychology) and student cohort. The purpose of the Rule is to find out whether students have obtained enough credits to progress to the from Year 1 to Year 2 of their academic program. The Rule is setup to run for a particular student.

Pages Used for Constructing Rules

Page Name	Definition Name	Navigation	Usage
Rule Search	SCC_RULE_SRCH_SEC	Set Up SACR, System Administration, Rules Engine, Rules Engine Manager, select the Search for a Rule radio button and click Next).	Search for existing Rules. You can also save Searches.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Page Name	Definition Name	Navigation	Usage
Rule Group Search	SCC_RULEGRSRCH_SEC	Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select the Search for a Rule Group radio button and click Next).	Search for existing Rule Groups. You can also save Searches.
Define Rule	SCC_RULE_GOV	Set Up SACR, System Administration, Rules Engine, Rules Engine Manager, select the Search for a Rule radio button and click Next, enter Search criteria, click Search button, select a Rule from results, click Next button). Set Up SACR, System Administration, Rules Engine, Rules Engine Manager, select the Add a New Rule radio button and click Next). Set Up SACR, System Administration, Rules Engine, Rules Engine Manager, select the Create a Rule from a Rule radio button and click Next, enter Search criteria, click Search button, select a Rule from results, click Next button). Set Up SACR, System Administration, Rules Engine, Rules Engine Manager, select the Create a Rule from a Rule Group radio button and click Next, enter Search criteria, click Search button, select a Rule Group from results, click Next button).	Define Rules.
Add a New Variable	SCC_RULE_VARD_SEC	Click the Add a New Variable button on the Define Rule page	Add Variables to a Rule.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Page Name	Definition Name	Navigation	Usage
Define Rule Groups Categories	SCC_RULEGR_CAT	Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select the Search for a Rule Group radio button and click Next, enter Search criteria, click Search button, select a Rule Group from results, click Next button, click Categories tab). Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select the Add a New Rule Group radio button and click Next, click Categories tab). Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select the Create a Rule Group from Rule radio button and click Next, enter Search criteria, click Search button, select a Rule from results, click Next button, click Categories tab). Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select the Create a Rule Group from Group radio button and click Next, enter Search criteria, click Search button, select a Rule Group from results, click Next button, click Categories tab).	Define valid Rule Categories for a Rule Group.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Page Name	Definition Name	Navigation	Usage
Define Rule Groups Definition	SCC_RULEGR_MGR	Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select the Search for a Rule Group radio button and click Next, enter Search criteria, click Search button, select a Rule Group from results, click Next button). Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select the Add a New Rule Group radio button and click Next). Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select the Create a Rule Group from Rule radio button and click Next, enter Search criteria, click Search button, select a Rule from results, click Next button). Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select the Create a Rule Group from Group radio button and click Next, enter Search criteria, click Search button, select a Rule Group from results, click Next button).	Define Rule Groups.
Define Rule Triggers	SCC_RULE_TRIG	Set Up SACR, System Administration, Rules Engine, Setup, Define Rule Triggers	Define Rule Triggers to determine from where in the system a Rule can be called.

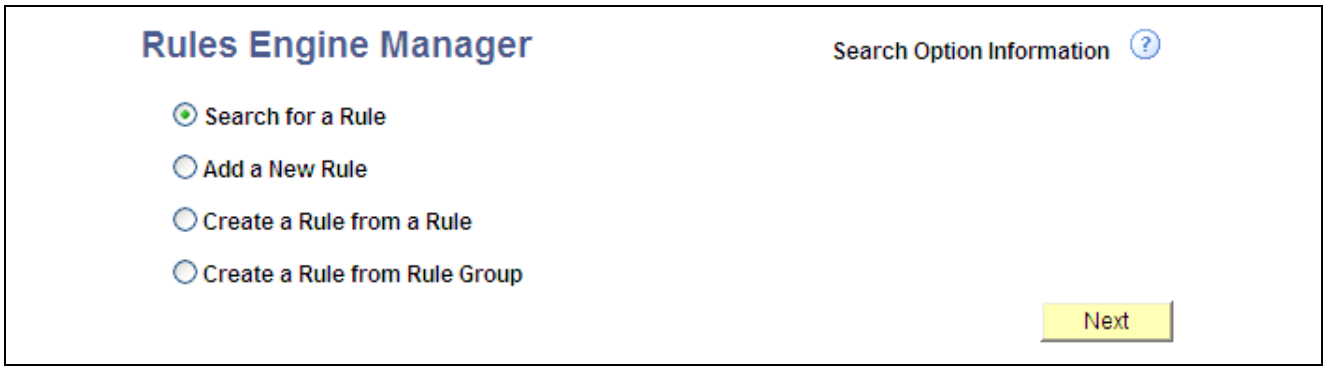
Using Rules Engine Search

Rules Engine Search is the starting point for the following activities:

- Searching for Rules.
- Adding new Rules.
- Creating a new Rule from an existing Rule.
- Creating a new Rule for a Rule Group.

Access the Rules Engine Search page (Set Up SACR, System Administration, Rules Engine, Rules Engine Manager).

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

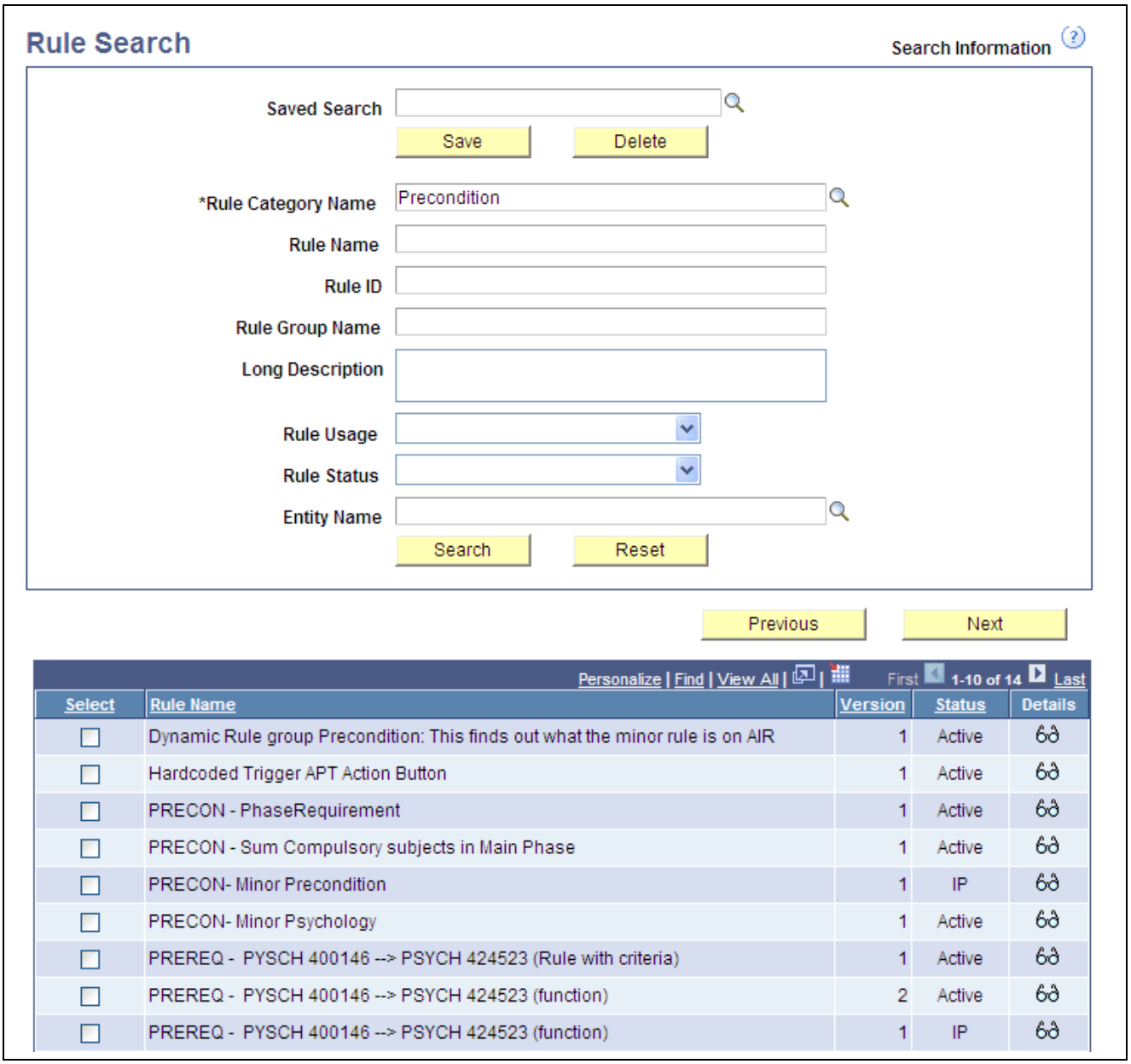


Rules Engine Manager Search page

Searching for a Rule

In the Rule Engine Manager search page, select Search for a Rule and click on the Next button to open Rules Search options page.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Rule Search Options page

Rule Category Name is a required field. If a unique Rule ID is known and the user has access to the Rule Category, the Rule Category is automatically selected and not required.

The following fields can be used with wildcard search option “%” and are *case-sensitive*:

- Rule Category Name
- Rule Name
- Rule Group Name
- Long Description

Click the Search button and any results are displayed in a grid below the Search fields. If there are no results, a message appears.

Click the Reset button to clear the Search fields.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To save the Search, enter a name in the Saved Search field and click the Save button.

To delete a Saved Search, enter the name in the Saved Search field and click the Delete button.

In the Search results grid, select the Select check box or click on the Details icon to select a Rule. Clicking the Details icon also displays a panel below the Search results grid displaying the Long Description, Rule Category, Rule Group Name, Rule Usage, and Entity Name of the Rule.

Click the Next button to open the selected Rule in the Rules Engine Manager and edit your Rule.

Click the Previous button to return to the Rules Engine Manager search page.

Adding a New Rule

In the Rules Engine Manager search page, select Add a new Rule and click on the Next button to open the Rules Engine Manager page where you can define the parameters of your new Rule.

For more information, see the *Constucting Rules* section of this chapter.

Creating a New Rule from an Existing Rule

In the Rules Engine Manager search page, select Create a Rule from a Rule and click on the Next button to open the Rule Search page.

When you have found the Rule you want to clone, click Next to open the Rules Engine Manager component. A new Rule is created with Rule Status *In Progress*, and it has inherited all the Variables, Criteria and evaluative statements from the Rule you selected to clone.

Note. The Rule Name is also copied. As a first step, you should rename the Rule to avoid confusion.

Creating a New Rule from a Rule Group

In the Rules Engine Manager search page, select Create a Rule from a Rule Group and click on the Next button to open the Rule Group Search page, which is similar to the Rule Search page.

When you have found the Rule Group to which you want to add a Rule, click Next to open the Rules Engine Manager component. A new Rule is created with Rule Status *In Progress*, and it has inherited the Input and Output Variables, Entity Profile, Rule Category Name, Rule Usage, Entity name, and Skill Level of the Rule Group to which it has been added. This means that the Rule Category Name, Rule Group Name, Rule Usage, Entity Name, and Skill Level are predetermined and cannot be altered for this new Rule.

Note. The Rule Name is also copied from the Rule Group Name. As a first step you should rename this Rule as to avoid confusion.

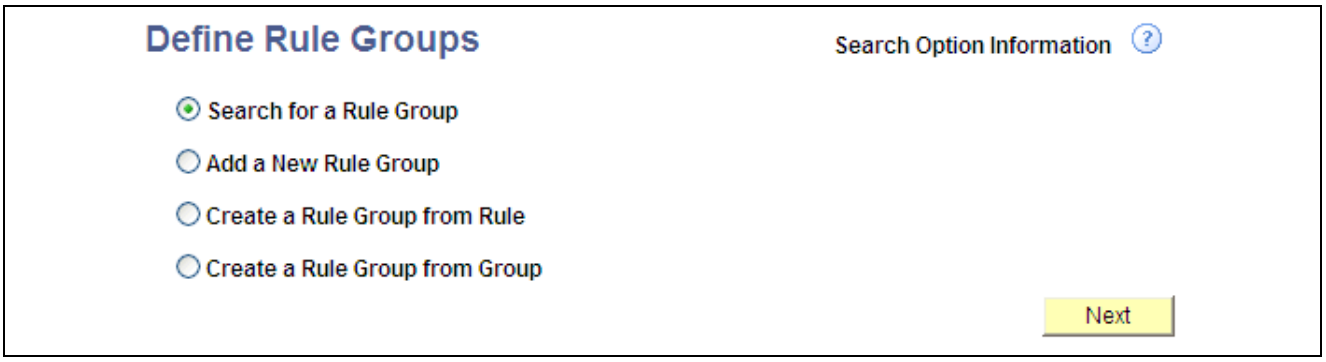
Using Rule Groups Search

Rule Groups Search is similar to Rules Engine Search and is the starting point for the following activities:

- Searching for a Rule Group.
- Adding a new Rule Group.
- Creating a new Rule Group from an existing Rule.
- Creating a new Rule Group from an existing Rule Group.

Access the Define Rule Groups Search page (Set Up SACR, System Administration, Rules Engine, Define Rule Groups).

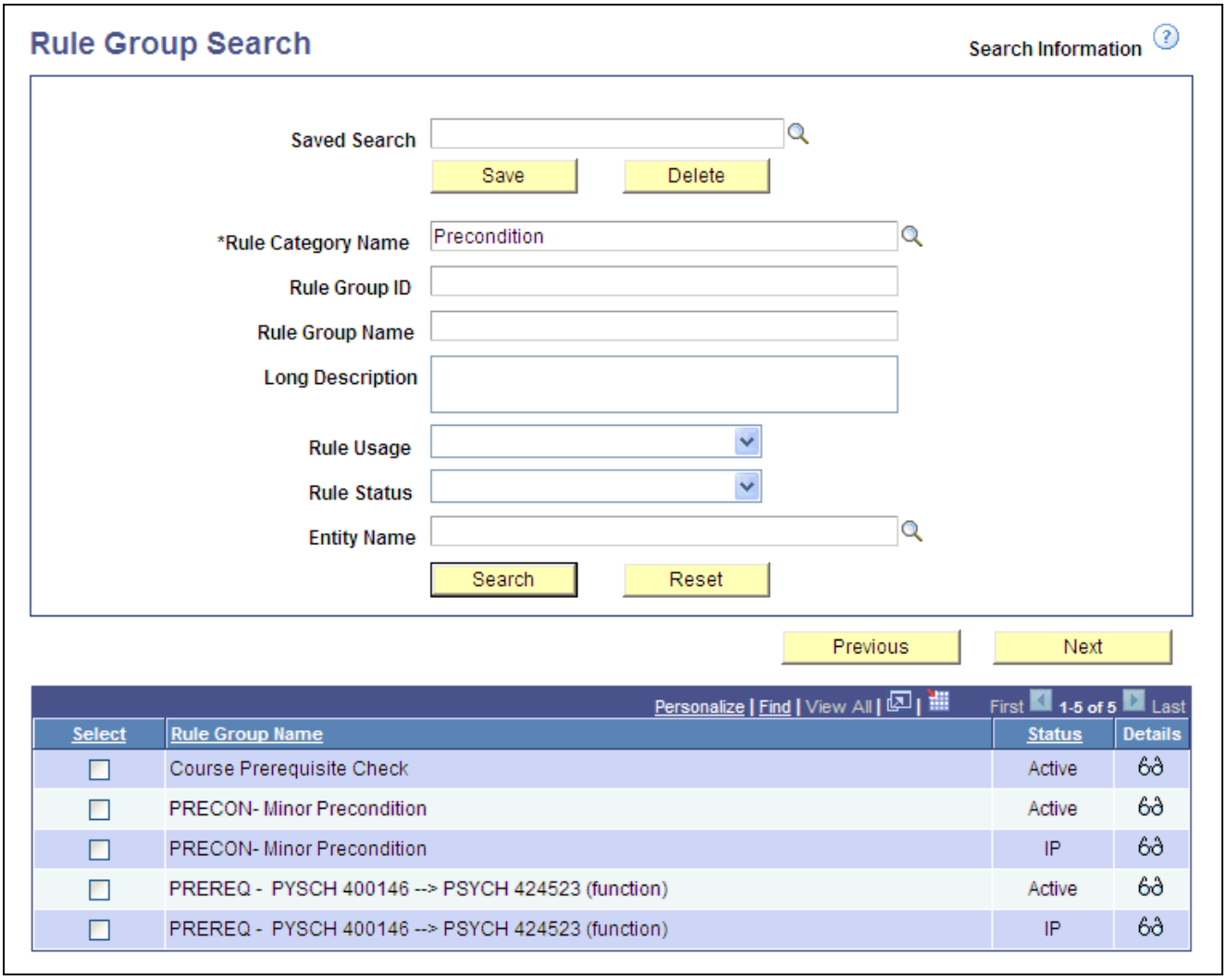
 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Define Rule Groups Search page

Searching for a Rule Group

In the Rule Group search page, select Search for a Rule Group and click on the Next button to open Rule Group Search options page.



Rule Group Search Options page

Rule Category Name is a required field. If a unique Rule Group ID is known and the user has access to the Rule Category, the Rule Category is automatically selected and not required.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The following fields can be used with wildcard search option “%” and are *case-sensitive*:

- Rule Category Name
- Rule Name
- Rule Group Name
- Long Description

Click the Search button and any results are displayed in a grid below the Search fields. If there are no results, a message appears.

Click the Reset button to clear the Search fields.

To save the Search, enter a name in the Saved Search field and click the Save button.

To delete a Saved Search, enter the name in the Saved Search field and click the Delete button.

In the Search results grid, select the Select check box or click on the Details icon to select a Rule. Clicking the Details icon also displays a panel below the Search results grid displaying the Long Description, Rule Category Name, Entity Name, Rule Usage, and all Input and Output variables as they are key features of the Rule Group Template.

Click the Next button to open the selected Rule Group in the Rule Groups Manager and edit your Rule Group.

Click the Previous button to return to the Define Rule Groups search page.

Adding a New Rule Group

In the Define Rule Groups page, select Add a New Rule Group and click on the Next button to open the Rules Groups Definition page where you can define the parameters of your new Rule Group.

For more information, see the *Defining Rule Groups* section of this chapter.

Creating a New Rule Group from an Existing Rule

In the Define Rule Groups search page, select Create a Rule Group from Rule and click on the Next button to open the Rule Search options page.

When you have found the Rule you want to base your new Rule Group on, click Next to open the Rule Groups Manager component. A new Rule Group is created with Rule Group Status *In Progress*, and it has inherited all the Variables, Entity Name, Skill Level and Rule Usage from the Rule you selected.

Note. The Rule Name is also copied. As a first step, you should rename the Rule Group Name to avoid confusion.

Creating a New Rule Group from an Existing Rule Group

In the Define Rule Groups search page, select Create a Rule Group from Group and click on the Next button to open the Rule Group Search page, which is similar to the Rule Search page.

When you have found the Rule Group that you want to clone for a new Rule Group, click Next to open the Rule Group Manager component. A new Rule Group is created with Rule Group Status *In Progress*, and it has inherited the Input and Output Variables, Entity Profile, Rule Category Name, Rule Usage, Entity Name, and Skill Level of the Rule Group chosen to clone.

Note. The Rule Group Name is also copied from the selected Rule Group. As a first step you should rename this Rule as to avoid confusion.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

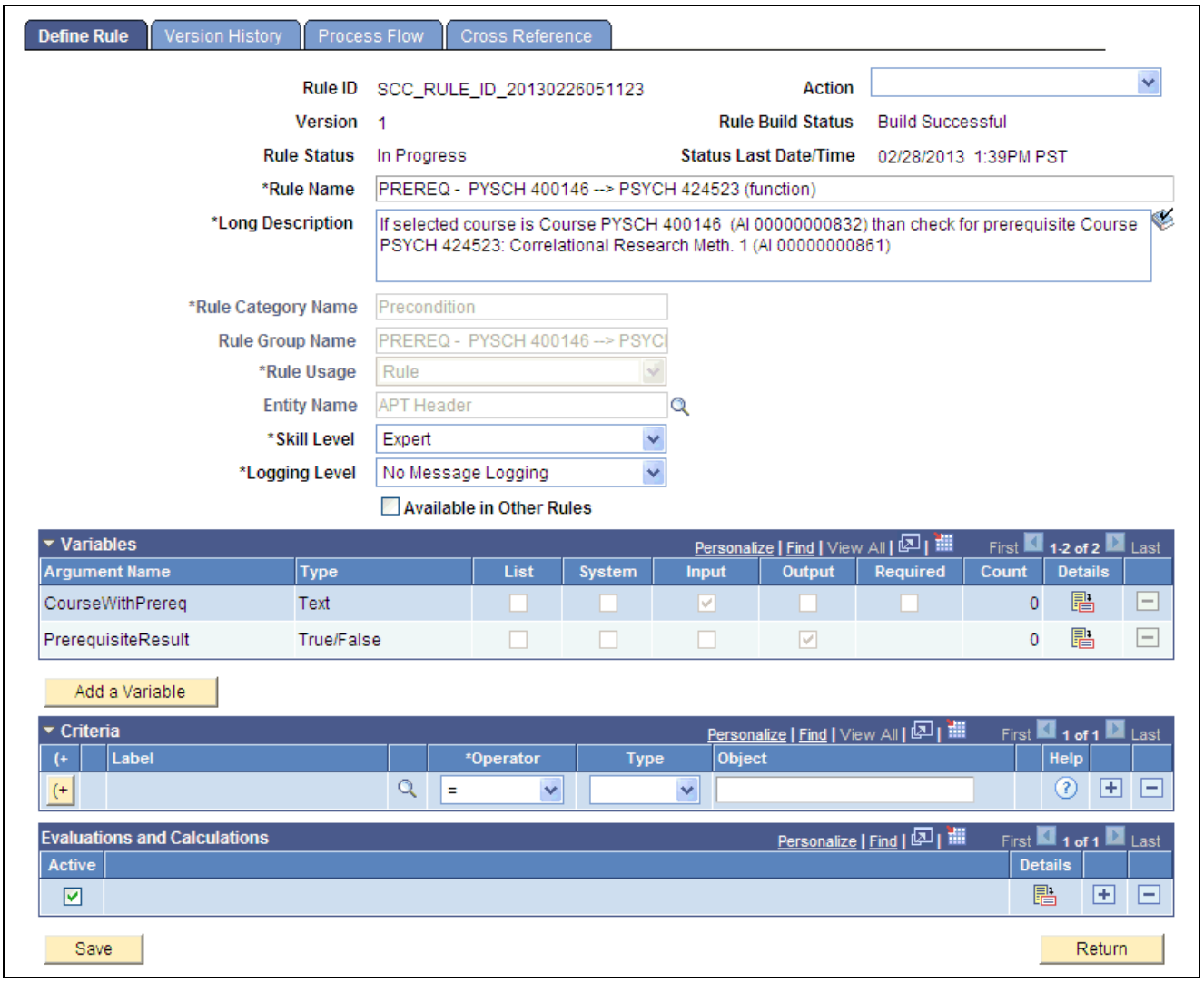
For more information on how to create and edit Rules,

For more information, see the *Defining Rule Groups* section of this chapter.

Creating Functional Rules

The section describes how to create functional Rules.

Access the Access the Rules Engine Manager Define Rule page (Set up SACR, System Administration, Rules Engine, Rules Engine Manager, select Add a New Rule, Define Rule).



Define Rule page

Action

Note. The list of available Actions is dynamic depending upon the Rule Status.

Select an Action for this Rule Group:

- *Activate Rule* – An *In Progress* Rule can be activated using the *Activate Rule* Action.
- *Build Rule* – Opens the Build Rule window.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

- *Create New Rule from Rule* – This action clones the current Rule and creates a new Rule using the values from this Rule.

  Note. The Name of the new Rule is copied exactly from the current Rule so your first action should be to rename the Rule.

- *Delete Rule* – Rules can be deleted. When choosing this Action, a warning is displayed, and, after choosing "OK", the Rule is deleted.
- *Inactivate Rule* – An *In Progress* Rule can be inactivated using the *Inactivate Rule* Action.
- *Test Rule* – Opens the Test Rule window.

  For more information, see the *Testing Rules* section of this chapter.

Rule ID

Displays a unique ID generated by the system. When adding a new value the default value is *NOID*. The unique ID is generated when the Rule is saved. The Rule ID is created by combining prefix SCC_RULE_ID_ with the system date and time stamp in format YYYYMMDDHHMMSS.

Version

Displays the Version number of the Rule.

Rules are automatically versioned. The first version of a Rule is 1. Only one version of a Rule can be active at any given time. The Action dropdown can be used to inactivate an *Active* Rule and to create a new Rule Version. When creating a new Version, the previous Version automatically becomes *Inactive*.

Rule Status

Displays the Rule Status

- Active
- Inactive
- In Progress. This is the initial Rule Status assigned when creating a Rule.

Rule Build Status

Displays the Rule Build Status. Values are:

- *Rule Not Built* – The Rule has never been built.
- *Build Not Current* – A change has been made to this Rule since the last build which may warrant a rebuild.)
- *Build Failed* – The build/compilation process for the Rule ran to error. Consult the process scheduler message log to view the error.
- *Build Successful* — The Rule was built without errors.

Status Last Date/Time

Displays the Last Date/Time of the Rule Build Status in native format including an indication of time zone. This field does not display until after an attempt to build this Rule.

Rule Name

Enter a Rule Name. The Rule Name is used when searching for a Rule and for display.

Long Description

Enter descriptive text explaining the function of the Rule.

Rule Category Name

Enter the Rule Category Name. The Rule Category restricts access to Rule Usages, Rule Categories, and Entities.

Available Rule Categories are limited to those for which you have been authorized. This is determined by a Role or our User ID.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

If you have access to multiple Rule Categories, you should choose the Rule Category that is most appropriate considering the data that the Rule needs to access and any Functions that the Rule may need to call. Functional Expert users who create Rules may not necessarily be familiar with how to create Rule Categories so should be provided with instructions by Rules Engine Administrators about the Rule Categories for which they have been authorized. For example, depending on your institution's policy, Functional Experts may only be authorized to use a single Rule Category.

Rule Group Name

Enter a Rule Group Name if this Rule needs to conform to a specific template with predefined input and output Variables and a predetermined Base Entity.

Rule Usage

Select a Rule Usage of *Function*, *Rule*, or *Trigger*. The available Rule Usages are limited to those for which the user has been authorized through Rule Category setup.

The selected Rule Usage impacts availability of Rules Engine Manager functionality as follows:

- Rule: The Criteria grid, the Variables Grid, and Evaluations and Calculations grids are available.
- Function or Trigger: The Variables grid and Evaluations and Calculations grids are available. The Criteria grid is not available.

Selection of Rule Usage needs to consider how this Rule will be used and what tasks it needs to perform.

- Rule – Rules can be used stand-alone (when executed from the Rules Tester), called from other Rules, or called from a Batch Process. Rules need criteria in order for the correct data to be selected from the system.
- Trigger – Triggers are tied to a specific system event.
- Function – A Function is a reusable Rule that performs a specific task, often performed in the context of a larger Rule. For example, when creating a Rule that evaluates whether Total Credit is greater than a specific Credit amount, first the Sum of all Credit needs to be established. The "Add" Function could be used to do nothing other than add retrieved Credit to a Credit total. The "Add" Function could be used to add any numeric value in the system, not just Credit, making it very reusable. A Function does not need specific criteria in order to determine an exact set of data. In this Rule Creation example, it only needs a numeric input to perform its task.

  When a Function uses a Base Entity, information to determine an exact set of data is passed from the calling Rule to the called Function using Contextual Reference.

  See the *Understanding Contextual Referencing* section of this chapter.

  This example is using a Rule Usage of *Rule* and Skill Level of *Expert* to demonstrate the use of Criteria to identify a specific set of data. The Criteria in this example are used to identify the exact student Academic Progress Tracker data from the Base Entity as chosen by selecting an Entity Name. If data from other application areas is needed, it can be collected using other Functions.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Entity Name

Enter the Entity to use as the Base Entity for the Rule. The Base Entity controls which application data you have access to when creating a *Rule*, *Function*, or *Trigger*. The Entities available are limited to those for which user has been authorized by attaching Entity Profiles to Rule Categories in the Rule Category setup.

Identifying which data to use by designating the Base Entity can be considered the "starting point" of the creation of a functional Rule. Further, using the designated Base Entity with the Rule Criteria makes it possible to identify the exact record to use in the Rule. For a Function that uses a Base Entity, the exact data needed is passed from caller to called Function using Contextual Reference.

In this example, a Rule is being created that look at Student Progression; therefore, the Rule needs to access Student Results Data as stored in the Academic Progress Tracker (APT) records. The APT data is represented in the Entity *APT Header*. By selecting the APT Header as the Base Entity, you can access all of the data associated with the APT Header Entity Tree. This means that you have access to all logical child Entities in that Tree structure like APT Program of Study, APT Course List, and APT Course as well as other data associated with those Entities.

As with Rule Categories, Functional Expert users should be provided with instructions on selecting Base Entities. Also consider that since the available Base Entity is determined by the selected Rule Category, it could be that only one Base Entity is presented for selection.

For more information, refer to the *Defining Rule Category Security* and *Understanding Contextual Referencing* sections in this chapter and

See [PeopleSoft Campus Community 9.0 Fundamentals PeopleBook]Setting Up Entity Registry.

Skill Level

Select the Skill Level required for creating this Rule:

- *Expert* – This Skill Level has access to all features delivered with Rules Engine.
- *Developer* – This Skill Level has limited access to Rules Engine features. A user accessing the Rules Engine Manager and selecting *Developer* Skill Level is assumed to create Rules using Application Package PeopleCode. The Criteria Grid and Evaluations and Statements Grid are not available with *Developer* Skill Level.

In this example, a functional Rule is being created and there is no "coding" occurring. Therefore, a Skill Level of *Expert* is being used to demonstrate the Statement, Operator, Function, and Variable features.

Rule Application Class

Note. This field is available and required when Skill Level is *Developer*.

Enter or select a Rule Application Class. A Rule Application Class can be selected from a list of Extension Application Classes.

Warning! Developer Rules should be built by Developers/Programmers who have extensive experience with Application Package/Class PeopleCode programming. Developers working with the Rules Engine should be familiar with Entity Registry concepts.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Logging Level

Select the Logging Level to use when testing this Rule:

- *Error Messages*
- *Informational Messages*
- *No Message Logging*
- *Trace Logging*
- *Warning Messages*

Note. Oracle recommends using the *Trace Logging* option to see correct trace results when testing Rules.

Available in Other Rules

Select this check box if you want to allow this Rule or Function to be available for use by other Rules or feature functionality. If you do select this check box, it is recommended that you do so *after* adding Variables and Criteria to the Rule and finalizing the functionality for your Rule.

Note. Save the Rule after completing the first portion of the Definition tab (everything above the Available in Other Rules check box) *and before* adding Variables and Criteria. At this point the Rule Status is *In Progress*, and the Rule can still be changed and retested.

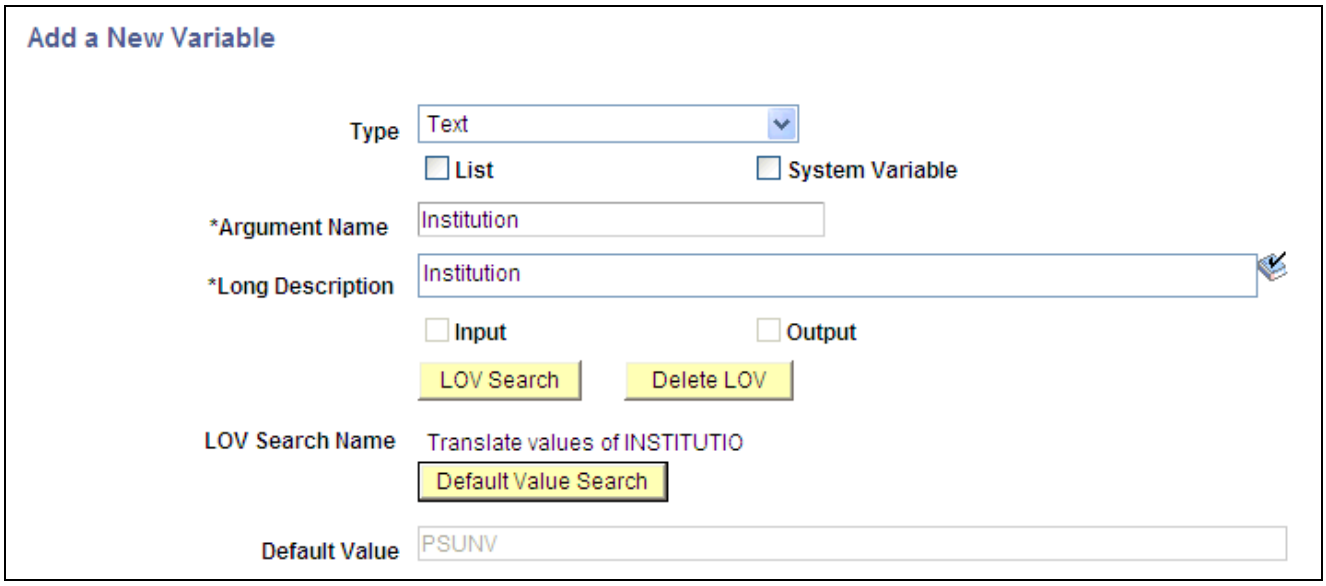
Adding Variables to a Rule

Once Rule options are defined, you can add the Variables you want to use in the Rule. The Variables grid on the Rules Engine Manager Definitions tabs is display-only. The grid shows whether variables are in use and whether they are used as input, output or required. Variables can be created before starting to incorporate evaluations or calculations or added on the fly.

Since the object of this example Rule is to determine whether students may or may not progress from Year 1 to Year 2 within an Academic Program, you already know that you want to return the Progression Status as *output*. You also know that you want to ensure that the process that calls this Rule can pass the correct parameters to retrieve a specific student. That is our *input*.

To add a Variable, access the Add a New Variable page (click the Add a Variable button on the Rule Definition page).

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Add a New Variable page

Type

Select the Variable Type that you want to add:

- *Data Set*
- *Date*
- *Datetime*
- *Number*
- *Text*
- *Time*
- *True/False*

Type is displayed in the Variables grid and can be selected when viewing or adding Variables.

List

Select this check box if the variable is a List variable which can contain multiple values. Use this option in combination with Type to obtain a list of specific values. List is displayed in the Variables grid and can be selected when viewing or adding Variables. Selecting this check box activates the Default Values button and the Default List Values field.

System Variable

Select this check box if the variable is a System Variable predefined in setup outside of this Rule. System Variable is displayed as *System* in the Variables grid and can be selected when viewing or adding Variables. Selecting this check box activates the System Variable Search button and the System Variable Name field.

Argument Name

Enter an Argument Name for Variable. The Argument Name is used when searching for a Variable and for display.

Long Description

Enter descriptive text explaining the function of the Variable..

Input

Select this check box if the variable is an Input Variable. Input is displayed in the Variables grid and can be selected when viewing or adding Variables.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Required

Select this check box if the variable is required. Required is displayed in the Variables grid and can be selected when viewing or adding Variables.

Output

Select this check box if the variable is an Output Variable. Output is displayed in the Variables grid and can be selected when viewing or adding Variables.

Default Value

Click this button to open the Create Default List Values page. Add List Values in the Object column and click the OK button.

Note. This button is available when the *List* option is selected.

To remove a List Value, select the corresponding Remove check box for an Object and click the Remove Selected Values button.

Note. If the *List* option is not selected, Default Value is an edit field. A Default Value can be entered here for this Variable.

Default List Values

Displays values for this Variable created using the Create Default List Values page.

System Variable Search

Click this button to open the Prompt for Rules Engine System Variables page. This button only appears when System Variable is selected above.

System Variable Name

Displays the selected System Variable Name. This field only appears when System Variable is selected above.

Data Set Search

Click this button to open the Prompt for Entity page.

Search on a Field Name of *Data Set ID*, *Entity Name*, or *Entity Profile Name*.

Results display a hyperlinked Entity Profile. Click the hyperlink to select an Entity. This returns you to the Add a New Variable page with the Data Set Name and the Data Set Properties grid displayed. The Data Set Properties grid displays the Property Type and Label of each Variable in the Data Set

Data Set Name

Displays the Data Set Name.

LOV Search

Select the button to open the Prompt for LOV Searches page.

Search on a Field Name of *Description*, *Dropdown Prompt Filed*, *Edit Table*, *LOV Context*, or *LOV Unique ID*.

Results display a hyperlinked Descr (description). Click the rink to select an LOV. This returns you to the Add a New Variable page with the LOV Search Name displayed.

LOV Search Name

Displays selected LOV.

Default Value Search

Click this button to open the Prompt for (selected LOV) page.

Search on a Field Name of *Description* or *Value*.

Results display hyperlinked Values.

Click the link to select a Default Value. This returns you to the Add a Variable page with the Default Value displayed.

Default Value

Displays Default Value.

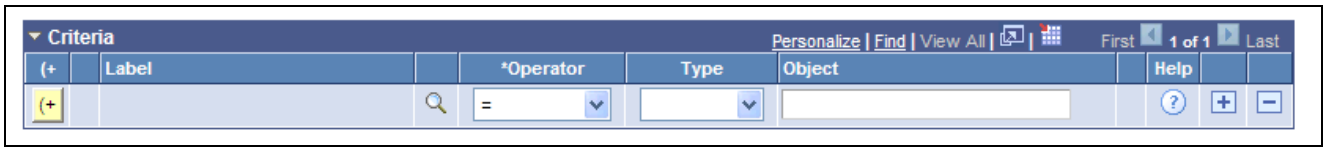
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Adding Criteria to a Rule

Once Rule options are defined and Variables are added for a Rule Usage of *Rule*, you can add Criteria to identify data from the chosen Base Entity. In the example being followed in this chapter, you want to add all criteria needed to identify and select the correct APT instance record for a particular student.

Note. If you are creating a Rule with a Rule Usage of *Function*, there is no option to add Criteria. For more information, see the *Understanding Contextual Referencing* section of this chapter.

To add Criteria, expand the Criteria group box by clicking the arrow to the left of Criteria in the group box header.



Criteria Grid – Initial Row

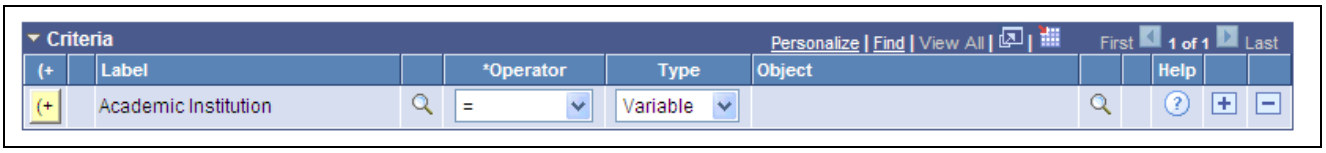
Then, click the Search icon to the right of the Label field to open the Prompt for page where you can select an Entity Property.

Property	Entity	Type	Key
ID	APT Header	String	*
Academic Institution	APT Header	String	*
APT Instance	APT Header	Number	*
Academic Career	APT Header	String	
Student Career Nbr	APT Header	Number	
Academic Plan	APT Header	String	
Academic Item ID	APT Header	String	
APT Status	APT Header	String	
Advisor Approval Status	APT Header	String	
Cohort ID	APT Header	String	
Cohort Term	APT Header	String	
Requirement Term	APT Header	String	
Locked	APT Header	String	
Created By	APT Header	String	
Created	APT Header	Datetime	
Updated By	APT Header	String	
Last Update Date/Time	APT Header	Datetime	

Prompt for page

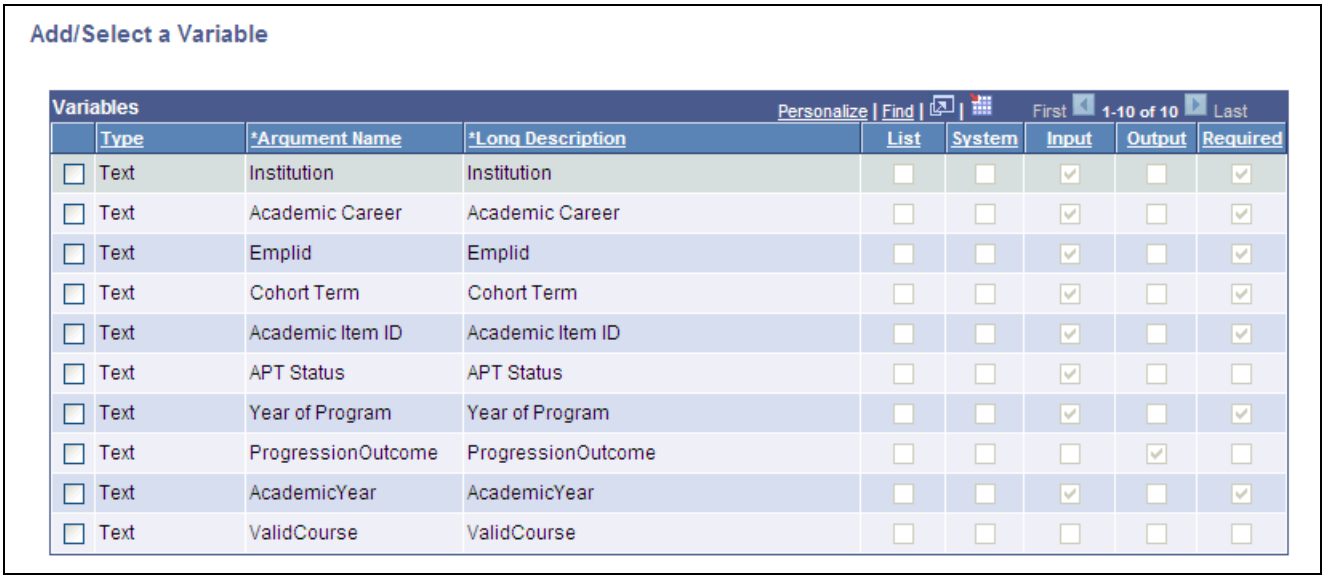
Click the *Academic Institution* link to select it as a Property. The Prompt for page closes, and you return to the Criteria grid with the Label column in the first row populated with *Academic Institution*. The Operator you want to use is = and the Variable Type you want to use is *Variable*, both of are already populated by default in this case. Since Type Variable is selected, the Search icon appears to the right of the Object field.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Example of Criteria Grid – Label Populated

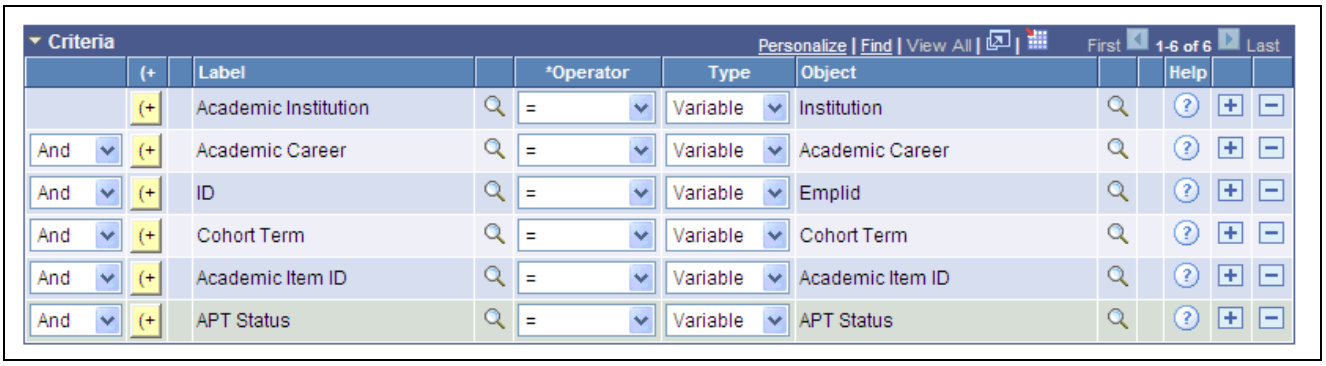
Click the Search icon to the right of the Object field to open the Add/Select Variable page.



Add/Select a Variable page

Select the check box next to the Variable you want to add, and click the Add button.

Once you've selected all the Criteria you want to add, the Criteria grid looks something like this:



Example of Criteria Grid

In this example, there is an assumption that a batch process is responsible for supplying the correct Variables to our Rule. This is why the Variables selected for the Criteria grid are Input Variables.

Here is more information about fields in the Criteria grid:

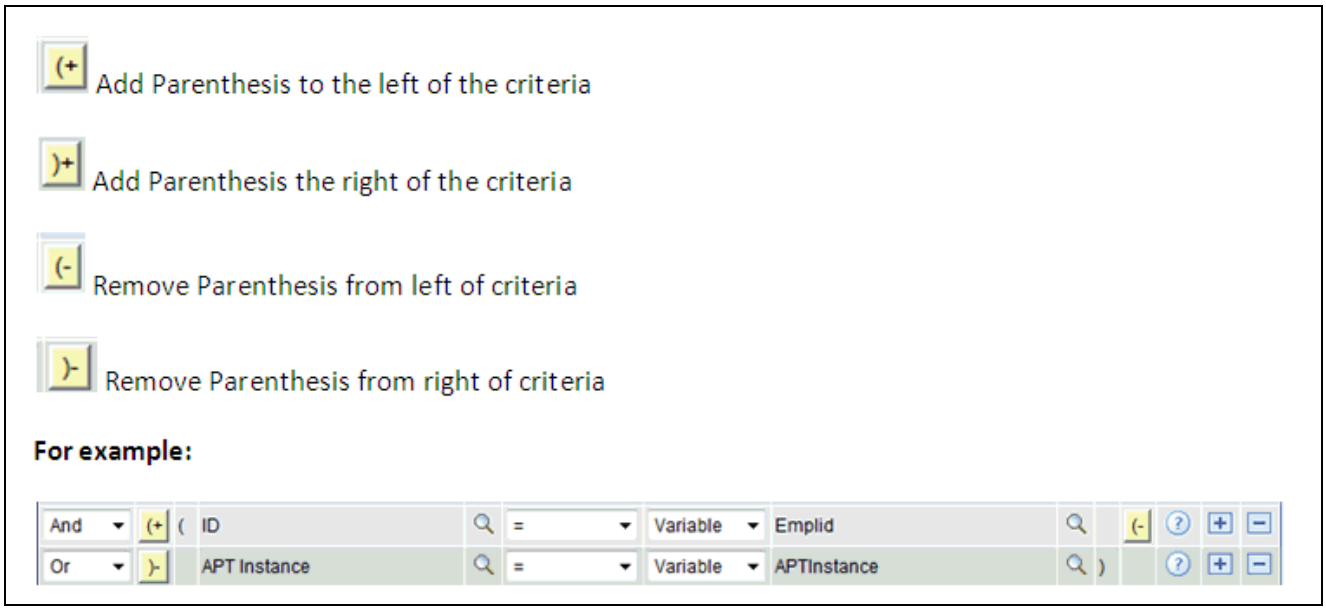
Label

Displays the name of the Property. Clicking the Search icon to the right of the Label field opens the Prompt for page where you can select from a list of properties from the Base Entity that is added in field Entity Name on the Rule Definition page.

Connectors and Parentheses

In the left-most columns of the Criteria grid are the AND/OR connectors and parenthesis. Select these to create complex (nested) Select Criteria.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Example of Connectors and Parentheses

Operator

Select an Operator to relate the Property Label to the Type/Variable. The Operator is a symbol or function used to express a mathematical function or logical action. The following Operators can be used when adding a Criteria line:

- < (less than)
- <= (less than or equal to)
- <> (not equal to)
- = (equal to)
- > (greater than)
- >= (greater than or equal to)
- *Exists* and *Not Exists* – The *Exists* and *Not Exists* operators can be used in combination with Types Text, Variable, Numbers, Date, Datetime, and Time. When using these Operators, an argument only exists on the left side of the operator, the Label. These Operators work differently based on the Object type:
  - If the Object is a string, Exists is true if Argument 1 is a non-blank value.
  - If the Object type is a number, Exists is true then whenever a value greater than 0 is found.
  - If the Object type is a date, time or date time, Exists is true if the value is anything other than null.
  - If the Object type is Boolean, the operator will always return true since true and false are both valid values.
- *In* and *Not In* – The *In* and *Not In* operators can only be used in combination with Type of *Variable*. The Variable in question needs to be a Variable of type List.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

- *Like* and *Not Like* – The *Like* and *Not Like* operators can be used in combination with Type Text and Variable. Numeric values in string fields can also be evaluated:
  - "%" (percent sign) – Use at the beginning or end of string to replace any length value in the comparison.
  - "_" (underscore) – Use to replace a single alphabetic value in the comparison.
  - "#" (hash or pound sign) – Use to replace a single numeric value in the comparison.
- *AsOfDate* – This operator automatically performs Effective Date selection as per standard Effective Dated processing. The *AsOfdate* operator is only available when the Base Entity has an Effective Date Key field.
- *FirstSeq* and *LastSeq* – These operators allow you to select a minimum or maximum effective sequence for those Entities that have an effective sequence Key field. The *LastSeq* and *FirstSeq* are only available on Base Entities with key field Effective Sequence.

Type

Select the Type of Object Property to be compared to the Label Property. The chosen Operator controls what Types are available.

For the Operators *Exists* and *Not Exists*, no Type can be chosen.

For the Operators *In* and *Not In*, only Variables of Type *List* can be chosen.

Here are the available types:

- *Date*
- *Datetime*
- *Number*
- *Property*
- *Text*
- *Time*
- *True/False*
- *Variable*

Object

The Object field behaves differently depending on which Type or Operator is chosen.

The Object field is an open edit field except in the following cases:

- The Object field is not available when the Operator is *Exists* or *Not Exists*.
- The Object field changes to a prompt when the Type is *Variable*, *True/False*, *Property*, or when LOV prompting has been enabled for properties.

  For more information, see the *Defining Lists of Values for Rules Engine Variables* section of this chapter.

Help

Hover your cursor over the Help icon (question mark) to open a popup window for information on how to use the selected Type and/or Operator.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Defining Rule Groups

This section covers defining Rule Groups.

Understanding Rule Groups

Rules that share the functional purpose and have similar functionality, input and output can be grouped together in a Rule Group. For Rules to share the same Rule group, they must:

- Be based on data in the same Entity Tree and use the same Entity Registry item as the basis for that Rule. This Entity is known as the Base Entity.
- Share the same required input parameters.
- Share a subset of non-required input parameters.
- Share the same output parameters and can pass back required output.
- Belong to the same Rule Category, giving them the same security Rules.

One advantage of combining Rules in Rule Groups is that it allows you to call all Rules associated with a Rule Group dynamically. You would not have to know before the fact which Rules you are calling. Rule Groups that have been created can be called from other Rules, Triggers or Functions in the Rules Engine Manager using the “CALL DYNAMIC RULE GROUP” Statement. In short, by allowing Rules to call Rules from Rule Groups dynamically without specifying Rules or Functions directly, it is possible to dynamically call one or more Rules with similar functionality.

For example, an institution has created an Academic Item Registry program with courses from which the student can choose. For all those courses that require a prerequisite, a functional Rule is created that takes the course selected as input and checks whether the student meets the prerequisite requirement. When students select courses to add to their Academic Progress Tracker (APT), the prerequisite is checked, and, based on the outcome, a message is displayed to the student indicating whether or not he or she may add the course to their APT.

Example Rules for this scenario could be:

- Students must have completed Introduction to Calculus or Elementary Algebra before attempting Advanced Calculus
- Students must have completed Introduction to Psychology and have completed 10 credits from the “Human Studies” Course List before attempting Advanced Psychology.

Another advantage to creating Rules in the same Rule Group is the ability to control that the input and output for all Rules in the Rule Group is the same.

Each of these Rules needs to take the selected Course as input as well as information from the program of study and the student. All Rules above would return a similar outcome of true or false as well as a message that can be displayed. The Rules can be grouped together in the same Rule Group.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

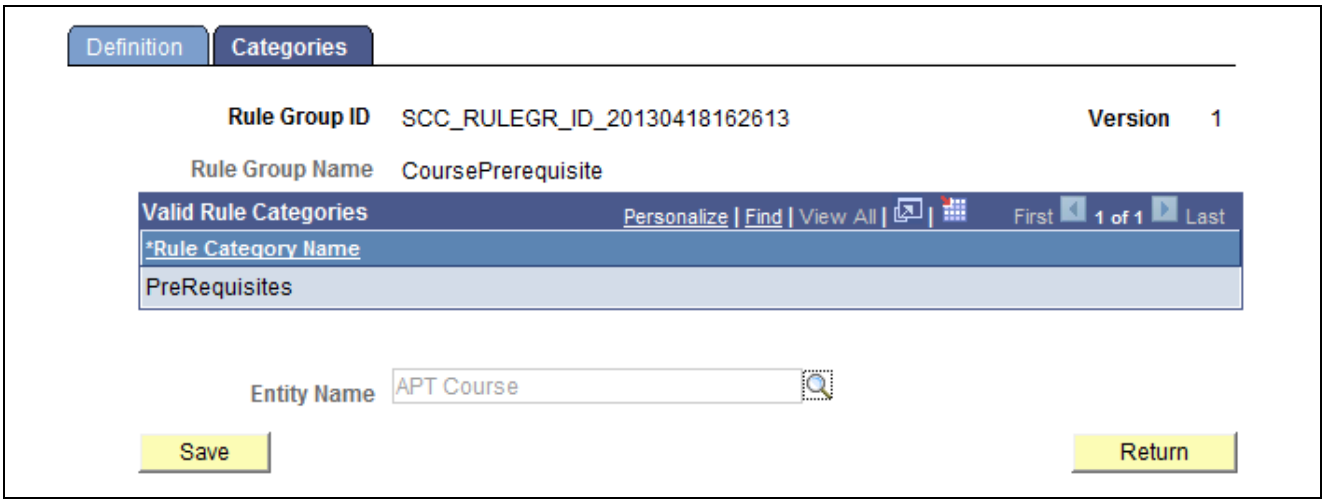
Note. Rule Groups should be created by Rules Engine Experts or Developers. The settings used in Rule Groups are enforced for any Rule added to the Rule Group. Once a Rule has been added to a Rule Group, the Rule's input and output parameters cannot be altered; however, it is still be possible to add to and alter Statements in the Evaluations and Calculations grid. Also, Rule Groups with Active Rules attached cannot be changed.

You may find it challenging to define a new Rule Group with input and output Variables and specifications for the Base Entity if you have not built the actual Rule that will be using the Rule Group . When building Rules, some experimentation is in order, and it is not always clear before the fact what exactly the input and the output of a Rule needs to be. Rule logic is optimized during the build process, and input or output Variables may need to be added based on new specifications or insights. Therefore, it may not be possible to define a template for input and output before any Rule has been built.

Rule Groups can also be created from Active Rules using the Rules Engine Manager Action *Create Rule Group*. Rule Groups can also be created using one of the available Rules Engine Search options.

Defining New Rule Groups

Access the Rule Groups Manager Categories page (Set up SACR, System Administration, Rules Engine, Define Rule Groups, select "Add a New Rule Group", select the Categories tab).



Define Rule Groups Categories page

Rule Category

Enter the Rule Categories for this Rule Group. Rule Groups are tied to one or more Rule Category Names.

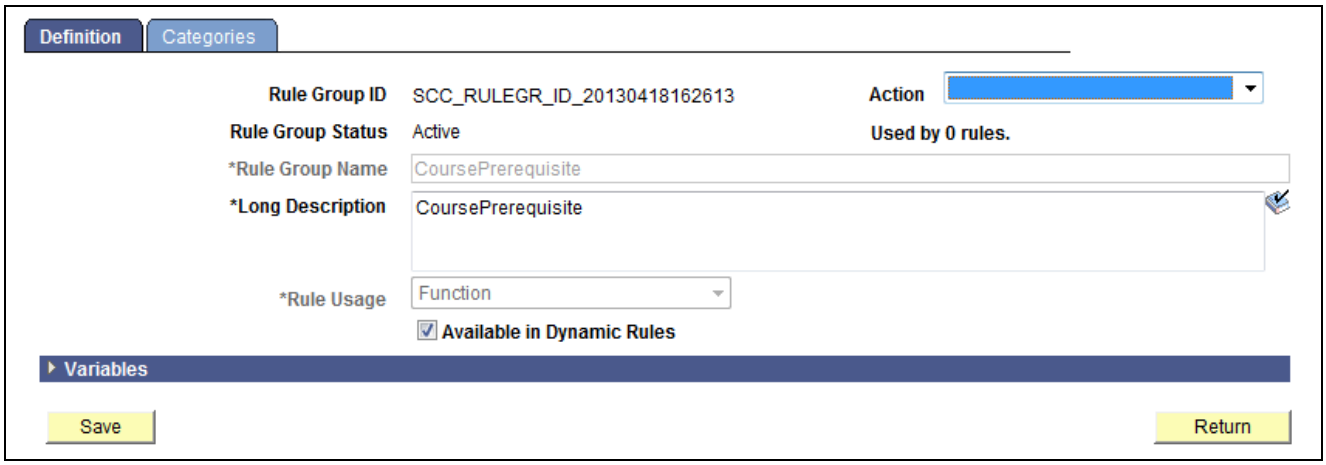
Note. Assign categories first before selecting a Base Entity in the Definition tab.

Entity Name

Enter the Entity for the Rule Group. The Entity entered here is the Default Base Entity for any Rules added to this Rule Group. The Base Entity controls which application data You have access to when building the Rule, Function, or Trigger.

Access the Rule Groups Manager Definition page (Set up SACR, System Administration, Rules Engine, Define Rule Groups, select "Add a New Rule Group", select the Definitions tab).

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Define Rule Groups Definition page

Action

Select an Action for this Rule Group:

- *Activate Rule Group* – In Progress Rule Group can be Activated using the Activate Rule Group option.
- *Create Rule Group from Group* – This action clones the current Rule Group and creates a new Rule Group using the values from this Rule Group.

  Note. The Name of the new Rule Group is copied exactly from the current Rule Group so your first action should be to rename the Rule Group.

- *Delete Rule Group* – Rule Groups can be deleted. When choosing this option a warning is displayed first. After choosing OK the Rule is Deleted. This is only available if no Rules are attached to this Rule Group
- *Inactivate Rule Group* – In Progress Rule Group can be Inactivated using the Inactivate Rule Group option. Only possible if zero Rules are attached to this Rule Group

Rule Group ID

Displays a unique ID generated by the system. When adding a new value the default value is *NOID*. The unique ID is generated when the Rule Group is saved. The Rule Group ID is created by combining prefix SCC_RULEGR_ID_ with the system date and time stamp in format YYYYMMDDHHMMSS.

Rule Group Status

- In Progress
- Active

Used By X Rules

Displays the number of Rules associated with this Rule Group; includes Active and Inactive Rules.

Rule Group Name

Enter a Rule Group Name. The Rule Group Name is used when searching for a Rule Group and for display.

Long Description

Enter descriptive text explaining the function of the Rule Group..

Rule Usage

- Function
- Rule
- Trigger

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Available in Dynamic Rules Select this check box if you want this Rule Group to be available for use by the statement CALL DYNAMIC RULE GROUP.

Note. If the Allow Changes to Active Rules check box on the Rule Engine Install Options page is not selected, the Available in Dynamic Rules check box cannot be selected here.

Dynamic Rule Variable A Dynamic Rule Variable is always created when a Rule Group is created. This Variable is used when calling Rules using the Dynamic Rule Group statement to call the required Rule ID.

Add a Variable Select this button to open the Add a New Variable page.

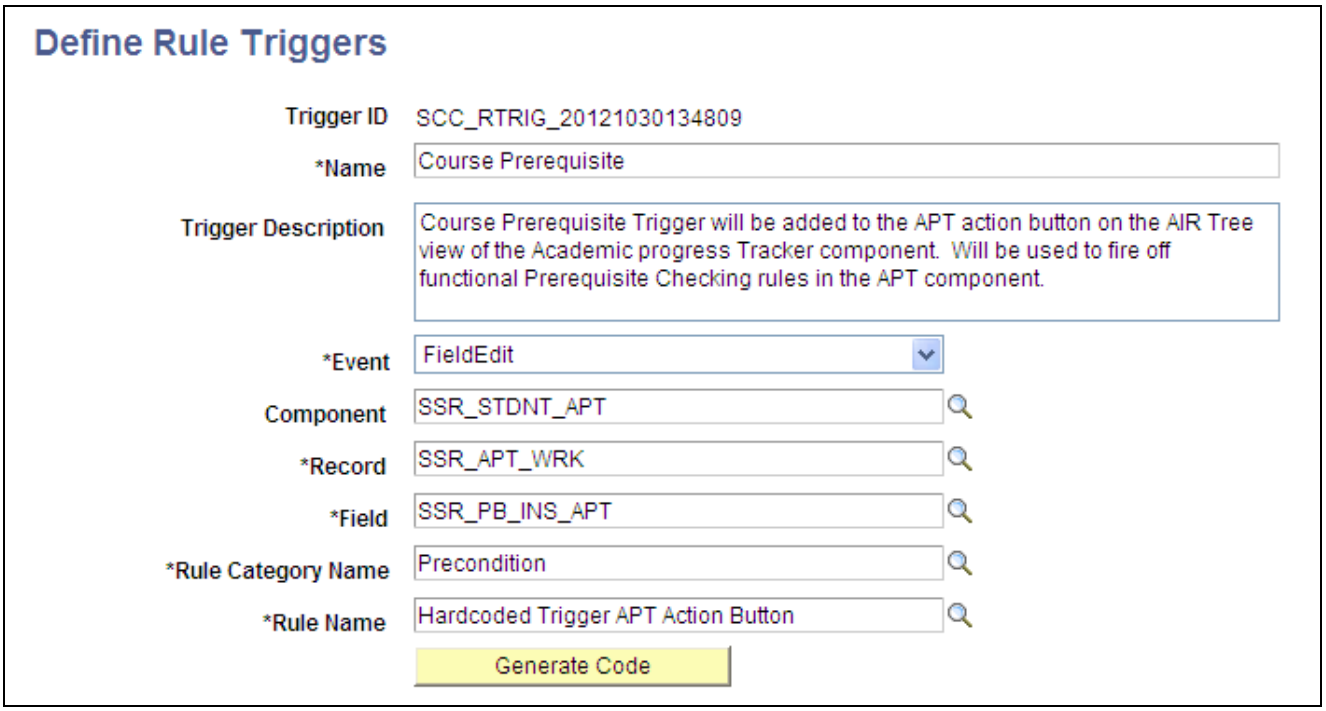
For more information, see the *Adding Variables to a Rule* section of this chapter.

Defining Rule Triggers

After a Trigger Rule is defined, a Trigger Definition can be created. Trigger Definitions allow you to associate the Trigger Rule to a Component Event or an Entity Method and generate PeopleCode that can be added to the associated component, record, or field event specified on the Define Rule Triggers page using the Generate Code button.

Note. The component helps to generate Template PeopleCode but does not automatically add this code to the specified event. You may want or need to adjust generated code to add institution specific business logic.

Access the Define Rule Triggers page (Set Up SACR, System Administration, Rules Engine, Setup, Define Rule Triggers).



Define Rule Triggers page

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Trigger ID

Displays a unique ID generated by the system. When adding a new value the default value is *NOID*. The unique ID is generated when the Rule Trigger is saved. The Trigger ID is created by combining prefix SCC_RTRIG_ with the system date and time stamp in format YYYYMMDDHHMMSS.

Name

Enter a Trigger Name. The Trigger Name is used when searching for a Trigger and for display.

Trigger Description

Enter descriptive text explaining the function of the Trigger..

Event

Note. This field is only available if the Trigger Type is Online Application.

Choose the PeopleCode Component Event to add to the Trigger:

- FieldEdit
- FieldChange
- SavePostChange
- SaveFieldChange

:

Component

Note. This field is only available if the Trigger Type is Online Application.

Select from a list of available Components in the environment by Component Name or Description.

Record

Note. This field is only available if the Trigger Type is Online Application.

Select from a list of available records for the selected Component *or* any record in the system if no Component is selected by Record Name or Description.

Field

Note. This field is only available if the Trigger Type is Online Application.

Select from a list of available fields for the selected Record selected by Field Name or Description.

Rule Category Name

Select from a list of Rule Categories for which the “Allow Trigger” option has been selected on the Define Rule Categories definition page by Rule Category Name or Long Description.

Rule Name

Select a Rule to associate with the Trigger.

Using Statements for Evaluation and Calculation in a Rule

The section discusses how to:

- Activate and move Statements within a Rule.
- Apply Rule concepts and add Statements to a Rule.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Pages for Using Statements for Evaluation and Calculation in a Rule

Page Name	Definition Name	Navigation	Usage
Select a Statement	SCC_RULE_STMT_SEC	On the Define Rule page, click the Details icon on a blank row in the Evaluations and Calculations group box.	Select a Statement to add to a Rule.
Define If Statement	SCC_RULE_IF_SEC	On the Select a Statement page, select IF.	Select an If Statement for a Rule.
Define Else Statement	SCC_RULE_ELSE_SEC	On the Select a Statement page, select ELSE.	Select an Else Statement for a Rule.
Define For-Each Statement	SCC_RULE_FOR_SEC	On the Select a Statement page, select FOR EACH.	Select a For-Each Statement for generic loop processing for a Rule.
Exit For-Each Statement	SCC_RULE_EXTF_SEC	On the Select a Statement page, select EXIT FOR EACH.	Select to exit a For-Each loop.
Define Assignment Statement	SCC_RULE_ASGN_SEC	On the Select a Statement page, select ASSIGN.	Select an Assignment Statement to assign data to an object.
Define Call Statement	SCC_RULE_CALL_SEC	On the Select a Statement page, select CALL.	Select a Call Statement to call a Rule.
Exit Rule Statement	SCC_RULE_EXTR_SEC	On the Select a Statement page, select EXIT RULE.	Insert to exit a Rule.
Create Entity	SCC_RULE_CREN_SEC	On the Select a Statement page, select CREATE ENTITY.	Create an Entity within a Rule.
Call Dynamic Rule Group	SCC_RULE_DYNR_SEC	On the Select a Statement page, select CALL DYNAMIC RULE GROUP.	Insert to call a dynamic Rule Group.
Define Add To List Statement	SCC_RULE_LADD_SEC	On the Select a Statement page, select ADD TO LIST.	Select an Add to List Statement to add a value to a list in a Rule.
Define Length of List Statement	SCC_RULE_LENL_SEC	On the Select a Statement page, select LENGTH OF LIST.	Select a Length of List Statement to return the length of a list in a Rule.
Define Sort List Statement	SCC_RULE_SRTL_SEC	On the Select a Statement page, select SORT LIST.	Define an Sort List Statement to sort a list for a Rule.
Define Clear List Statement	SCC_RULE_LCLR_SEC	On the Select a Statement page, select CLEAR LIST.	Select a Clear List Statement to reset a List variable to empty in a Rule.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Understanding Statements for Evaluation and Calculation in a Rule

You can use delivered Rules Engine Statements in the Rules Engine Manager Evaluations and Calculations grid to set up evaluative logic for your business Rules or Functions. Available Statements can be used to perform a specific task in the Rule you have created.

Access the Select a Statement page (click the Details icon on the left side of an empty row in the Evaluations and Calculations Grid).

Select a Statement

	Name	Description
☐	ADD TO LIST	Add a value to a list.
☐	ASSIGN	Assign data to an object
☐	CALL	Call Rule
☐	CALL DYNAMIC RULE GROUP	Call Dynamic Rule Group
☐	CLEAR LIST	Resets a list variable to an empty state.
☐	CREATE ENTITY	Create an entity and make the new entity the current context
☐	EXIT RULE	Exit a rule at this statement
☐	FOR EACH	For generic loop processing.
☐	IF	If
☐	LENGTH OF LIST	Returns the length of a list
☐	SORT LIST	Sort a list

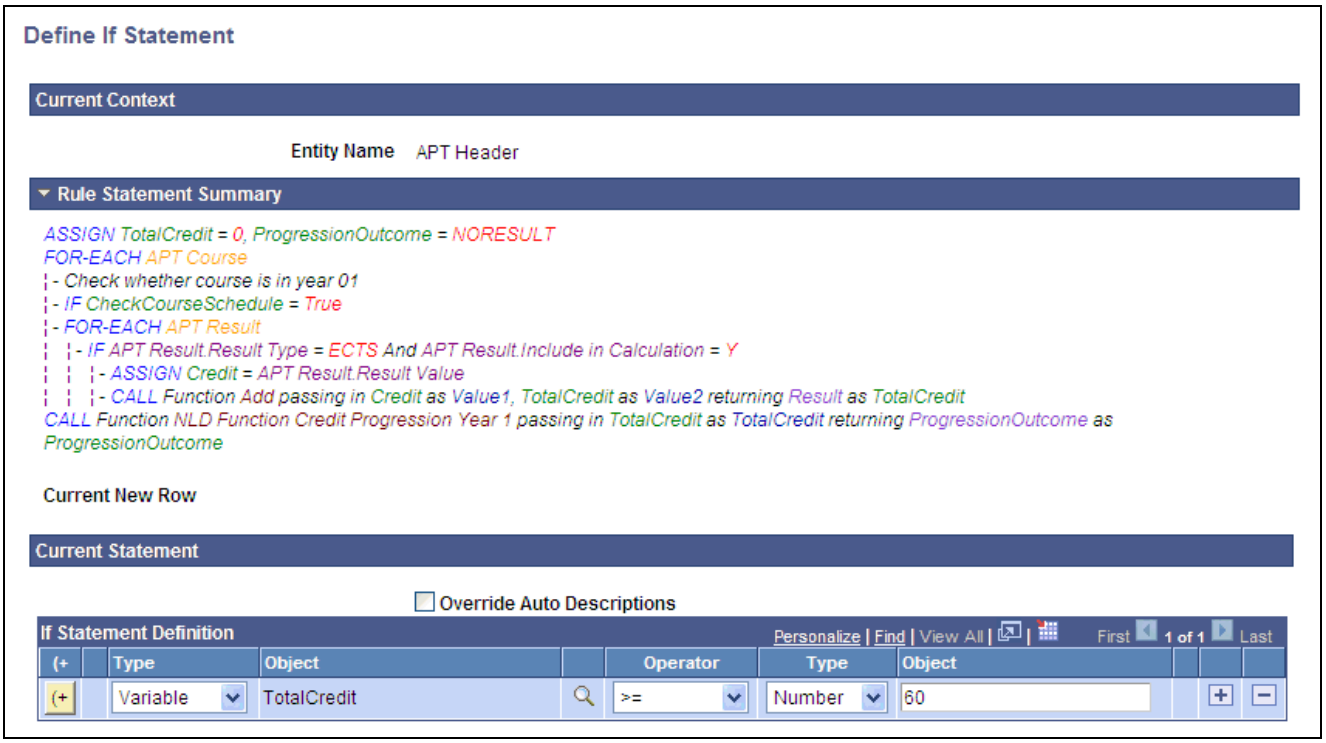
Return

Delivered Statements

Understanding Common Statement Attributes

When you select a Statement, a Define Statement page opens. Each Statement has a Define Statement page created specifically for that Statement. Although each Define Statement page is different, there are common fields and elements shared by all Statements. Each Define Statement page for each delivered Statement is explained below, but, first, to illustrate the common attributes of the Statement, access the Define If Statement page (select the check box next to the *IF* Statement Name on the Select a Statement page).

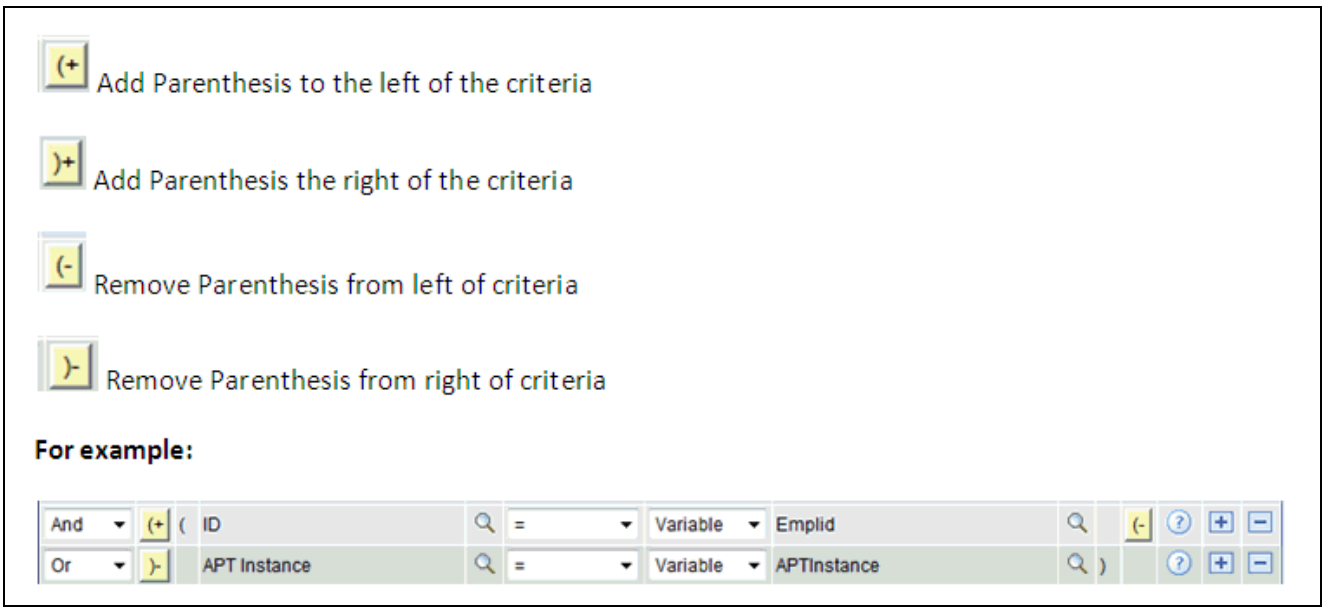
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Example of Common Statement Attributes

Connectors and Parentheses

Use the AND/OR connectors and parenthesis to create complex (nested) Statement definitions.

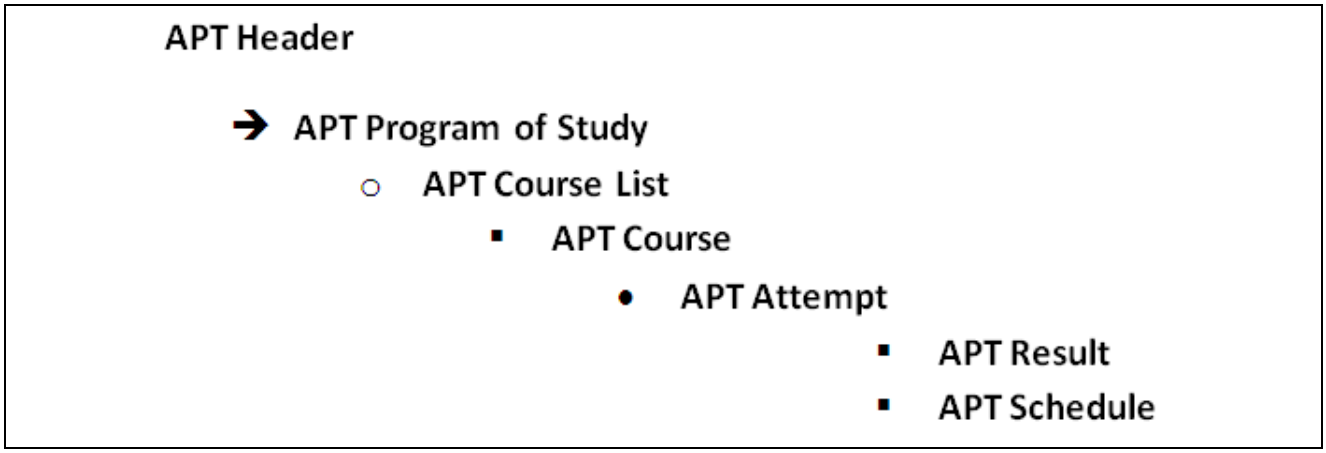


Example of Connectors and Parentheses

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Current Context

Displays the Current Context of the Statement. The Current Context provides information about the current Entity you are working with when you have a Rule that accesses multiple layers in a Data Hierarchy represented by the Base Entity. You always have access to properties from the Current Context and the immediate Parent. For example, when working with the Academic Progress Tracker (APT) using System Data only, your Current Context could be one of the following:



Example of Academic Progress Tracker Entity Hierarchy

In our example, the context is set to APT Header which means that we are able to use as data, within our IF Statement, all properties from the APT Header. If our current context had been APT Program of Study we would have been able to use all properties from the APT Program of study as well as the APT Header.

Rule Statement Summary

Displays a summary of the entire evaluative statement. The view is shown in color. Bold type indicates which Statement you are currently working with.

If you have not yet added a Statement, this is shown as "Current New Row".

Current Statement

Displays the short description of theCurrent Statement you are working with as well as the options for this Statement.

Override Auto Description

Select the Override Auto Description check box to override the Generated Description and provide your own using a Rich Text Editor.

Understanding Statement–Specific Attributes

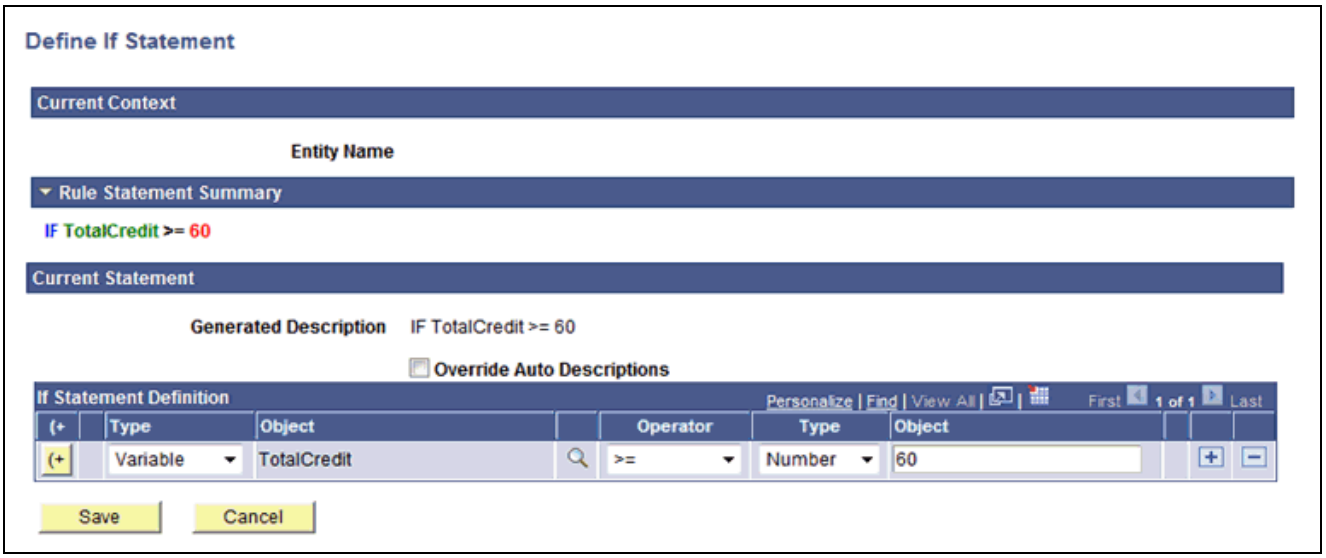
Each delivered Statement contains Statement-specific features; as described below with examples.

IF Statements

Use this statement to create evaluative Statements in your business rule. This is an encompassing statement; meaning that within the context of this statement, you can use other statements.

This example evaluates whether or not a student has achieved a sum of credits higher than 60 in Year 1 of his program. If the condition is met, the student may progress to Year 2.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



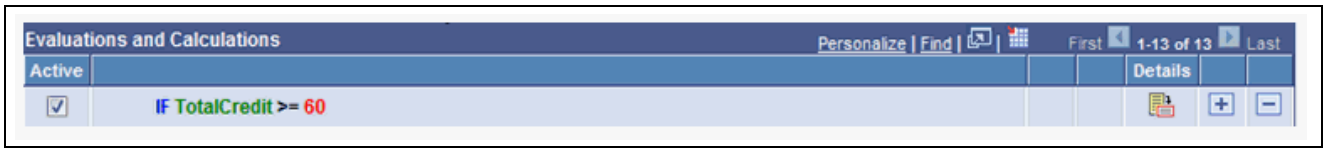
Example of IF Statement before Saving

The possible combinations of Objects and Types in IF Statements are:

Type	Object	Type	Object
Variable	Variable of Type *Number*	*Number*	Open field
		Property	Prompt for Search of Properties of Type Number
		Variable	Prompt for Search on Variables of Type Number
	Variable of Type *Text* **Note.** This works the same for Variables of Type *Date*, *Datetime*, *Time*, and *Entity*	*Property*	Prompt for Search on Properties of Type Text
		Text	Open field.
		Variable	Prompt for Search on Variables of Type Text
Property	Prompt for Search of Entity Properties of Type *Number* **Note.** This works the same for Variables of Type *Date*, *Datetime*, *Time*, and *Entity*	*Number*	Open field
		Property	Prompt for Search of Properties of Type Number
		Variable	Prompt for Search on Variables of Type Number
Property or Variable	Property with LOV prompting enabled or Variable with LOV prompting enabled	*Text*	Object field is disabled, but the prompt button is enabled which allows you to prompt on the relevant LOV value.
DS Prop (data set property) **Note.** Only available if Data Sets are in the Rule	Prompt for Data Set Property	Type associated with the Data Set Property	Open field or Prompts depending upon Type

The saved IF Statement looks like this:

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Example of IF Statement After Saving

ELSE Statements

Combine ELSE statements with IF statements to create complex evaluative business Logic. This statement can only exist in the context of an IF statement. The ELSE statement page itself offers no features besides the possibility to override the auto-generated text.

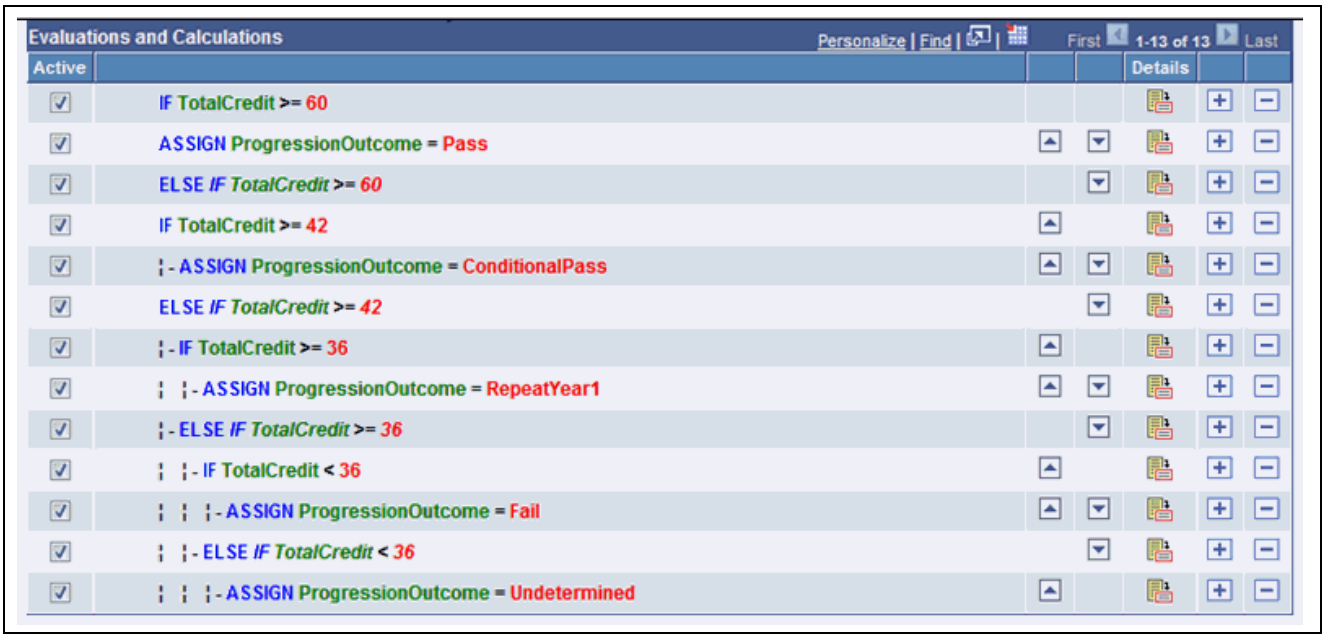
Note. If you do not already have an IF statement in your Rule, the ELSE statement is not shown as an option on the Select a Statement page.

This example evaluates whether or not a student has achieved a sum of credits higher than 60 in Year 1 of his program. If the condition is met, the student may progress to Year 2. However, if the student has less than 60 credits but greater than 40 credits, this student may need to repeat courses from Year 1.

Define ELSE Statement page

Here is an example of how an Evaluative Statement can be created combining IF and ELSE statements:

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

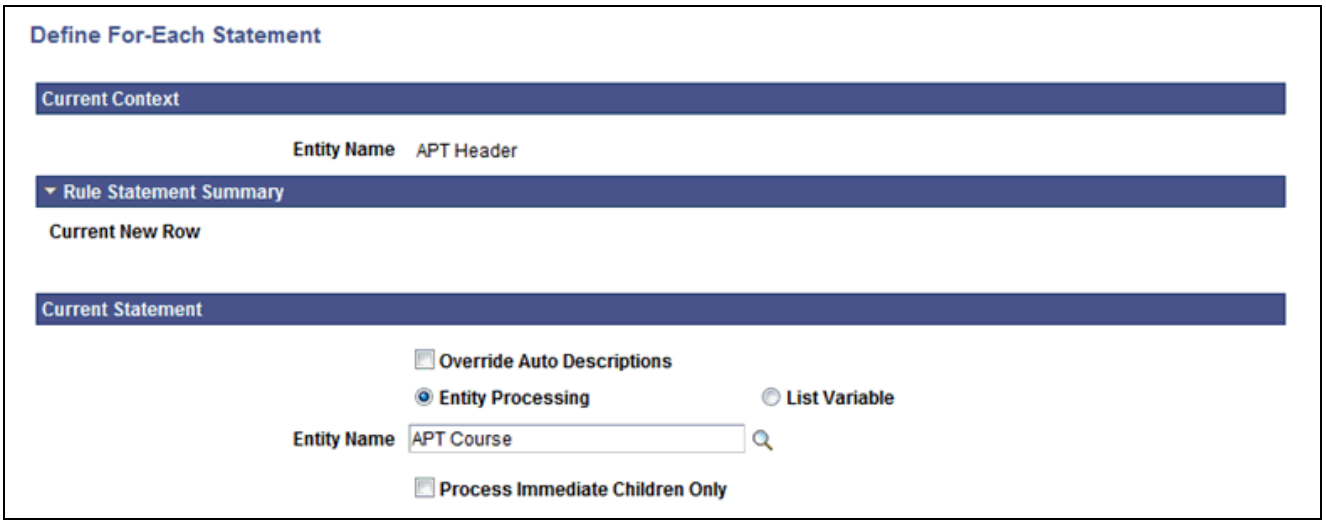


Example of IF and ELSE Statements Used Together

FOR EACH Statements

Use this statement to scroll through a set of elements which has been retrieved into the Rule. These elements can be Entities retrieved in the context of a Rule or Variables from a list of Variables. The FOR-EACH statement is an encompassing statement; meaning you can use other statements within the context of the FOR-EACH.

For example, to scroll through the Academic Progress Tracker entity to retrieve all courses in a program, select a FOR-EACH statement:



Define FOR-EACH Statement page

Statement-specific fields on this page are:

Entity Processing	Select this option to activate the Entity Name prompt and Process Immediate Children Only check box.
List Variable (list variable radio button)	Select this option to activate the List Variable prompt.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Entity Name	Select an Entity.
List Variable(list variable field)	Select a List Variable.
Process Immediate Children Only	Select this check box to process only immediate child records of the selected Entity. This means only a direct child record of the Entity is used in the process; *not* grand children of great-grand children in the same Entity structure.

For example, you want to evaluate whether a student has achieved a sum of credits higher than 60 in Year 1 of his program of study in order to progress to Year 2. To calculate the overall credit total, you must first retrieve all courses for Year 1 and sum the credits of each course. First, you retrieve Year 1 for the student, and then you use a FOR EACH (course) statement to loop through all courses and retrieve the credits.

Since the Courses in Year 1 may be part of a Course List or a Requirement and not an immediate child of the Year, you do not want to select the "Process Immediate Children Only" option. However, if you wanted to make sure that you only process immediate children of the Year, you do want to select this option.

EXIT FOR EACH Statement

The EXIT FOR EACH statement is only available in the context of a FOR EACH statement. Use this statement to exit scrolling through a set of elements which has been retrieved into the Rule. These elements can be Entities retrieved in the context of a Rule or Variables from a list of Variables.

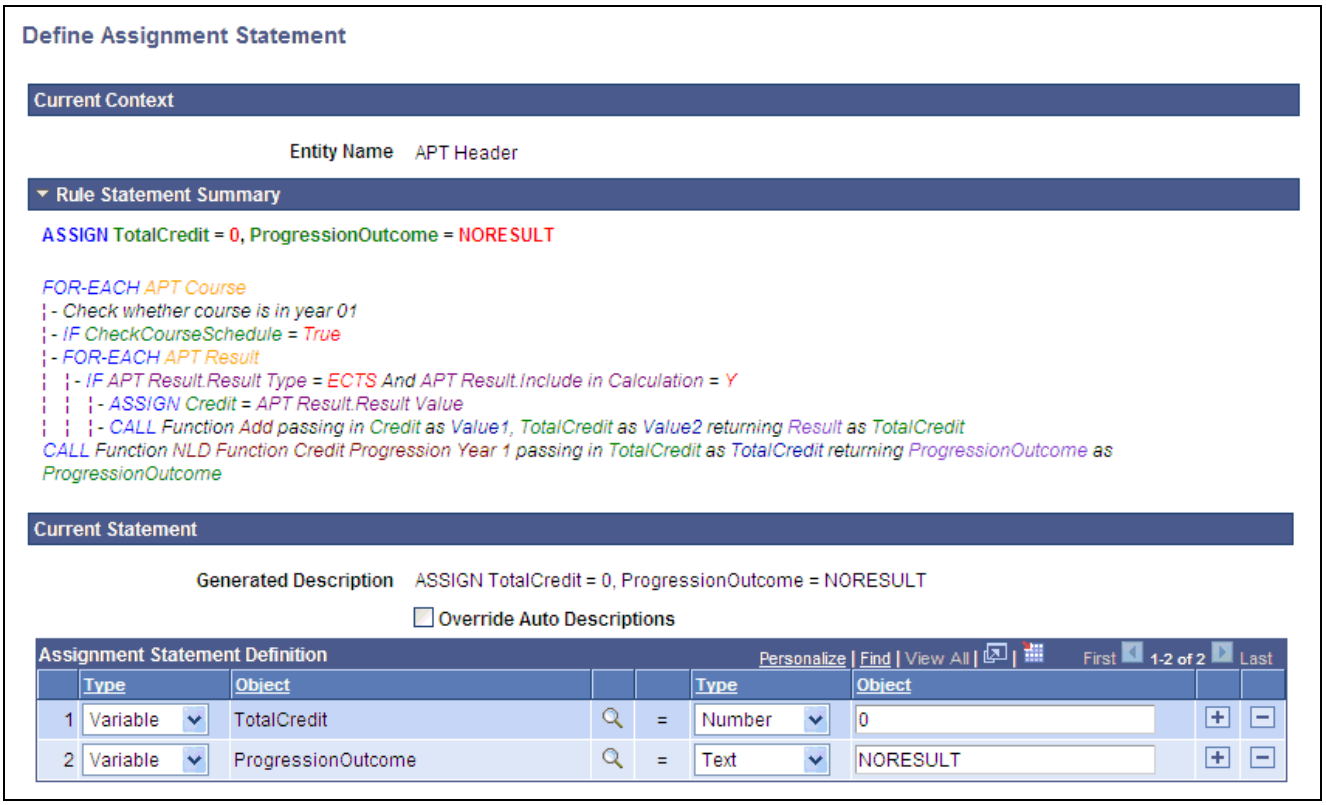
For example, you want to evaluate whether a student completed Course Math 101 in Year 1 of his program of study. If the condition is met, the student may progress to Year 2. When the condition has been met (Math 101 course has been found), there is no need to retrieve other Courses from the system. The EXIT FOR EACH can be used to exit the for each loop at this time.

ASSIGN Statements

Use this statement to assign values to Variables or Properties in your Rule

For example, to assign Credit = 2 to Variable Credit, select the *ASSIGN* statement from the Select a Statement page:

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Define Assignment Statement page

The possible combinations of Objects and Types in ASSIGN Statements are the same as for IF Statements. Please refer to the table in the *If Statements* subsection in the *Understanding Statement–Specific Attributes* section of the chapter:

CALL Statements

Use this statement to call a Function or Rule from within the current Function or Rule. The current Rule is the Calling Rule. The Function or Rule called executes a piece of business logic in the context of the Calling Rule.

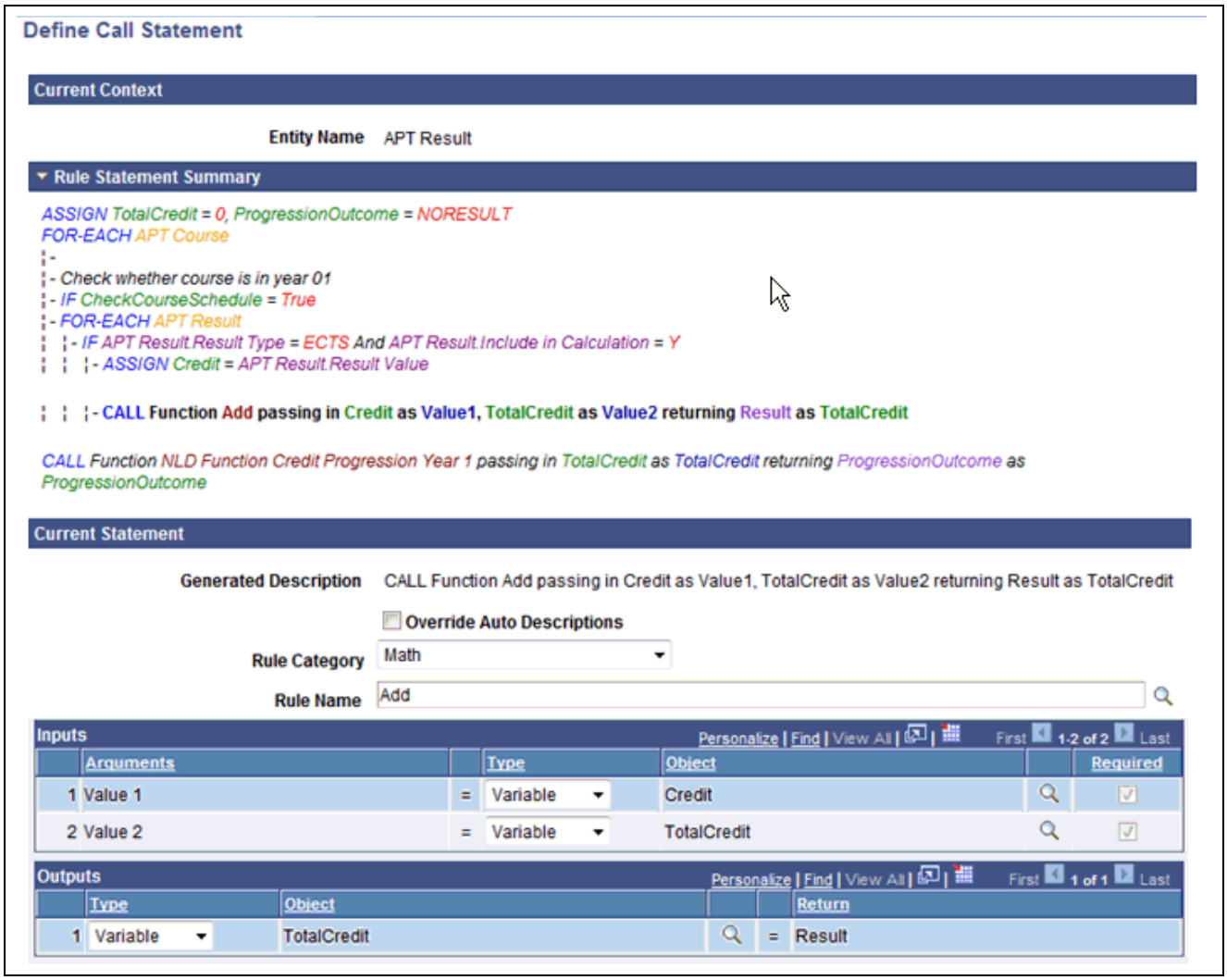
When you call a *Rule*, you need to pass all Criteria for that Rule as input to that Rule otherwise the correct data cannot be selected.

When you call a *Function* which uses Entity Data, you do *not* need to provide Criteria to find the correct data. That data is passed to the called Function automatically through Contextual Referencing.

For more information, see the *Understanding Contextual Referencing* section of this chapter.

In this example, a student is evaluated to see if he has achieved a sum of credits greater than 60 in Year 1 of his program of study 1. If that condition is met, the student may progress to Year 2. In order to obtain the student's credit total, you must first to retrieve all courses for Year 1 and sum the credit result by retrieving each course to obtain the overall sum of credit. This is done using a FOR EACH statement that retrieves Year 1 and loops through all Courses and retrieves credits. This uses an Add math Function to add Course Credits to a variable, SumCredit. At the end of the FOR EACH loop, the total number of credits for all courses in Year 1 is counted. Here is a similar example of the Add Function Add being called from a Rule:

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Define Call Statement page

These are the Call Statement-Specific fields:

Rule Category Select a valid Rule Category from which to retrieve the Function you want to use. The Rule Category available must be one of the valid categories on the Rule Category Definition page.

Rule Name Click the Search icon to the right of the Rule Name to open the Rules Engine Search page and select a Rule.

Inputs Select an Argument and Type for each Object. This is required so that all Input can be passed from the Calling Rule to the Called Function. The arguments are the Input Variables that have been defined in the Called Function. The Operator field is restricted to only show the *Equals to* operator. The Type is restricted to the Type that is assigned to the Argument.

Outputs Select a Type and Object for each Output. The Return represents the Variable(s) that has been defined as Output Variable in the called Function. The Type is restricted to the Type that is assigned to the Argument.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

EXIT RULE Statement

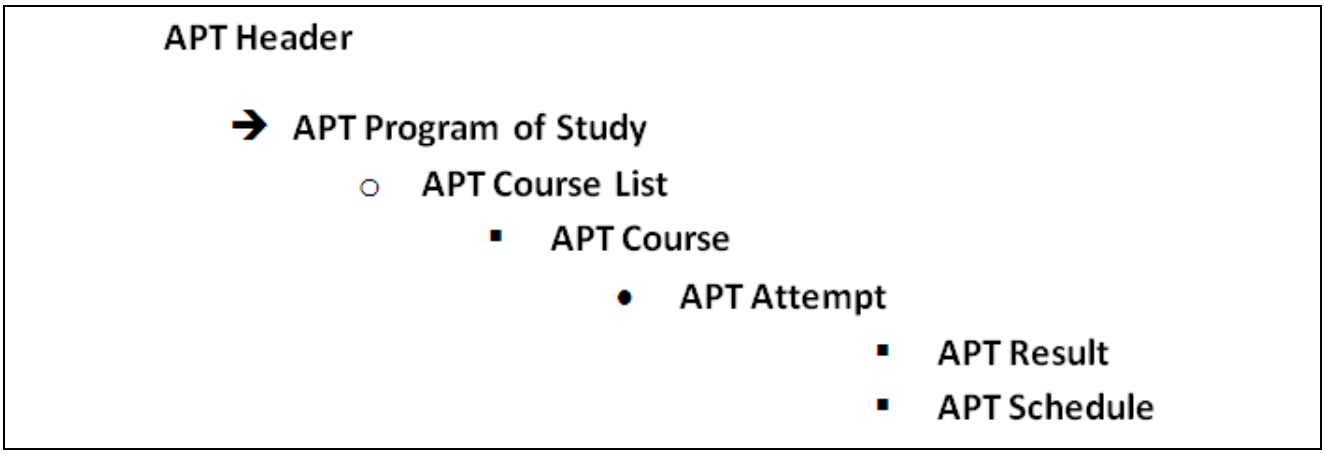
Use this statement to exit a Rule or Function in its entirety. The outcome for this Rule would be to process to success.

For example, to find out whether a student completed Course Math 101 in Year 1 of his program of study, you can create a Rule to retrieve only that information. If you have use a FOR EACH statement in the Called Rule to scroll through all of the courses in Year 1, you can use an EXIT RULE statement to exit the FOR EACH loop and the Rule entirely once the condition has been met.

CREATE ENTITY Statement

Use this statement to create an Entity. After the Entity is created, the user is transferred to the context of the created Entity and all Entity properties are available.

You can only create an Entity in the context of a parent Entity. For example, when in the context of APT Course, you can create Entity APT Attempt, and when working in the context of APT Attempt, you can create and APT Result Entity or APT Schedule Entity.



Example of APT Header Hierarchy

When the Entity is created, it does not yet exist in the database. The CREATE ENTITY statement can be used together with the delivered Function Save Entity to insert new rows into database tables.

Note. You can only create an Entity as a child within the Entity of the Current Context.

The following Functions can be used after having changed an entity or after having created an entity:

Function	Function Description
SaveEntity	Saves the Current Entity in context and all of its children. Performs all validation and pre-save logic and deletes any entities marked for deletion.
SaveAllEntities	Saves all the Entities in context and all of their children. Performs all validation and pre-save logic and deletes any entities marked for deletion.
DeleteEntity	Marks the Current Entity and all of its children for deletion. SaveEntity Function must be called to actually delete the Entity.
UndeleteEntity	Marks the Current Entity and all of its children to be un-deleted. SaveEntity Function must be called to actually delete the Entity

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

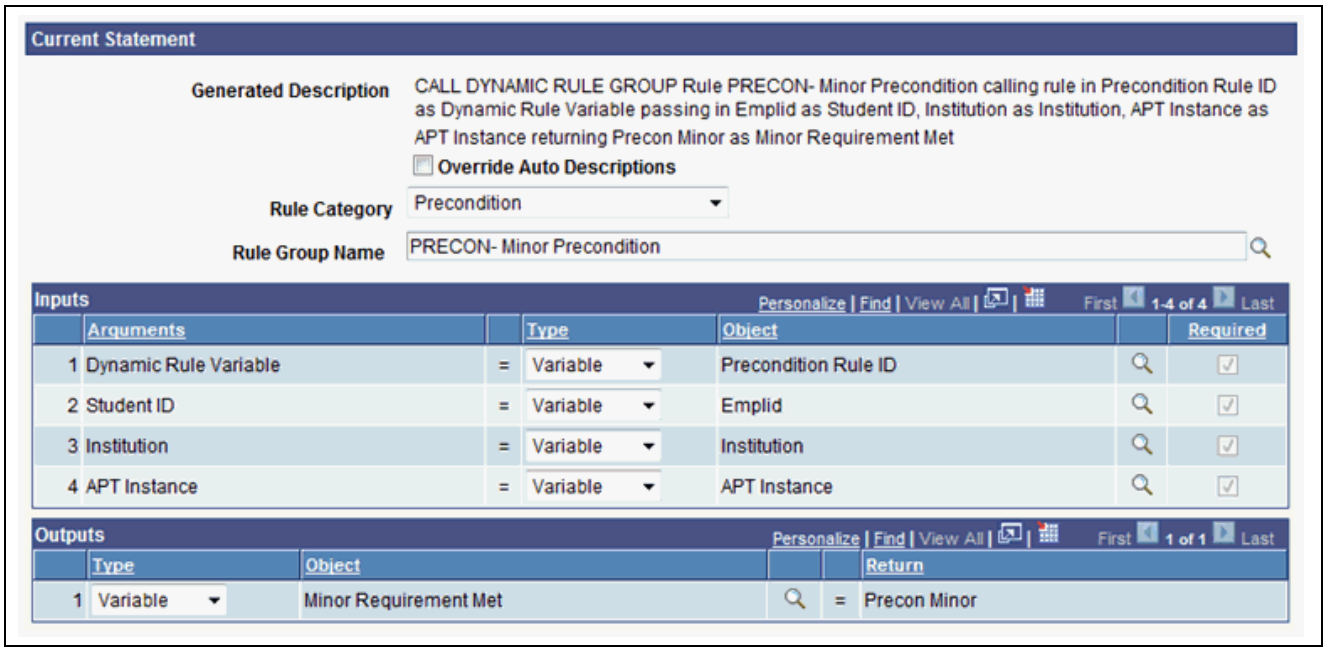
Warning! Using the CREATE ENTITY statement in combination with delivered Functions which can save an Entity, like the Save Entity Function, inserts data into referenced database tables. These Statements and Functions should only be used by experienced users with a Skill Level of *Expert* who have a good understanding of the Campus Solutions record structure. Rules Engine administrators should minimize risk by allowing Functional Experts to create Entities but not authorizing them to use one of the Save Entity Functions *or* by creating separate Functions in which CREATE ENTITY and SAVE ENTITY statements are combined.

For example, continuing the example of evaluating whether a student has achieved a sum of credit higher than 60 in his program of study year 1, you can use the CREATE ENTITY statement to create and Entity to store the result of the that calculation in APT result. Once the Entity is created, you populate the appropriate fields with values. In this case, Result value with the Sum of Credits retrieved by the Rule. To do this, use the Assign statement to assign the actual Sum of Credits to the Result value property for Result Type *Sum Credit*. Once assigned, you can save the Entity, which creates a row in the database using the system delivered Function Save Entity.

CALL DYNAMIC RULE GROUP Statements

Use this statement to call a Rule from a dynamic Rule Group. A Rule Group provides Rules with a standard template with predefined input and output parameters as well as a Base Entity. Rules which share the same template characteristics can be grouped in the same Rule Group. When calling Dynamic Rule Groups, all Rules belonging to this Rule Group are called and executed. The following example passes a retrieved Rule ID to the Rule Group PRECON-Minor Precondition. The Rule associated with the ID is then called.

For more information, see the *Understanding Rule Groups* and *Define Rule Groups* sections of this chapter.



CALL DYNAMIC RULE GROUP Statement page

These are the Call Statement-Specific fields:

Rule Category Select a valid Rule Category from which to retrieve the Rule Group you want to use. The Rule Category available must be one of the valid categories on the Rule Group Category Definition page.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Rule Group Name Click the Search icon to the right of the Rule Name to open the Rule Group Search page and select a Rule Group.

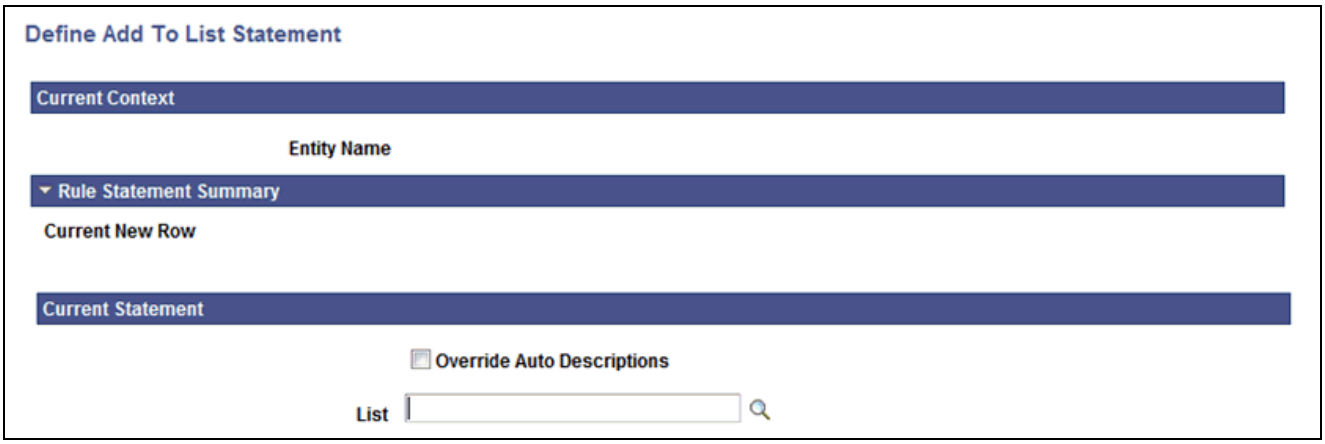
Inputs Select an Argument and Type for each Object. This is required so that all Input can be passed from the Calling Rule to the Called Function. The arguments are the Input Variables that have been defined in the Called Function. The Dynamic Rule Variable will always be the first required Input parameter when calling Dynamic Rule Groups. The Operator field is restricted to only show the *Equals to* operator. The Type is restricted to the Type that is assigned to the Argument.

Outputs Select a Type and Object for each Output. The Return represents the Variable(s) that has been defined as Output Variable in the called Function. The Type is restricted to the Type that is assigned to the Argument.

ADD TO LIST Statements

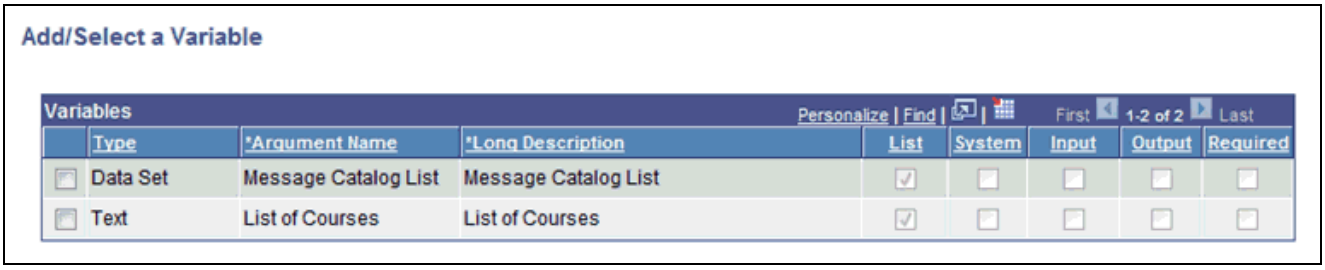
Use this statement add values to a List.

The List to which you want add Variables must have been created before using this statement. ADD TO LIST statements can also be used in the context of a FOR EACH loop; however, when using ADD TO LIST, you cannot add to the same List that you are scrolling through within the context of a FOR EACH loop.



Define ADD TO LIST Statement page

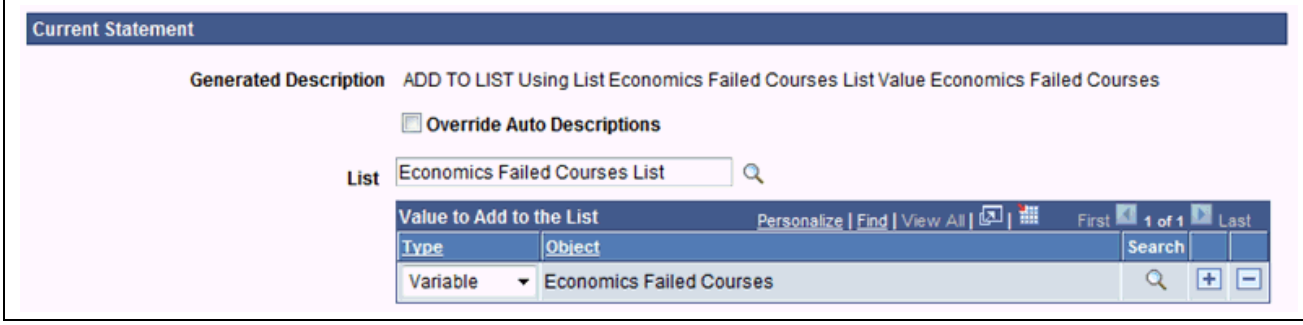
Click the Search icon to the right of the List field to open the Add/Select a Variable page:



Example of Add/Select a Variable page with only List Variables

After having selected the List, you are returned to the ADD TO LIST statement page to add relevant Variables to the list:

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



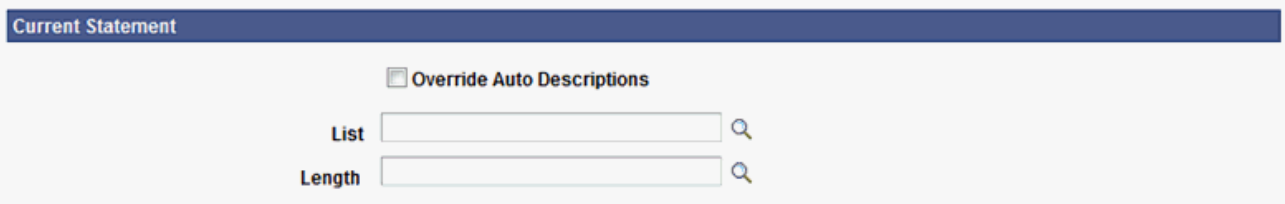
Example of ADD TO LIST statement page with Value to Add to the List Grid

- If the selected List is a Data Set, you may only add Type Variable to the List, and the Variable must be a Data Set.
- If the list is *not* a Data Set, you may add the following to the List:
  - A Variable which matches the Type of the List (for example, *Text*).
  - A Data Set Property which matches the Type of the List.
  - A Property which matches the Type of the List.
  - A user–defined value of Type *Text* .

For example, a curriculum requirement dictates that students may only fail three courses in subject area “Economics”. A Rule is needed to capture a List which contains both the Course as well as the failed Mark. For this, use a Data Set for the purpose of capturing the failed courses in subject area Economics. At the time of creation, the List is empty. Associate the List Variable with the created Data set. In the same Rule, also add a simple List containing all Courses. In a FOR EACH loop which scrolls through all the Courses in a program of study, evaluate the subject area for each course, determine whether the course is failed and, if meeting the criteria, add the Course ID as well as the Mark to the Data Set list. Use a ADD TO LIST to add Course ID’s and Marks as they are found.

LENGTH OF LIST Statement

Use this statement to determine the length of any List.



Define Length of List Statement page

Click the Search icon to the right of the List field to open the Add/Select a Variable page.

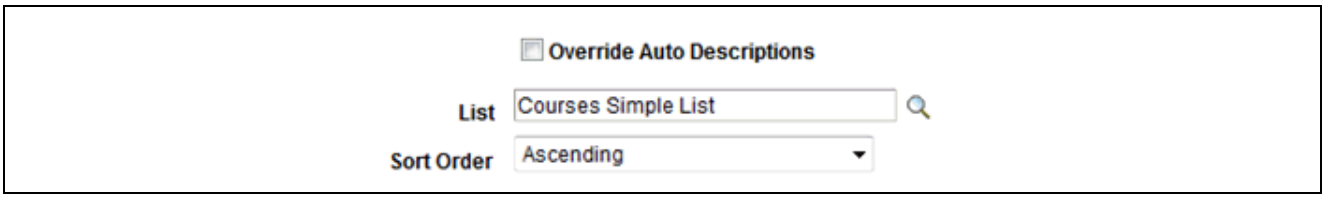
Enter a Number Variable in the Length filed or create a Numeric Variable on the fly to designate the Length of the List.

SORT LIST Statements

Use the SORT LIST statement to sort a List by any List value. Depending of the type of List you select, you are presented different options upon returning to the Define statement page.

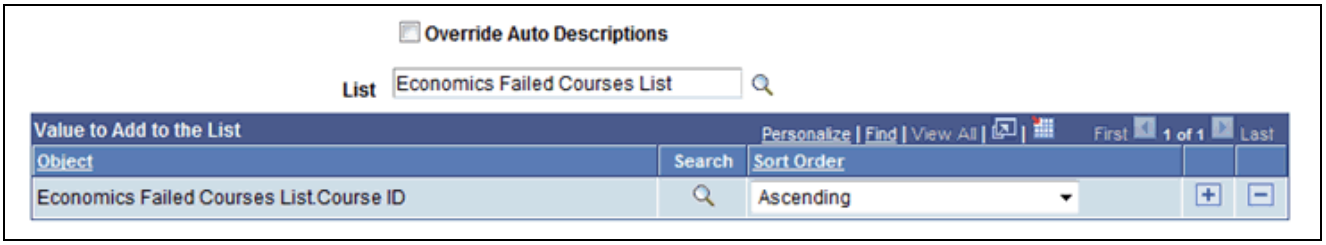
For Lists that *are not* Data Sets, select a Sort Order of *Ascending* or *Descending*.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Example of a Sort Order Specification that is not a Data Set

For Lists that *are* Data Sets, select the Search icon to the right of the Object to open theData Set Property Search page and select a Data Set. Upon return to the Define SORT LIST statement page, select a Sort Order of *Ascending* or *Descending* for selected Data Set.



Example of a Sort Order Specification that is not a Data Set

CLEAR LIST Statements

Use this statement to clear a list and remove all list data.

Click the Search icon to the right of the List field to open the Add/Select a Variable page and select a List.

Activating and Moving Statements within a Rule

Once Statements are added to a Rule, they can be activated or inactivated, and they can be moved within the Rule to alter the Rule logic.

Activating and Inactivating Statements

Select the Active check box to the right of each Statement to activate or inactivate the Statement.

An Inactive statement (check box is unchecked) is ignored by the Rules Engine compiler and skipped when executing a Rule build. Inactivated statement are displayed in non-bold italic text.

Moving Statements within a Rule

One way that Rule logic can be modified is by moving Statements within the Rule. There are two ways that a Statement can be moved within a Rule:

- Up and Down

Up and Down Toggle for Statements

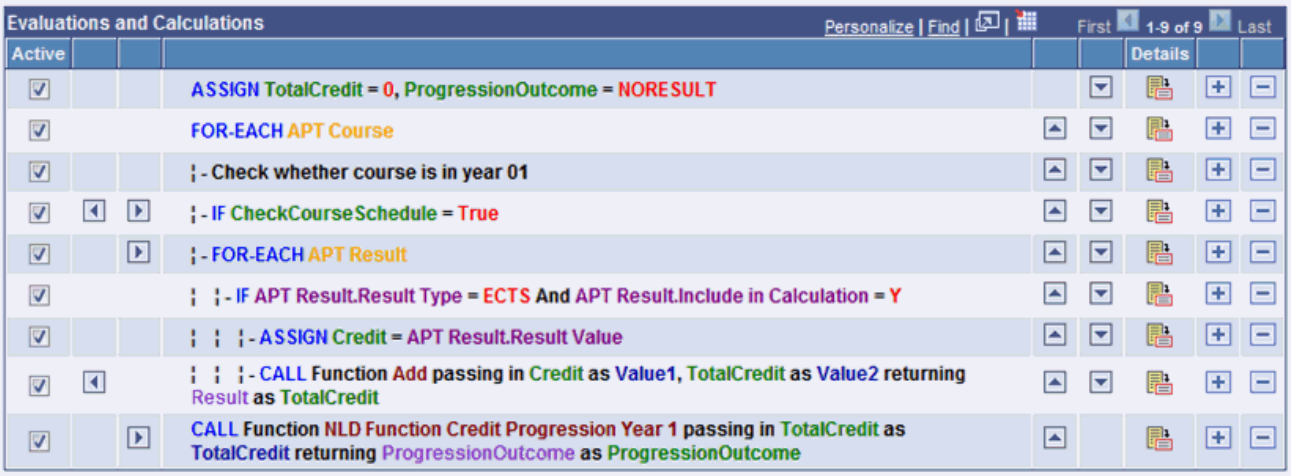
  Use the Up and Down Toggle fields to move a Statement before or after other Statements, changing the order in which Statements are executed. The type of Statement and where it is located in relation to other Statements determines if it can be moved up and/or down.

- By Indentation

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Indentation Toggle for Statements

Use the Indentation Toggle fields to move a Statement within or outside of other Statements, changing when a Statement is executed in relation to Statements above or below it.



Example Showing Statements with Indentations and Up and Down Toggles

For an example of incorrectly indented Statements, see the *Add a CALL Statement to Determine Academic Progression* subsection of the *Applying Rule Concepts and Adding Statements to a Rule* section of this chapter.

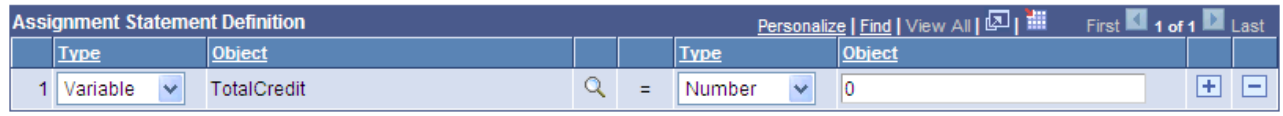
Applying Rule Concepts and Adding Statements to a Rule

In this section, an example Rule illustrates how Rule Options, Variables, Criteria, and Statements are used together. Previous sections of the chapter describe creating Variables, using Criteria to define a data set, and the features of delivered Statements for Rule evaluation and calculation.

The objective of this example Rule is to evaluate whether a student has obtained enough credits to be able to progress to the next phase of an academic program after having completed Year 1. To start, the Rule accesses the courses for the program of study for Year 1 and sums the credits for the courses the student has passed. In order to insure an accurate TotalCredit value, the Rule sets the TotalCredit Variable to zero prior to passing the sum result into a Variable, TotalCredit.

Add an ASSIGN Statement to Set the TotalCredit Field to Zero

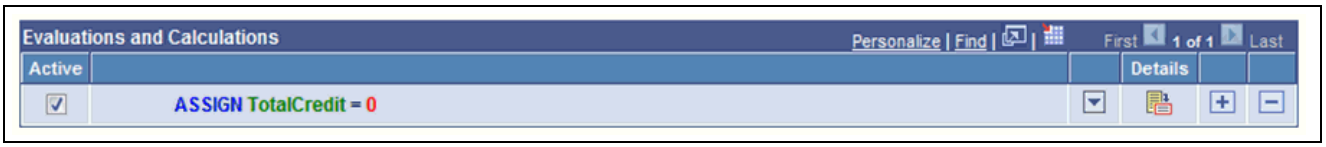
With Rule Options, Variables, and Criteria defined, begin by adding an ASSIGN statement that selects the Variable *TotalCredit* and set it equal to the Number *0* (zero):



Example of Assignment Statement Definition

Upon saving, the ASSIGN statement is added to the Evaluations and Calculations grid of the Define Rule page with the Rule text automatically generated using the text color setup:

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

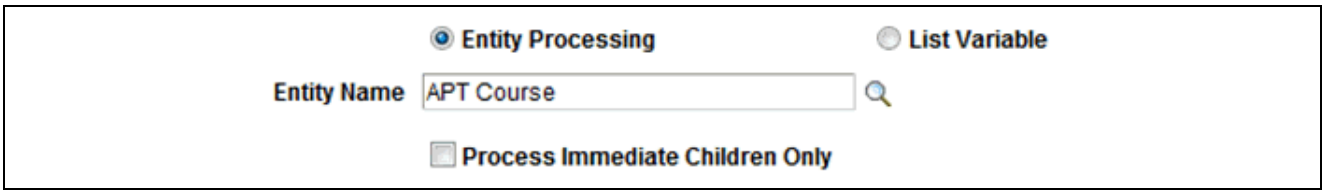


Example of ASSIGN Rule Text

For more information about ASSIGN statements, see Understanding Statement-Specific Attributes, ASSIGN Statements section in this chapter.

Add a FOR EACH Statement to Select All Courses

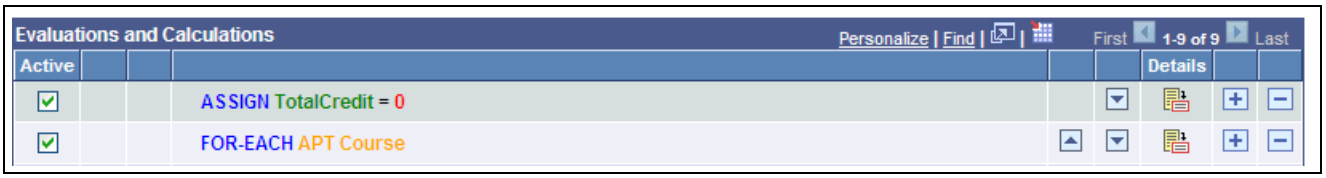
The next step is to retrieve all courses by adding a FOR EACH statement:



Example of FOR EACH Statement Definition

Select Entity Processing to activate the Entity Name field and prompt. Select the Search icon and choose the *APT Course* Entity. By selecting *APT Course*, you can retrieve all courses that are part of the APT instance selected as Base Entity. Do *not* select Process Immediate Children Only so the Rule selects Courses wherever they exist in the structure and not just the immediate children of the APT Header.

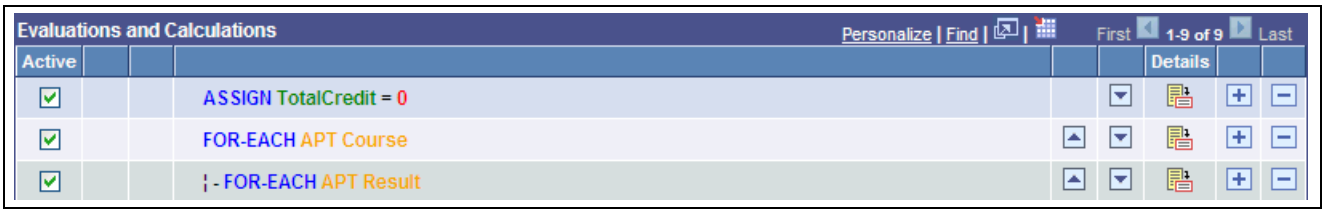
Upon saving, the FOR EACH statement is added to the Evaluations and Calculations grid of the Define Rule page with the Rule text automatically generated using the text color setup:



Example of FOR EACH Rule Text

Add a FOR EACH Statement to Select a Result for Each Course

Next, you add another FOR EACH Statement to Select APT Results for each Course. This statement is indented under the FOR EACH APT Course Statement so that it is executed for each to retrieve the APT Result for each APT Course.

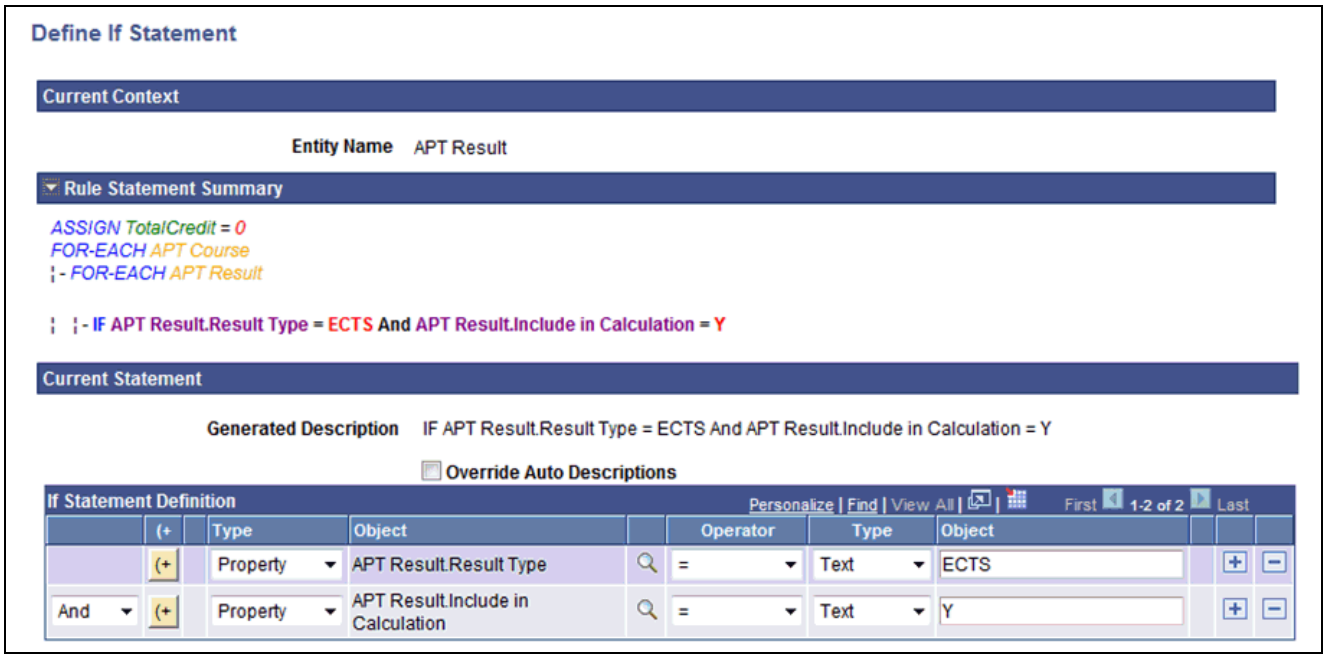


Example of Indented FOR EACH APT Result Rule Text

Add an IF Statement

Next add an IF statement to make sure that we select the correct result before we add it to our total credit count. Do this by restricting the APT Results by Result Type and whether the Result is included in the Calculation:

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



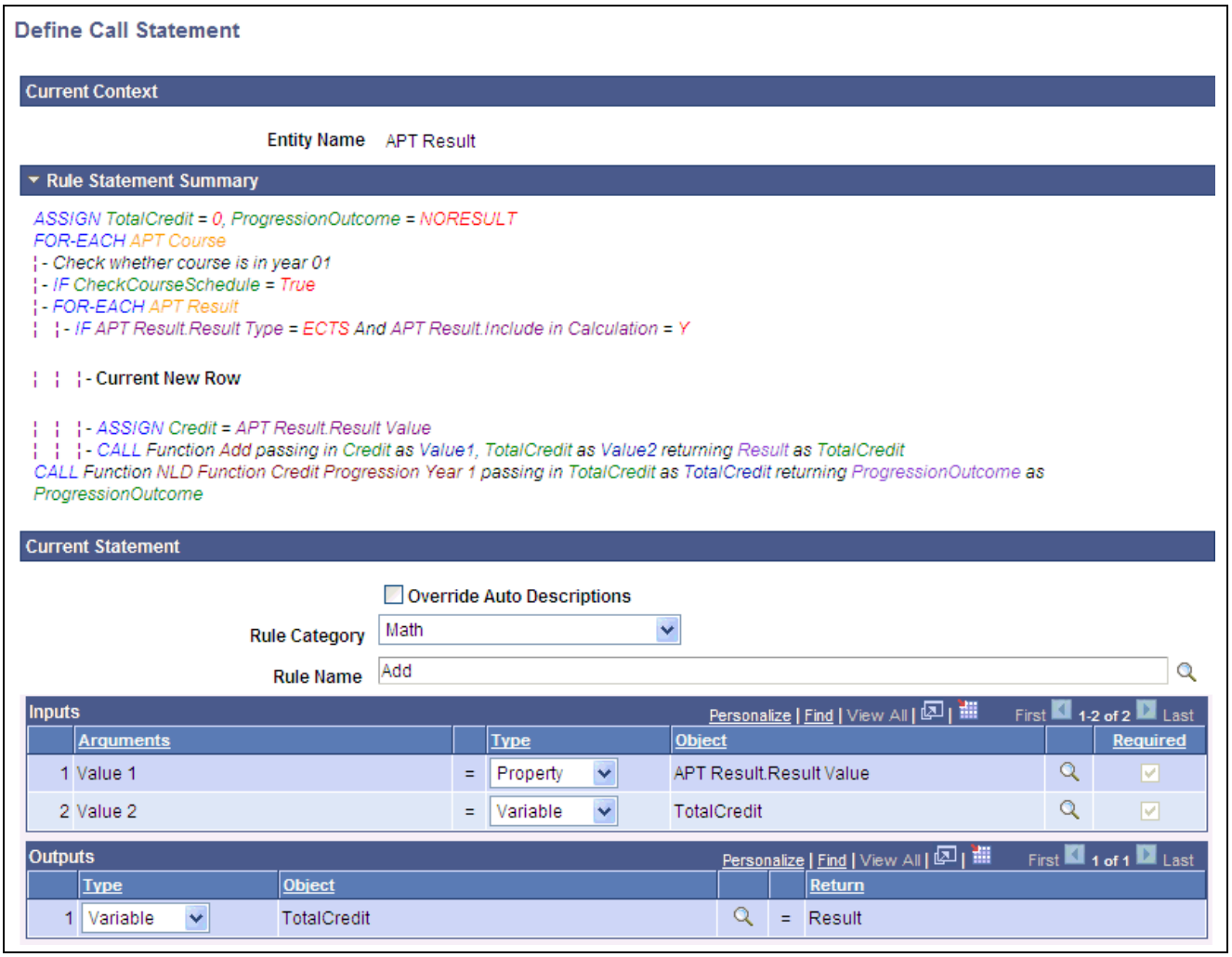
Example of Define IF Statement Definition

Add a CALL Statement to Sum Credits

After adding specified criteria for selecting results, use a CALL statement to call a Function to sum the retrieved credits. In the Define CALL Statement page, pre-delivered Functions are available that are restricted to those which are available in the Rule Categories, including common math functions.

When calling a Function, the required parameters are automatically displayed and the available parameter values are restricted to those that can retrieved for the Rule. For example, when selecting Rule Category *Math* and Rule Name *Add*, the Input and Output parameters are displayed. The selection of variables and properties and variables are restricted to show the correct property and variable types. In this case only numeric variables and properties can be selected:

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Example of Define CALL Statement

Upon saving, the CALL statement is added to the Evaluations and Calculations grid of the Define Rule page with the Rule text automatically generated using the text color setup:

Example of CALL Statement Rule Text

Add a CALL Statement to Determine Academic Progression

After calculating a student's total credits, you can use that result to evaluate whether the student may progress to Year 2 of his program of study. Assuming that multiple programs of study use the same credit requirement for academic progression, you can create a Rule of type Function that can be used by multiple academic progression Rules:

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Define Rule | Version History | Process Flow | Cross Reference

Rule ID	SCC_RULE_ID_20120801120231	Action	
Version	1	Rule Build Status	Rule Not Built
Rule Status	Active	Status Last Date/Time	03/01/2013 7:34AM PST
*Rule Name	NLD Function Credit Progression Year 1		
*Long Description	NLD Progression Rule For Year 1: >= 60 then Pass Between 42 and 60 Conditional Pass		
*Rule Category Name	NLD Demo Rules		
Rule Group Name			
*Rule Usage	Function		
Entity Name			
*Skill Level	Expert		
*Logging Level	Informational Messages		

☑ Available in Other Rules

Variables — Personalize | Find | View All | First 1-2 of 2 Last

Argument Name	Type	List	System	Input	Output	Required	Count	Details
TotalCredit	Number	☐	☐	☑	☐	☐	4	
ProgressionOutcome	Text	☐	☐	☐	☑		5	

Add a Variable

Example of Academic Progression Function (1 of 2)

Evaluations and Calculations — Personalize | Find | First 1-13 of 13 Last

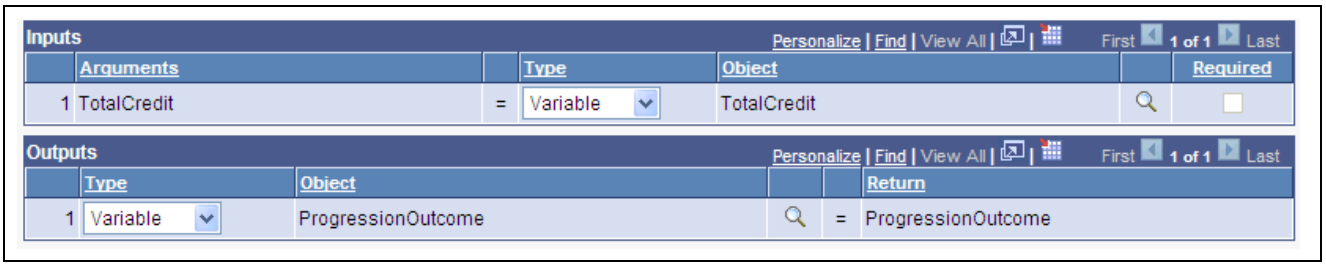
Active				Details		
☑	IF TotalCredit >= 60					
☑	¦ - ASSIGN ProgressionOutcome = Pass	▲	▼			
☑	ELSE *IF TotalCredit >= 60*		▼			
☑	¦ - IF TotalCredit >= 42	▲				
☑	¦ ¦ - ASSIGN ProgressionOutcome = ConditionalPass	▲	▼			
☑	¦ - ELSE *IF TotalCredit >= 42*		▼			
☑	¦ ¦ - IF TotalCredit >= 36	▲				
☑	¦ ¦ ¦ - ASSIGN ProgressionOutcome = RepeatYear1	▲	▼			
☑	¦ ¦ - ELSE *IF TotalCredit >= 36*		▼			
☑	¦ ¦ ¦ - IF TotalCredit < 36	▲				
☑	¦ ¦ ¦ ¦ - ASSIGN ProgressionOutcome = Fail	▲	▼			
☑	¦ ¦ ¦ - ELSE *IF TotalCredit < 36*		▼			
☑	¦ ¦ ¦ ¦ - ASSIGN ProgressionOutcome = Undetermined	▲				

Save Return

Example of Academic Progression Function (2 of 2)

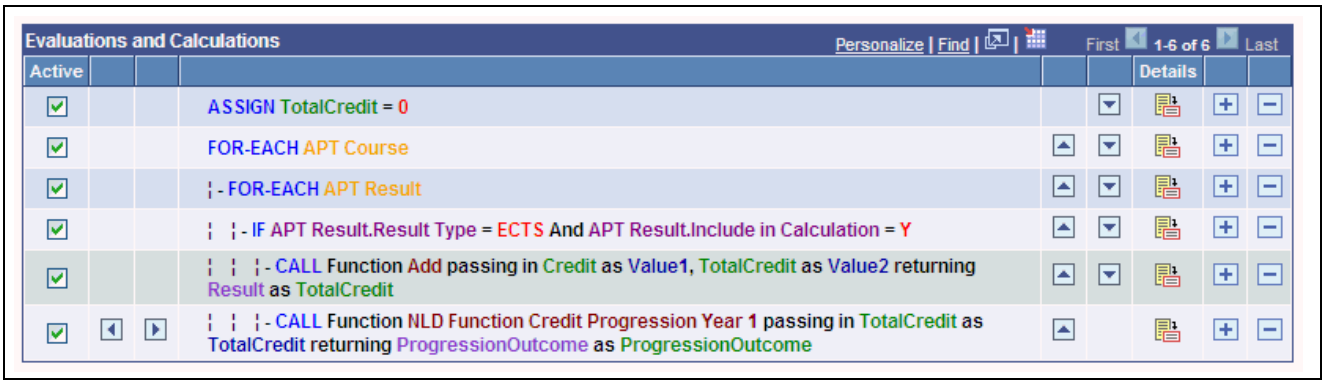
This Function can be called from the main Rule by adding a row to the Evaluations and Calculations grid and using the CALL statement. The Parameters from the Rule are automatically offered as Input and Output parameters on the Define Call Statement page:

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Example of Academic Progression Input and Output

Upon saving, the academic progress CALL Function is added to the Evaluations and Calculations grid of the Define Rule page with the Rule text automatically generated using the text color setup. However, its placement would result in its being executed at the wrong time:



Example of Academic Progression Rule Text with Progression Function Incorrectly Indented

With the CALL Function in the current position the evaluation will take place for each result selected as follows:

- Fetch of Course *A*
  - Retrieval of Result for Course *A*
  - — Add retrieved result for Course *A* to TotalCredit
  - — Call Function Credit progression Year 1
- Fetch of Course *B*
  - Retrieval of Result for Course *A*
  - — Add retrieved result for Course *B* to TotalCredit
  - — Call Function Credit progression Year 1

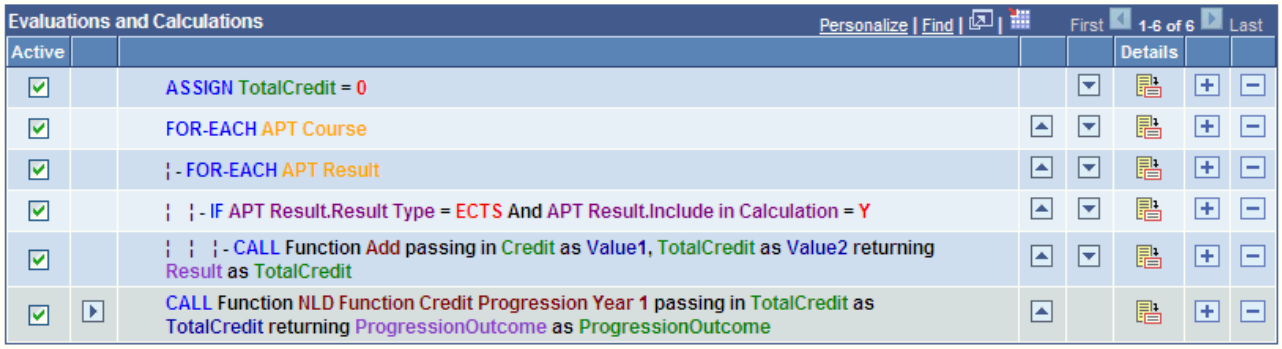
The evaluation of academic progression takes place each time a result is retrieved. This is not what you want the Rule to do. What you want is for the academic progression evaluation to take place after all the course credit have been added up:

- Fetch of Course *A*
  - Retrieval of Result for Course *A*
  - — Add retrieved result for Course *A* to TotalCredit
- Fetch of Course *B*
  - Retrieval of Result for Course *A*
  - — Add retrieved result for Course *B* to TotalCredit

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

- Call Function Credit progression Year 1

Use the remove indentation field to place the statement in the correct place:



Example of Academic Progression Rule Text with Progression Function Correctly Indented

Building and Testing Rules

This section discusses how to:

- Build Rules.
- Test Rules.
- Create a new version of a Rule.
- View the Page Process Flow.
- View Rule Cross-references.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Pages for Building and Testing Rules

Page Name	Definition Name	Navigation	Usage
Rule Builder	SCC_RULE_BUILDER	Set up SACR, System Administration, Rules Engine, Rules Engine Manager, select Search for a Rule, Define Rule, select an Action of Build Rule	Select Rules to build.
Build Rules	SCC_RULE_REBUILD	Set up SACR, System Administration, Rules Engine, Setup, Build Rules	Search for Rules to build.
Rules Engine Tester	SCC_RULE_REBUILD	Set up SACR, System Administration, Rules Engine, Rules Engine Manager, select Search for a Rule, Define Rule, select an Action of Test Rule	Test a Rule.
Rules Engine Manager Version History	SCC_RULE_VERSION	Set up SACR, System Administration, Rules Engine, Rules Engine Manager, select Search for a Rule, Version History tab	View and manage Version History for a Rule.
Rules Engine Manager Cross Reference	SCC_RULE_XREF	Set up SACR, System Administration, Rules Engine, Rules Engine Manager, select Search for a Rule, Cross Reference tab	View the other Rules where this Rule is used.
Rules Engine Tester	SCC_RULE_REBUILD	Set up SACR, System Administration, Rules Engine, Rules Engine Manager, select Search for a Rule, Define Rule, select an Action of Test Rule	Test a Rule.

Building Rules

Before a Rule can be tested or used, the code in the Rule needs be converted to executable code. This is done by the Rules Engine Build code compilation process. This process can be initiated:

- From an individual Rule using the Rule Action *Build* on the Define Rule page.
- In batch using the Build Rules process, where you can build multiple Rules simultaneously.

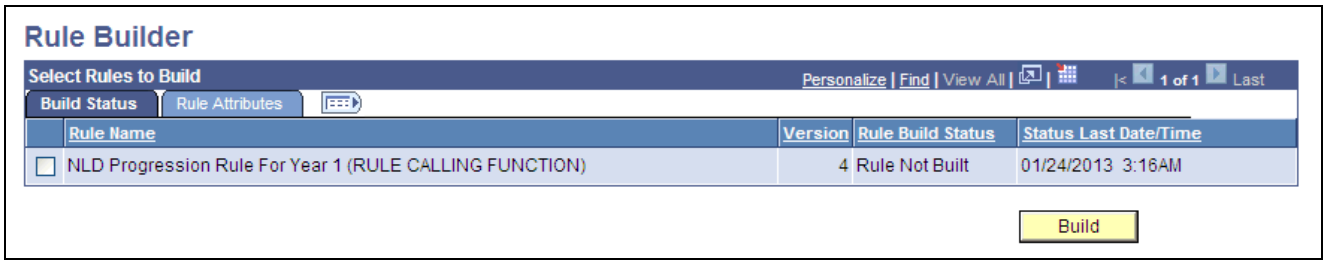
Building a Rule from the Define Rule Page

Access the Define Rule page (Set up SACR, System Administration, Rules Engine, Rules Engine Manager, select Search for a Rule, Define Rule).

Select the *Update Status Information* Action to display the current Rule Build Status.

Then, select the *Build* Action to open the Rule Builder page, select the Rule, and click the Build button to compile the Rule:

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

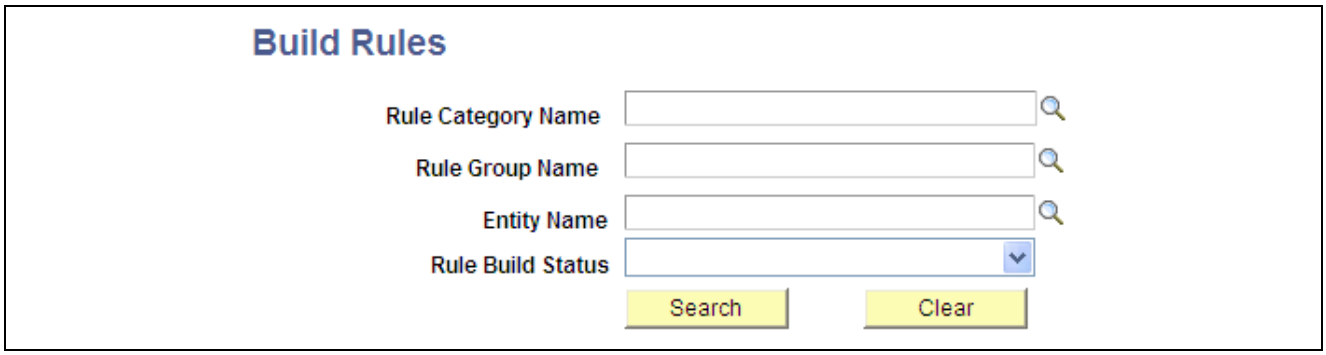


Rule Builder Page

If the Rule Build Status is updated to *Build Successful*, you may proceed to testing the Rule. Otherwise, consult your Rule for Current Logic. A flaw in Rule Logic may be the issue.

Building Multiple Rules at the Same Time

Access the Build Rules page (Set up SACR, System Administration, Rules Engine, Setup, Build Rules).



Build Rules page

Select any search parameter fields you want to use to search for Rules you want to build and click the Search button. If you click the Search button without making any parameter selection, all Rules are returned.

Select the Rules you want to build and click the Build button.

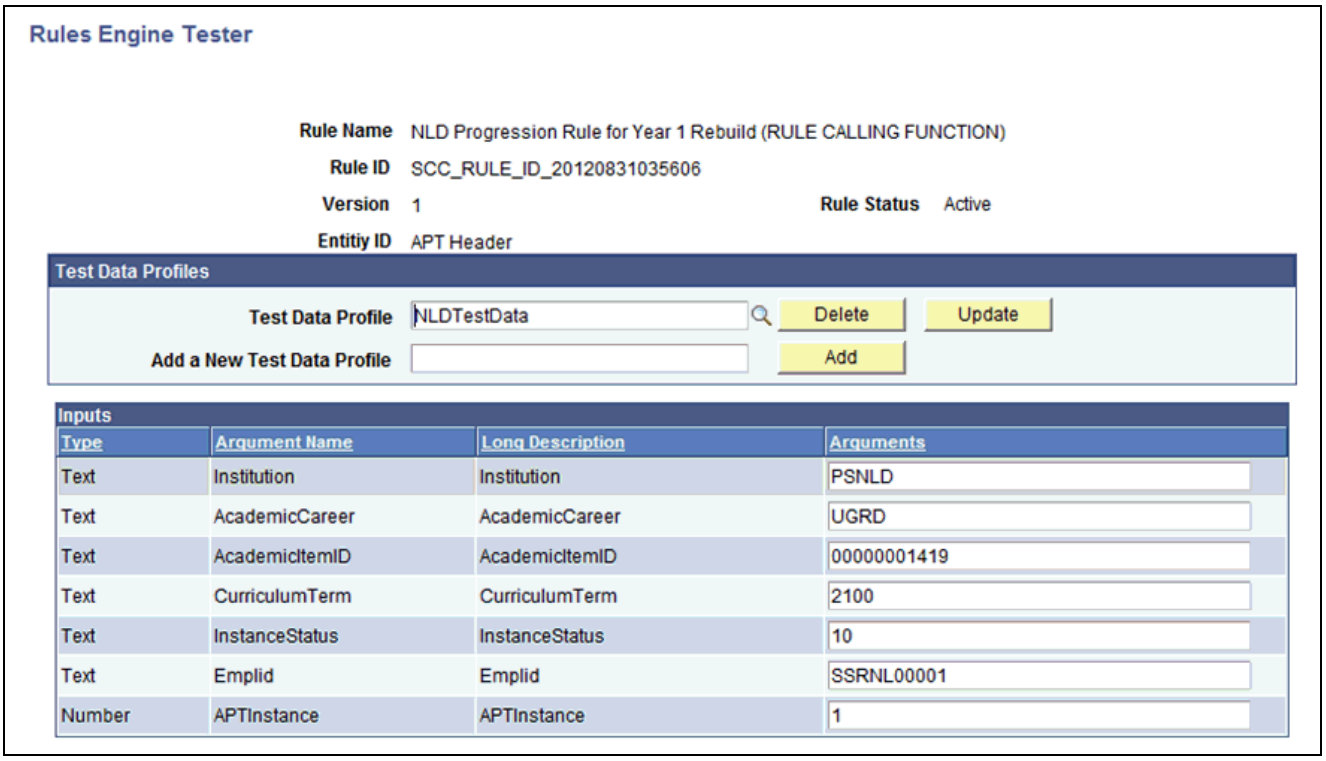
Testing Rules

After a Rule is built successfully, you can test the Rule.

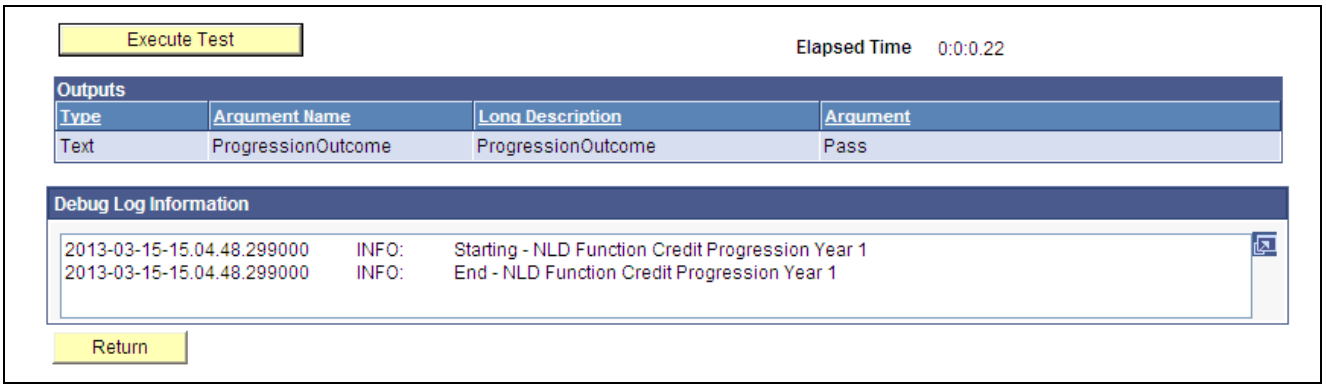
Access the Define Rule page (Set up SACR, System Administration, Rules Engine, Rules Engine Manager, select Search for a Rule, Define Rule).

Select the *Test Rule* Action to open the Rules Engine Tester page:

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Rules Engine Tester page (1 of 2)



Rules Engine Tester page (2 of 2)

Enter the required Arguments that are needed to test the Rule. After test parameters have been added the Parameters can be saved as a Rule Test Profile.

Test Data Profile	Select a saved Test Data Profiles.
Delete	Click this button to delete a stored Test Profile.
Update	Click this button to update a stored Test Profile with new parameter.
Add a New Test Data Profile	Enter a name if you want to save a new Test Data Profile and click the Add button.
Inputs	Displays all Parameters/Variables which have been listed as Input.
Arguments	Enter the required Arguments needed to test the Rule.
Execute Test	Select the button to execute the Rule. If a Rule needs parameters and none are provided, the Rule cannot be executed.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Outputs Displays all output Parameters/Variables with the output result when the Rule executes successfully.

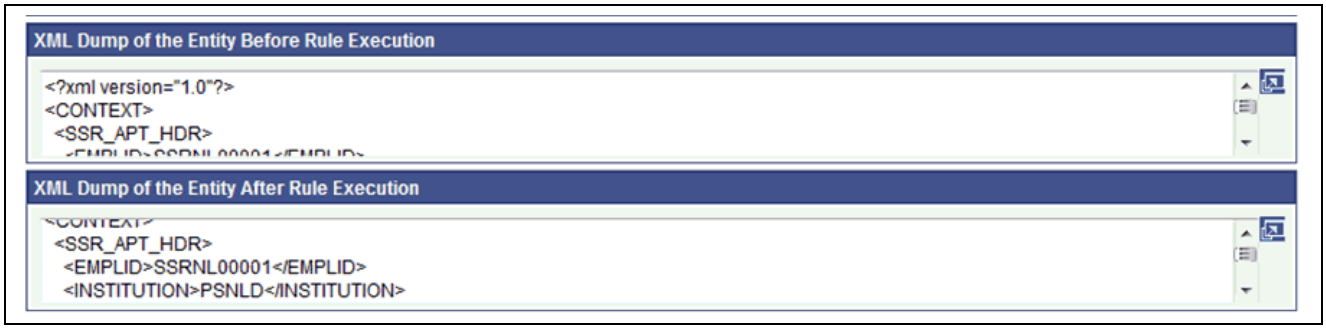
Debug Log Information Displays debug information is shown if the Logging Level is set on the Define Rule page. The Debug Log Information shows the execution logic of the Rule in order of execution.

Refer to the "Starting Query" for information about how the Base Entity is selected and with which criteria the selection is done. If this is a Rule, a Select statement is generated based on the underlying tables associated with the Base Entity attached to the Rule. The Criteria attached to the Rule is used to create the Starting Query's Criteria.

Refer to the individual debug statements to view what information was selected per statement. The timestamp option shows the elapsed time.

Return Select this button to return to the Define Rule page.

XML Dumps of the Entity, before and after Rule execution, are also displayed:



Example of XML Dump in Rule Tester

XML Dump of the Entity Before Execution Displays all data selected for the Base Entity and made available to the Rule.

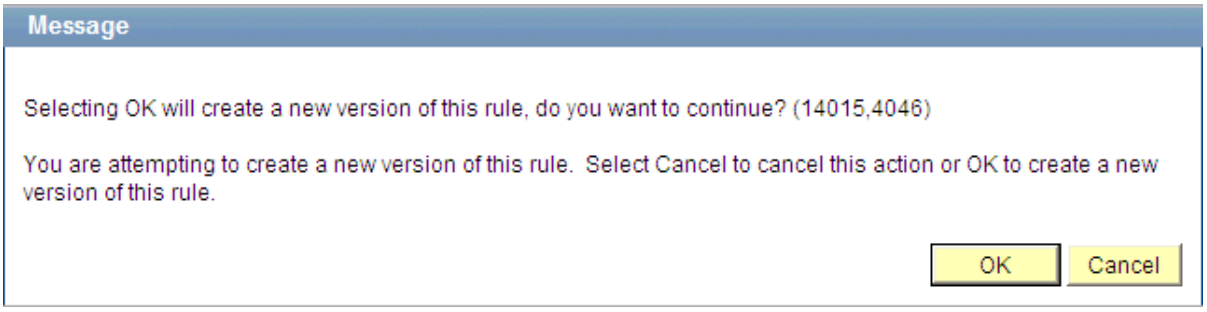
XML Dump of the Entity after Execution Displays all current data for the Base Entity after the Rule is executed. This information is relevant in instances where the Rule is used to update or insert data in the database.

Creating a New Version of a Rule

Active Rules can be referenced by other Rules or by online triggers and/or batch processes. In a production environment, it should not be possible to change anything in an active Rule. Changing an Active Rule can be disruptive to business processes. Therefore, Rule changes should be made by creating a new version of an Active Rule. Only one Active version of a Rule exists for any Rule at any given time. Since Rules are referenced by other systems using the Rule ID, it is always the Active version of the Rule that is executed. Use the *Create new Version of Rule* Action on the Define Rule page to create a new version of an Active Rule.

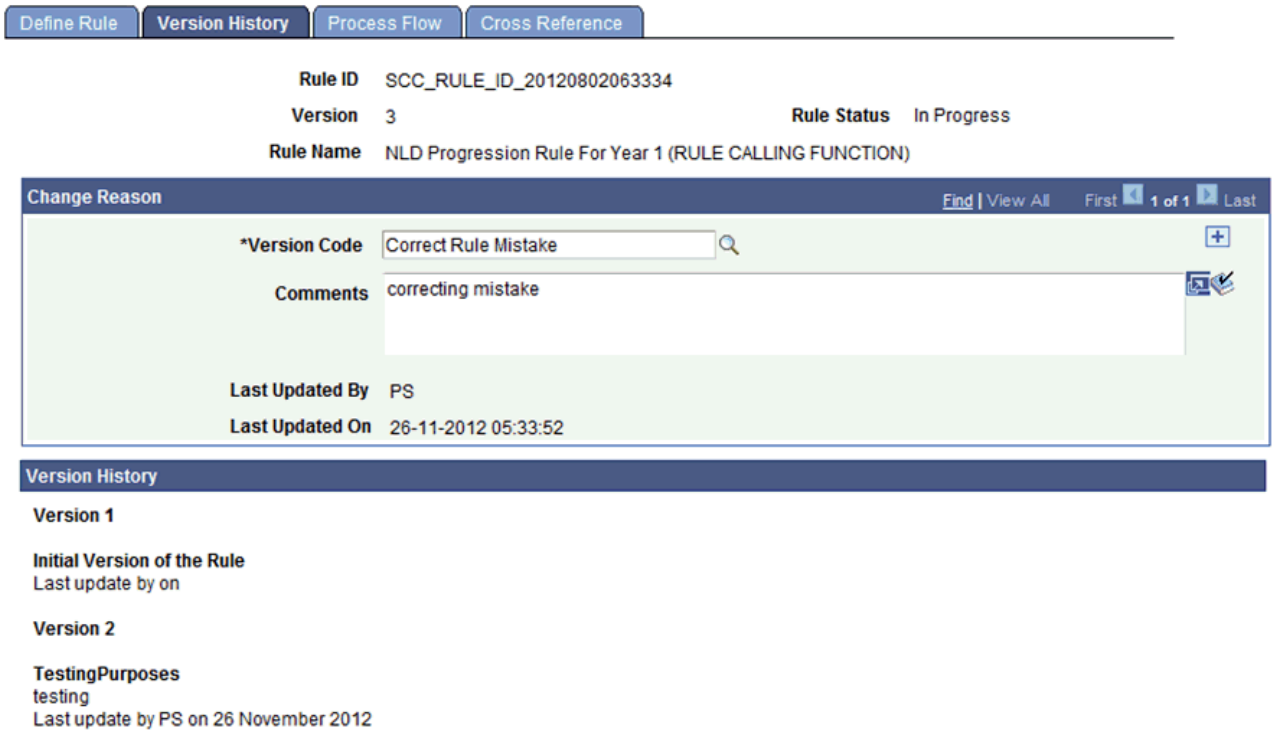
After the option Create new version of Rule has been selected the user is shown a warning message:

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Example of Create New Rule Warning Message

After clicking OK on the warning message, the Version History page opens:



Version History page

Note. You may also access the Version History by navigating to Set up SACR, System Administration, Rules Engine, Rules Engine Manager, Version History.

Older versions of the Rule are shown as Version History. Any Version Codes and Comments are displayed by Version. You can still ignore the new Rule Version created by cancelling out of the page without saving.

After saving the Rule Version, an Operator ID and Date Time stamp are automatically updated, and the status of the new Version is *In Progress*. In the example illustrated above, . Version 2 remains *Active* until Version 3 is activated. When activating Version 3, Version 2 becomes Inactive automatically.

Note. It is not possible to reactivate “old” versions of Rules.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Viewing Rule Cross References

Access Rules Engine Manager Cross Reference page (Set up SACR, System Administration, Rules Engine, Rules Engine Manager, Cross Reference).

Define Rule | Version History | Process Flow | Cross Reference

Rule ID SCC_RULE_ID_20120801120231

Version 1 **Rule Status** Active

Rule Name NLD Function Credit Progression Year 1

Rule Cross-Reference Personalize | Find | View All | First 1-10 of 15 Last

NLD Progression Rule For Year 1 (RULE WITH SCHEDULE CHECKS)	Version	Rule Status	Rule Category Name	Occurances
NLD Progression Rule For Year 1 (RULE WITH SCHEDULE CHECKS)	2	Inactive	NLD Demo Rules	1
NLD Progression Rule For Year 1 (RULE CALLING 1 RULE AND 1 FUNCTION)	1	Active	NLD Demo Rules	1
NLD Progression Rule For Year 1 (RULE CALLING FUNCTION)	1	Inactive	NLD Demo Rules	1
NLD Progression Rule For Year 1 (RULE CALLING FUNCTION)	2	Inactive	NLD Demo Rules	1
NLD Progression Rule For Year 1 (RULE CALLING FUNCTION)	3	Active	NLD Demo Rules	1
NLD Progression Rule for Year 1 Rebuild (RULE CALLING FUNCTION)	1	Active	NLD Demo Rules	1
NLD Progression Rule for Year 1 Rebuild (RULE WITH SCHEDULE CHECKS)	1	Active	NLD Demo Rules	1
NLD Progression Rule for Year 1 Rebuild (RULE CALLING FUNCTION- CE without Save)	1	Active	NLD Demo Rules	1
NLD Progression Rule for Year 1 Rebuild (RULE CALLING FUNCTION- CREATE ENTITY) 1	1	Active	NLD Demo Rules	1
NLD Progression Rule for Year 1 Copy (RULE CALLING FUNCTION- CREATE ENTITY)	1	In Progress	NLD Demo Rules	1

Cross Reference page

A Rule Cross Reference page is available which shows all Rules which are referencing the current Rule via a call statement. Click the link to navigate directly to the referenced Rule. If there no other Rules reference the current Rule, a message is displayed: “This Rule is not used by another Rule.”

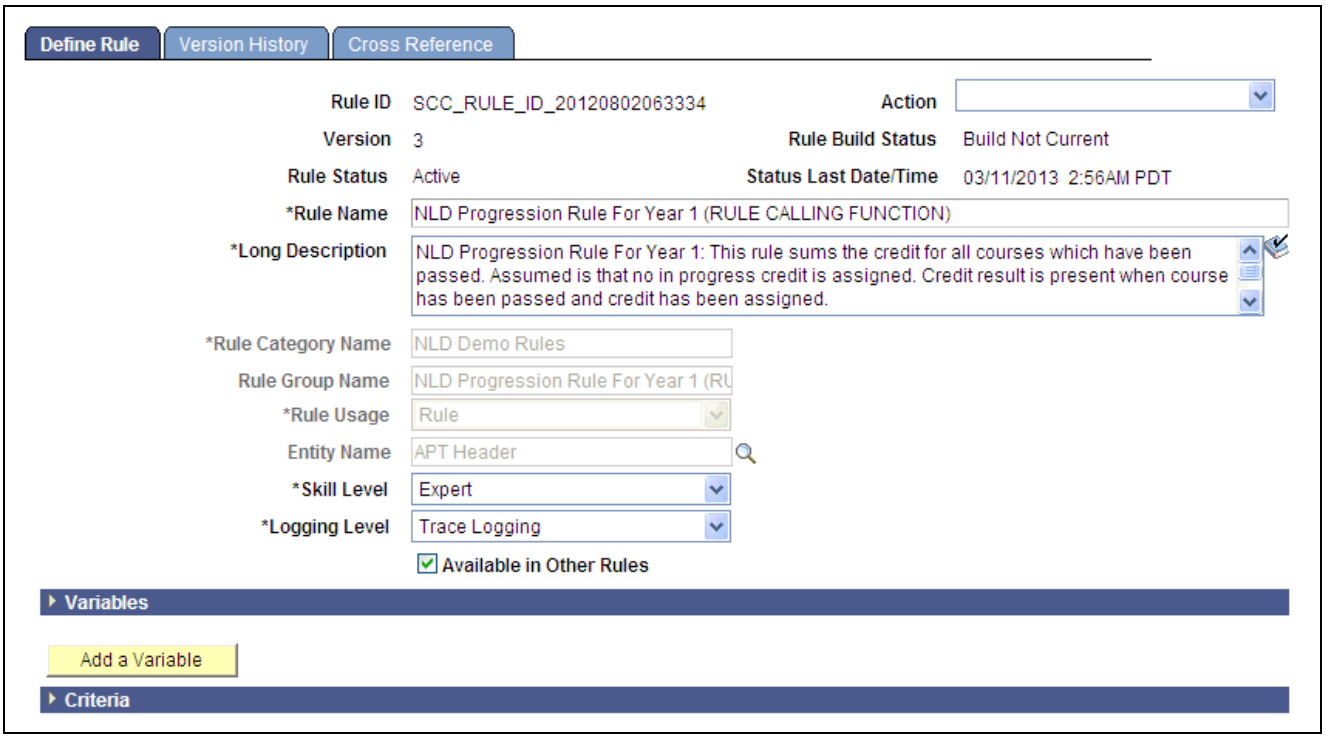
Understanding Contextual Referencing

The Rules Engine includes a feature called Contextual Reference. This feature ensures that when Functions are called by other Rules (Calling Rules), the Entity of the Calling Rule is passed to the called Function.

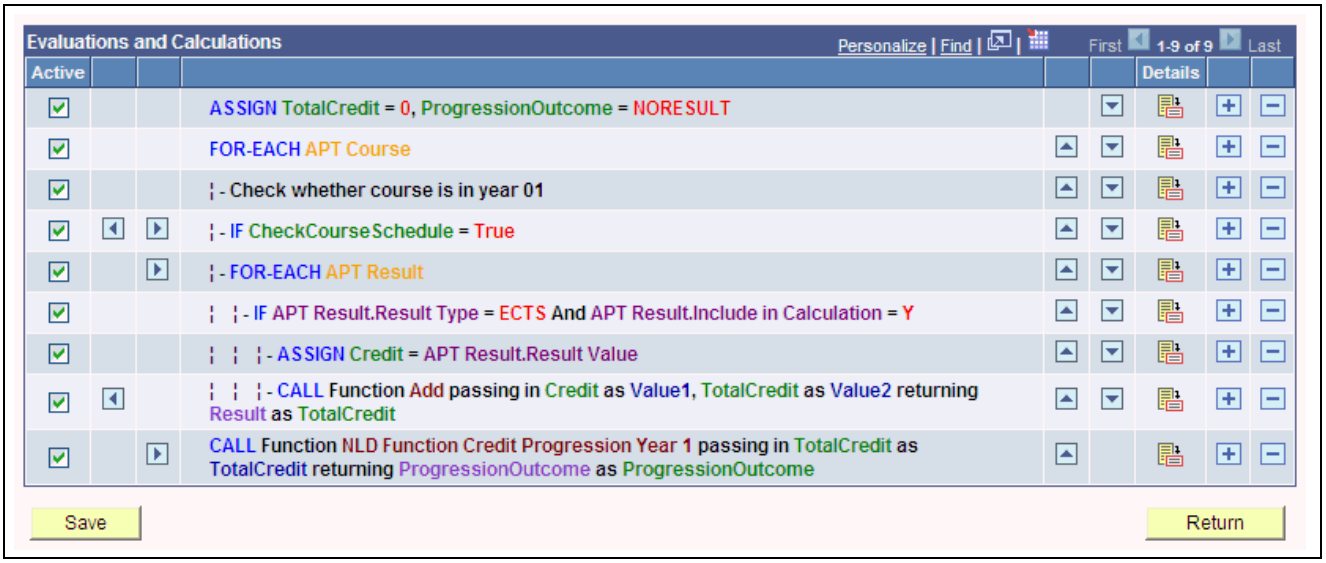
The following is an example of a Rule using Contextual Referencing which does the following:

1. Select all relevant Courses.
2. For each Course, determine whether en enrollment record exists in Year 1 of the academic program according to the APT schedule record using user defined Function “NLD Function Year of Program Course”. The system generated text has been overridden to show “Check whether Course is in Year 01”
3. Select the appropriate result which is stored in the result record as Result Type “ECTS”.
4. Use the result value obtained in the previous step to calculate the sum of credit using delivered math Function “Add” in Category “Math”.
5. Evaluate whether total credit is enough to progress to year 2 of the academic program using user defined Function “NLD Function Credit Progression Year 1”.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Progression Rule for Contextual Reference Example (1 of 2)



Progression Rule for Contextual Reference Example (2 of 2)

The Functions used to perform summing of Total Credit (Steps 4 and 5) receive either an Input value of type Variable or Property by calling Rule “NLD Progression Rule for Year 1 (RULE CALLING FUNCTION)”. An input Variable is all that these Functions need to execute the required processing logic. Shown below is an example of the custom Function “NLD Function Credit Progression Year 1”. The Function performs a series of evaluative Statements using the single input variable TotalCredit and passes back the result into variable Progressionoutcome. The Function has no Entity of its own and is without criteria.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Define Rule | Version History | Cross Reference

Rule ID	SCC_RULE_ID_20120801120231	Action	
Version	1	Rule Build Status	Build Successful
Rule Status	Active	Status Last Date/Time	03/15/2013 3:03PM PDT
*Rule Name	NLD Function Credit Progression Year 1		
*Long Description	NLD Progression Rule For Year 1: >= 60 then Pass Between 42 and 60 Conditional Pass		
*Rule Category Name	NLD Demo Rules		
Rule Group Name			
*Rule Usage	Function		
Entity Name			
*Skill Level	Expert		
*Logging Level	Informational Messages		

☑ Available in Other Rules

Variables — Personalize | Find | View All | First 1-2 of 2 Last

Argument Name	Type	List	System	Input	Output	Required	Count	Details
TotalCredit	Number	☐	☐	☑	☐	☐	4	
ProgressionOutcome	Text	☐	☐	☐	☑		5	

Add a Variable

Progression Function for Contextual Reference Example (1 of 2)

Evaluations and Calculations — Personalize | Find | First 1-13 of 13 Last

Active				Details		
☑	IF TotalCredit >= 60					
☑	¦ - ASSIGN ProgressionOutcome = Pass					
☑	ELSE *IF TotalCredit >= 60*					
☑	¦ - IF TotalCredit >= 42					
☑	¦ ¦ - ASSIGN ProgressionOutcome = ConditionalPass					
☑	¦ - ELSE *IF TotalCredit >= 42*					
☑	¦ ¦ - IF TotalCredit >= 36					
☑	¦ ¦ ¦ - ASSIGN ProgressionOutcome = RepeatYear1					
☑	¦ ¦ - ELSE *IF TotalCredit >= 36*					
☑	¦ ¦ ¦ - IF TotalCredit < 36					
☑	¦ ¦ ¦ ¦ - ASSIGN ProgressionOutcome = Fail					
☑	¦ ¦ ¦ - ELSE *IF TotalCredit < 36*					
☑	¦ ¦ ¦ ¦ - ASSIGN ProgressionOutcome = Undetermined					

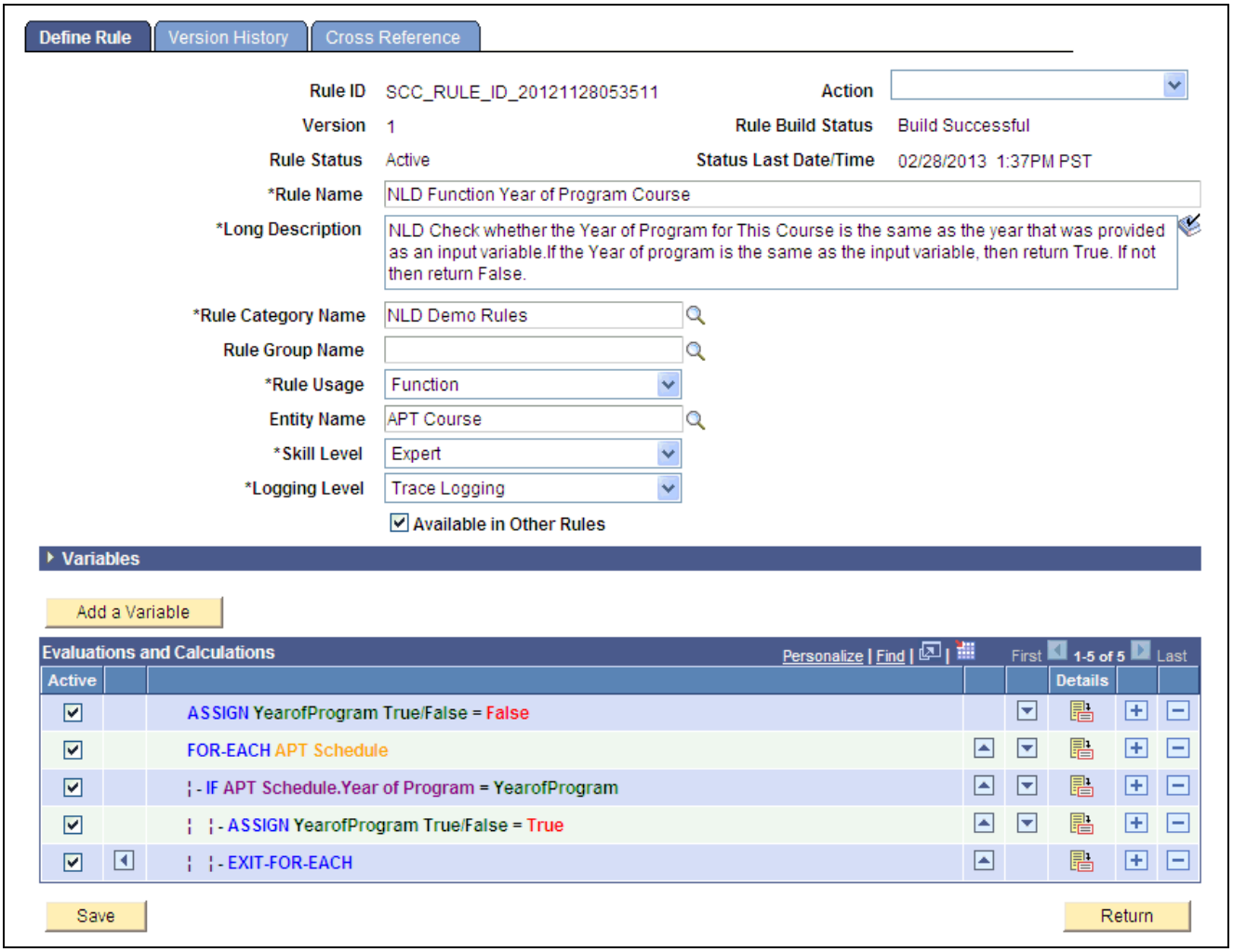
Save Return

Progression Function for Contextual Reference Example (2 of 2)

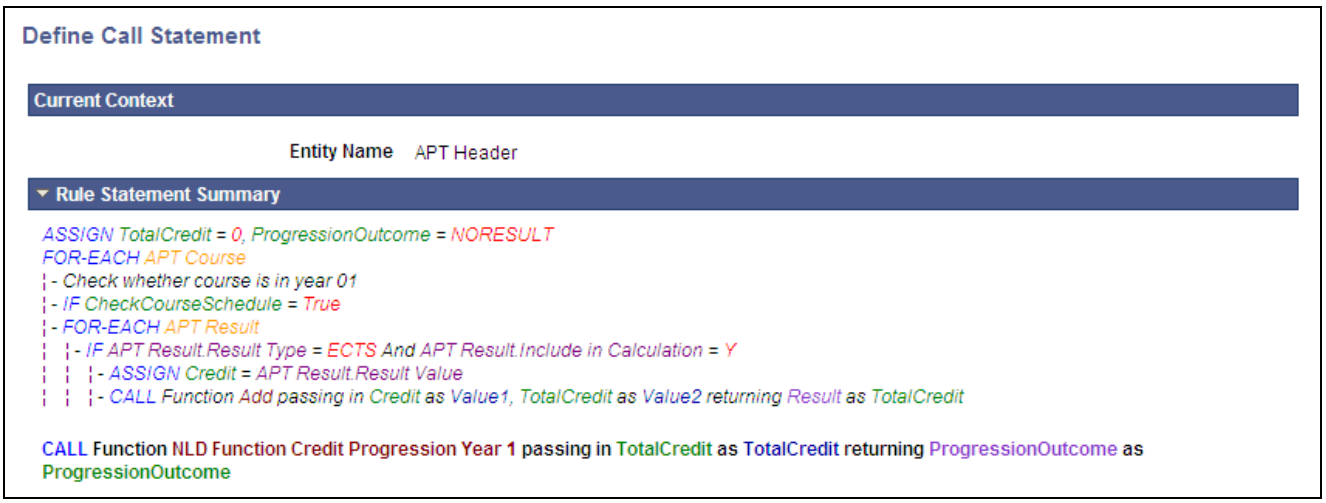
Next is an example of the Function “NLD Function Year of Program Course”. This Function does have its own Base Entity “APT Course”, but Input and Output parameters are defined in the Call Statement shown after the Function example. The criteria identifying the exact Course are not explicitly passed from the Calling Rule to the Called Function.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

However, when the Function is called, the exact Course information is passed from the calling Rule to the called Function by virtue of Contextual Reference. This is because the called Function's Base Entity is APT Course which is the same context that the Calling Rule is working from when retrieving Courses using the FOR-EACH loop.

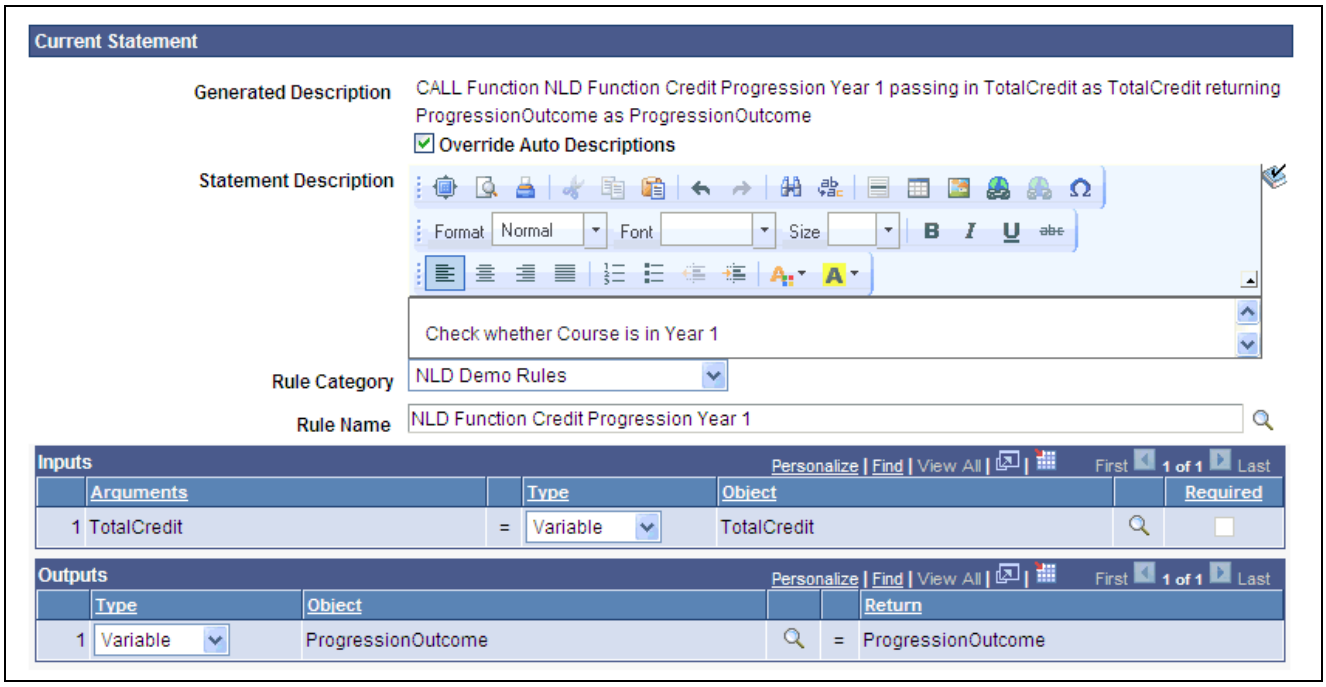


Program Course Function for Contextual Reference Example



Call Statement for Contextual Reference Example (2 of 2)

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.



Call Statement for Contextual Reference Example (2 of 2)

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.